Microsoft®

Microsoft® Internet Information Services 5.0 Resource Guide

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 2000 by Microsoft Corporation

Library of Congress Cataloging-in-Publication Data
Microsoft Windows 2000 Server Resource Kit / Microsoft Corporation.
p. cm.
ISBN 1-57231-805-8
1. Microsoft Windows 2000 server. 2. Operating systems (Computers) I. Microsoft Corporation.
QA76.76.O63 M5241328 2000
005.4'4769--dc21 99-045616
CIP

Printed and bound in the United States of America.

6 7 8 9 10 QWT 7 6 5 4

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *rkinput@microsoft.com.*

Macintosh, Power Macintosh, and QuickTime are registered trademarks and the Mac OS Logo and the QuickTime Logo are trademarks of Apple Computer, Inc. used under license. Active Directory, ActiveX, Authenticode, BackOffice, DirectX, FrontPage, JScript, Microsoft, Microsoft Press, MS-DOS, NetShow, Sidewalk, SQL Server, Visual Basic, Visual C++, Visual InterDev, Visual J++, Visual Studio, Win32, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person, or event is intended or should be inferred.

Acquisitions Editor: Juliana Aldous
Project Editor: Maureen Williams Zimmerman
Part No. 097-0002130

Contributing Writers:

Seth Manheim ("Introduction"); Curt Johnson (Chapter 1, "Overview of Internet Information Services 5.0"); Megan Davis (Chapter 2, "Managing the Migration Process," Chapter 3, "Migrating a Web Server to IIS 5.0"); Jon Singer (Chapter 4, "Capacity Planning," Chapter 5, "Monitoring and Tuning Your Server"); John Meade, John Sudds (Chapter 6, "Developing Web Applications," Chapter 7, "Data Access and Transactions"); Curt Johnson, Shawn Bice (Chapter 8, "Administering an ISP Installation"); Michael Howard (Chapter 9, "Security"); John Meade (Chapter 10, "Access to Legacy Applications and Data," Appendix A "ASP Best Practices," Appendix B "Site Security Planning")

Resource Kit Program Managers: Noland Angara, Seth Manheim

Documentation Manager: Seth Manheim

Editing Manager: Karen Carter-Schwendler

Lead Editor: Susan Joly

Contributing Editor: Rebekka Stahl

Copyeditor/Production Support: Christy Jobe

Production Lead: Stacia Scott

Indexer: Patsy Rae Dawson

Design Lead: Scott Pickle

Graphic Artist: Peter Ogden

Technical Consultants:

Bilal Alam, Bill Andreozzi, Aaron Barth, Kresimir Bozic, Paolo Branchesi, Gabe Bratton, Barry Butterklee, Philip Carmichael, Kevin Chen, Quentin Clark, Matthew Connelly, Nicholas Criss, Rich Demar, Ryan Donovan, Jaroslav Dunajsky, Peter Durham, Okechukwu Echeruo, John Edwards, Jeff Fink, Amar Gandhi, Matthew Gibbs, Geoff Gray, Sonali Gururaja, Wade Hilmo, Lei Jin, John Jones, Stuart Klingman, Murali Krishnan, David LeBlanc, Eric Lee, John Ludeman, Richard Maring, Robert McMurray, Mike Moore, Jim Morey, Jamie Myxter, Eric Nace, Eric Niebler, Mark O'Brien, Matt Odhner, Jee Fung Pang, Kestutis Patiejunas, Sam Patton, Steve Peschka, Seth Pollack, Lenn Pryor, George Reilly, Allison Robin, Dmitry Robsman, Valerie See, Alan Shen, Martin Sleeman, Paul Spurgeon, Michael Stephenson, Adam Stone, Luc Talpe, Michael Thomas, Van Van, Todd Wanke, Lester Waters, Todd Weeks, and Stuart Wiley.

Contents

Introduction ..... **xix**
About the Internet Information Services Resource Guide xx
Resource Kit Compact Disc xxi
Resource Kit Support Policy xxi

CHAPTER 1 Overview of Internet Information Services 5.0 ..... **1**
What's New in IIS 5.0 2
Security 2
Administration 3
Programmability 3
Internet Standards 4
IIS 5.0 Architecture 5
Architecture Overview 5
Microsoft Management Console 6
Active Server Pages and Microsoft Script Debugger 6
Indexing Service 7
Certificate Services 7
Message Queuing 8
Database Access Component 8
Administrative Architecture 9
IIS Administration Objects 9
Internet Services Manager 10
Built-in and Customized Scripts 10
Programmability Architecture 11
Common Gateway Interface 12
ISAPI Filters 13
ISAPI Extensions 13
Active Server Pages 14
Active Scripting 14
Component Services 15

Publishing on Web Sites 17
WebDAV 17
FrontPage Server Extensions 18
FTP 19
Additional Resources 19

CHAPTER 2 Managing the Migration Process 21

Migration Process Overview 22
1. Envisioning 24
Define the Project 25
Create a Requirements Definition 26
Develop a Conceptual Design 27
Assess Risk 28
Define the Project Team 29
2. Planning 31
Team Roles during the Planning Phase 32
Gather Information 32
Server and Network Environment 32
Tools and Utilities in Use 33
Users 34
Standards 35
Define the New Service Offering 36
Functional Specification 36
Solution Prototype 38
Assess Resource Needs 38
Staff 38
Migration Tools 39
Server Software 40
Hardware 40
Build the Master Project Plan 42
Draft the Project Schedule 43
Check Your Assumptions 44
3. Developing 44
Team Roles During the Developing Phase 45
Validate the Design 45
Unit Testing 47
Integration Testing 48
Application Testing 49

Build out the System . . . 49
Begin Training . . . 50
Conduct Pilot Testing . . . 50
4. Deploying . . . 51
Team Roles During the Deploying Phase . . . 52
Finish Training . . . 52
Roll out the New System . . . 53
Monitor the System . . . 54
Additional Resources . . . 55
Support Documents . . . 55
IIS 5.0 Migration Tools . . . 56
IIS Migration Wizard . . . 56
Microsoft Interoperability Lab . . . 56
Other Migration Tools . . . 56
Migration and Integration Resources . . . 56
Planning and Testing Resources . . . 57
Test Tools . . . 58
General Administration Books and Training . . . 58

CHAPTER 3 Migrating a Web Server to IIS 5.0 . . . 59

Basic Steps to Migrate a Web Server . . . 60
Assessing Hardware Requirements . . . 61
Preparing the Destination Server . . . 63
Using the IIS Migration Wizard . . . 67
Migrating Web and FTP Sites . . . 67
Replicating Windows-Based Files . . . 67
Replicating UNIX-Based Files . . . 69
Completing Web Site Migration . . . 72
Completing FTP Site Migration . . . 74
Replicating and Configuring Applications . . . 74
Migrating Log Files . . . 75
Migrating Configuration Settings . . . 75
Securing the Server . . . 77
Migrating Users and Groups . . . 77
Setting NTFS Permissions . . . 78
Setting IIS 5.0 Permissions . . . 79
Setting Permissions Based on Content . . . 80
Migrating Security Certificates . . . 82
Integrating UNIX and Windows 2000 Server Security . . . 82

Migrating from Apache HTTP Server 83
Comparing Apache and IIS 5.0 83
Administration Interface 83
Security 84
User Directory 84
Virtual Host 84
Alias/Directory Alias 85
Custom Error Messages 85
Redirects 85
Migrating Apache Directives 86
Server Directives 86
Resource Configuration 92
Access Configuration 96
Migrating Custom Modules 98
Migrating from Netscape Enterprise Server 98
Comparing NES and IIS 5.0 98
Terminology 98
Migrating NES Configuration Settings 99
Server Preferences 100
Applications 102
Server Status 102
Configuration Styles 104
Content Management 104
Web Publishing 106
Agents and Search 106
Auto Catalog 106
Upgrading or Replicating an IIS Web Server 107
Choosing an Approach 107
Recommendations for Upgrading or Replicating 109
Migrating Web Applications 110
IIS 5.0 Application Technologies 110
Deciding to Port or Rewrite CGI Applications 112
Performance vs. Development Work 112
Effort to Develop and Maintain 113
Porting CGI Applications 114
Configuring a Script Interpreter 114
Special Considerations for UNIX Applications 116
Migrating a UNIX Perl Script 117
Converting UNIX Application Files 118

Converting CGI to ISAPI . . . 119
Migrating from CGI to ASP . . . 120
ASP Scripting Support . . . 120
Analyzing the Application . . . 120
Input Processing . . . 121
Business Logic . . . 124
External Gateway and Database Logic . . . 124
Maintaining State . . . 126
Output Handling . . . 127
The File System . . . 130
Reproducing Common CGI Services . . . 131
Electronic Mail Delivery . . . 131
Page Counters . . . 134
Additional Resources . . . 134
IIS 5.0 Migration Tools . . . 134
IIS Migration Wizard . . . 134
Application Migration Tools . . . 135
Server Administration and Interoperation Tools . . . 136
Reference Books . . . 137
Security Resources . . . 137

CHAPTER 4 Capacity Planning . . . 139

Capacity Planning Issues . . . 140
Traffic . . . 140
Considerations . . . 146
Secure Sockets Layer . . . 147
Web Application Performance . . . 148
Reliability . . . 150
Server Clustering . . . 150
Network Load Balancing . . . 152
Determining Your Installation's Requirements . . . 153
A Capacity Planning Checklist . . . 154
Capacity Planning Scenarios . . . 156
The Intranet Web Site . . . 157
The Internet Marketing Web Site . . . 159
The Internet Transactional Web Site . . . 161
The Internet Commerce Web Site . . . 163

A Large-Site Case Study: microsoft.com . . . 166
A Snapshot of the Site . . . 166
Summary . . . 173
General Guidelines . . . 173
Additional Resources . . . 174
Web Links . . . 174
Books . . . 174

CHAPTER 5 Monitoring and Tuning Your Server . . . 175

Using This Chapter . . . 176
Memory . . . 177
Monitoring Overall Server Memory . . . 180
Monitoring the File System Cache . . . 189
Preventing Processor Bottlenecks . . . 195
Suggestions for Improving Processor Usage and Performance . . . 204
Network I/O . . . 206
Network Bandwidth Requirements of a Server Running IIS 5.0 . . . 206
Monitoring the Network Connection . . . 206
Optimizing the Network Connection . . . 215
Disk I/O . . . 217
Monitoring a Remote Computer . . . 217
Web Applications . . . 218
Tuning the ASP Queue and Thread Pool . . . 218
How This Affects Server Administration . . . 219
Optimizing for Web Applications . . . 220
Short-Term Problems and Temporary Fixes . . . 221
Tuning Tips . . . 221
Monitoring Security Overhead . . . 222
Measuring Security Overhead with WCAT . . . 223
Tools . . . 228
The System Monitor . . . 228
Performance Counter Check . . . 230
The HTTP Monitoring Tool . . . 230
NetStat and NetMon . . . 230
Process Viewer, Process Explode, Process Monitor, and Event Viewer . . . 231
Web Application Stress Tool and WCAT . . . 231
Getting Started with the Web Application Stress Tool . . . 231
General Guidelines for Using the Web Application Stress Tool . . . 238
Suggestions . . . 238

Useful Counters for Stress Testing . . . 239
Suggested Values . . . 244
Examining the Results . . . 245
Baseline Logging . . . 245
Additional Resources . . . 246
Web Links . . . 246
Books . . . 246
Tools . . . 246

CHAPTER 6 Developing Web Applications . . . 247

Building on Client/Server . . . 248
Client/Server Revisited . . . 248
Multi-Tier Design . . . 249
Windows DNA . . . 251
The Future of Applications on the Internet . . . 252
Client-Side Technologies . . . 252
Text and HTML . . . 252
Graphics and Multimedia . . . 253
Hyperlinks . . . 253
Client-Side Script . . . 253
ActiveX Controls . . . 254
Cascading Style Sheets . . . 256
Dynamic HTML . . . 256
Data Binding . . . 257
Browser Support . . . 258
Limitations of Client Technologies . . . 259
The Middle Tier . . . 259
CGI Applications . . . 260
ISAPI Extensions and Filters . . . 260
Active Server Pages . . . 261
ASP Server-Side Scripting . . . 262
Execution Behavior of Scripts in ASP Pages . . . 264
Built-in Objects and Server-Side Components . . . 266
Why Components? . . . 268
Building Windows Script Components . . . 269
ASP Applications . . . 269
ASP Session Management . . . 270
Building an ASP Application . . . 274
Selecting Object Scope . . . 279

Process Isolation and Crash Recovery 279
Applications and Processes 279
Configuring an Isolated Process 282
Out-of-Process Components 282
Security Considerations 284
A Note About Application Testing 284
Design Patterns for Web Applications 285
Factoring Your Application 285
Using Forms for Input 286
The Difference between GET and POST 287
Client-Side Form Validation 289
Hidden Form Fields 291
Redirection 292
Client-Side Redirection 294
Redirecting During Session_OnStart 295
Debugging Applications and Components 295
Script Debugging in Active Server Pages 296
Avoiding Common Mistakes 296
Debugging ASP 298
Script Management 301
Debugging ISAPI and Server Components 302
Disabling Debug Exception Handling 302
Debugging IIS 5.0 303
Inability to Create Components 305
Additional Resources 305
Web Links 305
Books 306

CHAPTER 7 Data Access and Transactions 307

Web Database Technologies 308
Why a Web Database? 308
Data Publishing Considerations 309
Industrial Strength Information 310
Microsoft Data Access Components 312
ODBC and OLE DB 313
ADO and RDS 314

Other Data Access Methods . . . 315
ADC . . . 315
Jet Database Engine and DAO . . . 316
RDO . . . 316
The Cost of Data Access . . . 317
Client-Side Data Access . . . 318
Client-Tier Elements . . . 319
Data-Aware Controls . . . 320
Data Cache . . . 324
Client Cursor Engine . . . 324
Business Object Proxies and the RDS.DataSpace Object . . . 325
Middle-Tier Elements of Client-Side Data Access . . . 326
IIS 5.0 and ADISAPI . . . 326
The RDS Data Factory and Custom Business Objects . . . 327
Designing Custom Business Objects . . . 327
Accessing Data with ASP and COM Components . . . 328
Preparing the Database . . . 329
Connection Strings . . . 329
Creating an ODBC-Compliant Data Source . . . 329
"DSN-less" Connections . . . 330
Selecting an OLE DB Provider . . . 331
Data Source Permissions . . . 332
Security and SQL Server . . . 332
The Database Connection . . . 335
ODBC Connection Pooling . . . 335
Tips for Optimizing Database Connections . . . 337
Recordsets and Cursors . . . 342
Forward-Only Cursors . . . 344
Static vs. Dynamic Cursors . . . 345
Keyset Cursors . . . 346
Cursor Concurrency . . . 346
Cursor Location . . . 347
Managing Records in a Recordset . . . 348
Keeping Track of New Records . . . 348
Avoiding Query Time-outs . . . 350
Purging Deleted Records . . . 350
References to Field Values . . . 350
VBScript Example: Filling a List Box . . . 351
PerlScript Example: Filling a Table . . . 352

Limiting the Number of Records . . . 353
Visual Basic Example: Paging through a Recordset . . . 353
Retrieving Image Data . . . 355
Stored Procedures . . . 356
Returning Values from Stored Procedures . . . 357
Prepared Queries . . . 358
Transaction Processing on the Web . . . 358
Transactions Explained . . . 359
Extending the Limits of Transactions . . . 360
Transactional ASP . . . 361
Business Objects vs. Script in ASP Pages . . . 362
Transactional Components . . . 362
Business Logic in Components . . . 362
Participating in Transactions . . . 364
Using Database Access Interfaces with Component Services . . . 366
Distribution and Scaling Issues . . . 367
Introducing Message Queuing . . . 368
Time-Independent Transaction Processing with Message Queuing . . . 369
Additional Resources . . . 371
Web Links . . . 371
Books . . . 372

CHAPTER 8 Administering an ISP Installation . . . 373

Configuring IIS 5.0 . . . 374
Creating Web Sites . . . 374
Creating a Company Web Site . . . 374
Creating a Personal Web Site . . . 379
Restricting Content . . . 381
Managing Your Installation . . . 382
Enhancing Reliability . . . 382
Replication and Clustering in IIS 5.0 . . . 383
Running Applications . . . 390
Recovering from Crashes . . . 396
Automating Administration . . . 397
Windows Script Host . . . 397
IIS Admin Objects and ADSI . . . 398
Executing Scripts . . . 398
Examples . . . 399

Administering a Site Remotely . . . 404
IIS Snap-in . . . 404
Internet Services Manager (HTML) . . . 404
Command Line . . . 405
Terminal Services . . . 405
Telnet . . . 406
Turning Users into Web Site Operators . . . 407
Configuring FrontPage Server Extensions . . . 407
Introducing the FrontPage Snap-in . . . 408
Managing Content . . . 409
Overlapping Virtual Servers . . . 411
Setting Security on an IIS 5.0 Server . . . 412
Setting Permissions . . . 413
Configuring E-mail . . . 421
Uploading Content through FTP . . . 424
Internet Connection Services for Remote Access . . . 425
Administering Older Version Servers . . . 426
Allocating Resources . . . 428
Logging Resources . . . 428
Controlling Bandwidth . . . 430
Process Accounting and Process Throttling . . . 431
Monitoring Performance with the HTTP Monitoring Tool . . . 433
Customizing Your Installation . . . 434
Hosting Multiple Sites with One IP Address . . . 434
Supporting Non–HTTP 1.1–Compliant Browsers . . . 438
Sending Cookies . . . 438
Components for Administration (Sample Setup) . . . 440
Making Redirects Work for You . . . 442
Setting Up Custom HTTP Headers . . . 443
Content Expiration Example . . . 443
Rating Content with PICS . . . 445
Customizing HTML Footers . . . 446
Building a Web Cluster . . . 446
Defining Clustering . . . 447
Defining Load Balancing . . . 447
Grouping Load Balancing Features . . . 448

Creating a Three-Tier Web Cluster . . . 448
Calculating Hardware Needs . . . 450
Building the First Tier . . . 450
Building the Second Tier . . . 452
Building the Third Tier . . . 452
Reviewing the Three Tiers . . . 453
Adapting IIS 5.0 to a Sample Installation . . . 453
Several Business Clients . . . 454
Three Networks in One . . . 455
Expanding the Platform: Considerations and Suggested Guidelines . . . 457
Some Notes on Security . . . 458
Additional Resources . . . 460
Web Links . . . 460

CHAPTER 9 Security . . . 461

Foundations of Computer Security . . . 462
Threats, Vulnerabilities, and Attacks . . . 464
Types of Attack . . . 465
The Bottom-Line Cost of Security . . . 466
Using the Built-in Security Features of Windows 2000 Server . . . 467
Configuring IIS 5.0 Security . . . 480
IIS 5.0 Authentication Modes . . . 480
Extending IIS 5.0 Security . . . 496
File and Directory Security . . . 498
Virtual Directory Security . . . 499
Secure Communications with SSL and TLS . . . 503
How Access Is Determined . . . 508
Troubleshooting Permissions . . . 510
An End-to-End Troubleshooting Example . . . 515
Defending Against Malicious Attacks . . . 522
Using the Security Templates . . . 523
Auditing Access with IIS 5.0 Logs . . . 524
Useful IIS Admin Objects/ADSI Security Settings . . . 524
IIS 5.0 Security Checklist . . . 525
Additional Resources . . . 527
Books . . . 527
Security Theory . . . 527
Firewalls and Proxy Servers . . . 527
System Intrusion . . . 528

General Security . . . 528
Encryption . . . 529
Windows NT Security . . . 529
General Networking . . . 529

CHAPTER 10 Access to Legacy Applications and Data . . . 531

Identifying Strategies . . . 532
Connecting to SNA . . . 533
Integrating IIS and Legacy Applications . . . 534
COM Transaction Integrator . . . 535
Functional Overview of the COM Transaction Integrator . . . 536
COMTI Development Scenarios . . . 536
Gaining Access to Legacy Data . . . 543
Legacy File Data and IIS 5.0 . . . 543
Access to VSAM and AS/400 files with OLE DB and ActiveX Data Objects . . . 544
Functional Overview of the Data Provider . . . 544
Replicating Legacy Databases . . . 548
Why Replication? . . . 548
Replicate Data Using Data Transformation Services . . . 549
Replicating DB2 Tables by using Host Data Replicator . . . 550
Two-Way Replication . . . 551
Flexible Processing and Filtering . . . 551
Scheduling . . . 551
Statistics . . . 551
Security . . . 552
Performance . . . 552
Supported Platforms . . . 552
Migrating Transaction Processing . . . 553
Why Use Transactions? . . . 553
Migrating to Component Services . . . 553
Features and Capabilities . . . 554
Additional Resources . . . 558
Web Links . . . 558
Books . . . 558
SNA Server 4.0 Software Product Documentation . . . 558

APPENDIX A ASP Best Practices **559**
When to Use ASP 560
Project Directories and Files 561
Organizing Application Directories and Files 561
Using File Name Extension Standards 564
Style Guide for Scripts in ASP Pages 565
HTML Standards 578
Scripting for Performance 581
Object and Variable Initialization 581
Working with Connections 583
Visual Basic Applications as DLLs 585
Additional Resources 586
Web Links 586
Books 586

APPENDIX B Site Security Planning **587**
Assessing Threats to Security 588
Threat Identification 588
Threats on Intranet 590
Threats over the Internet 592
Where to Spend the Effort 594
Why Is Security Difficult? 594
A Least-Access Approach 595
Making Policy 599
Vigilance and Revision 600
Adopting Technologies and Standards 602
Additional Resources 608
Web Links 608
Books 609

Glossary **611**

Index **661**

Introduction

Welcome to the *Internet Information Services Resource Guide*, a volume of the *Microsoft® Windows® 2000 Server Resource Kit*.

The *Internet Information Services Resource Guide* provides detailed information about Internet Information Services (IIS) version 5.0. This information is intended as a supplement to the online documentation included with IIS version 5.0. It does not replace that documentation as the primary source for learning how to use the product's specific features.

About the Internet Information Services Resource Guide

This book contains the following material:

Chapter 1, "Overview of Internet Information Services 5.0," introduces the new features of IIS 5.0, and gives an overview of the product's architecture.

Chapter 2, "Managing the Migration Process," is the first of two chapters about migrating to IIS. It provides general and conceptual information about planning and managing the migration process, when it involves multiple servers on an enterprise network.

Chapter 3, "Migrating a Web Server to IIS 5.0," provides instructions on migrating configuration settings, content, and applications to IIS 5.0 from another Web server, including UNIX-based Web servers. In addition, the chapter discusses upgrading IIS 4.0 to IIS 5.0, replicating an IIS 5.0 Web server, and using the IIS Migration Wizard. It also covers several approaches for migrating a Web application to IIS.

Chapter 4, "Capacity Planning," discusses issues involved in planning Web server installations, such as what equipment and software to acquire for Web sites. It also provides ways to determine where bottlenecks are likely to occur, and shows how much network bandwidth you will need.

Chapter 5, "Monitoring and Tuning Your Server," discusses issues involved in optimizing and tuning your IIS-based Web server, and covers some of the tools you can use for accomplishing these goals. It also provides some guidelines to help your server recover from bottlenecks.

Chapter 6, "Developing Web Applications," presents the features and benefits of the underlying technology in IIS 5.0, stressing the important functionality that can be obtained by creating Web applications.

Chapter 7, "Data Access and Transactions," introduces key components of Web data access, and discusses how to harness the power of a data-driven approach for Web content publishing.

Chapter 8, "Administering an ISP Installation," is a guide to running and maintaining an Internet service provider (ISP) installation in the context of Microsoft® Windows® 2000 Server and IIS 5.0.

Chapter 9, "Security," addresses how to configure a secure Web server that is running Windows 2000 Server and IIS 5.0, for use on the Internet or on an intranet.

Chapter 10, "Access to Legacy Applications and Data," discusses using the Web to make legacy data and applications easily accessible (via the Internet or a company intranet) to customers and members of the internal organization who use Web browsers. This chapter also describes how you can use IIS 5.0 for efficient utilization of legacy data and applications in Web solutions.

Appendix A, "ASP Best Practices," is an intranet standards planning guide that provides corporate guidelines for Active Server Pages (ASP) usage.

Appendix B, "Site Security Planning," describes how to set policies for securing Web resources, applications, and data in a company intranet. It also demonstrates, through scenarios, how to use IIS 5.0 in order to secure communications over the Internet, including e-commerce applications.

Glossary of Internet-related terms used in this book.

Resource Kit Compact Disc

The *Microsoft® Windows® 2000 Server Resource Kit* companion CD includes a variety of tools, components, and utilities to help you work more efficiently with IIS and ASP. Sample applications and the source code to the utilities are included that demonstrate how to write components, as well as IIS filters and applications. Documentation for each tool is included when you install from the CD.

Resource Kit Support Policy

The software supplied in the *Windows 2000 Server Resource Kit* is not officially supported. Microsoft does not guarantee the performance of the *Internet Information Services Resource Guide* tools, response times for answering questions, or bug fixes to the tools.

However, Microsoft does provide several ways for customers who purchase the *Windows 2000 Server Resource Kit* to report bugs and receive *possible* fixes for their issues. You can submit feedback on the *Windows 2000 Server Resource Kit* by sending e-mail to Rkinput@microsoft.com. This e-mail address is only for Resource Kit–related issues. For more general feedback on IIS 5.0 and on the *IIS Resource Guide*, and to report IIS 5.0 bugs, send e-mail to IISDocs@microsoft.com.

CHAPTER 1

Overview of Internet Information Services 5.0

Microsoft® Windows® 2000 Server's built-in Web server, Internet Information Services (IIS) 5.0, makes it easy to share documents and information across a company intranet or the Internet. IIS 5.0, the fastest Web server for Windows 2000 Server, is completely integrated with Microsoft® Active Directory™ directory service. This combination of Web and operating system services makes it possible to deploy scalable and reliable Web-based applications. The new generation of networked business solutions brings legacy data and applications to the World Wide Web, and lets companies redefine internal and external business processes.

IIS 5.0 introduces several new features to help Web administrators and Internet service providers (ISPs) create scalable Web applications, Web sites, and Web clusters. Advancements in IIS 5.0 Web publishing, security, administration, and applications work together to increase performance and reliability, while lowering the cost of ownership and improving the Web application environment.

Along with IIS 5.0, Microsoft offers several other products that you can add to your installation for added power and flexibility. For example, by adding Microsoft® Site Server, you can easily manage large clusters, customize logging, and create detailed reports.

In addition to introducing new features, this chapter gives an overview of IIS 5.0 architecture. By understanding how IIS 5.0 is put together, you can effectively manage Web sites and virtual directories, and configure applications so that they perform efficiently in an IIS 5.0 installation.

In This Chapter

What's New in IIS 5.0

IIS 5.0 Architecture

Publishing on Web Sites

Additional Resources

What's New in IIS 5.0

In IIS 5.0, you will find the following new features and improvements. For a complete list of new features, with detailed descriptions and procedures, see the IIS 5.0 online product documentation.

Security

This section briefly describes the new security features in IIS 5.0. For details about how security works, see "Security" in this book.

- **Digest Authentication** Adds security and reliability to user authentication across proxy servers and firewalls. IIS 5.0 still offers previous means of authentication: Anonymous, HTTP Basic, Windows NT Challenge/Response, and NTLM authentication (now known as integrated Windows authentication).
- **Server-Gated Cryptography** Allows financial institutions with export versions of IIS to use strong 128-bit encryption. Server-Gated Cryptography (SGC) is an extension of Secure Sockets Layer (SSL). Although SGC is built into IIS 5.0, a special SGC certificate is required.
- **New Security Wizards** Simplify server administration tasks.
  - **Web Server Certificate Wizard** Simplifies certificate administration tasks in IIS 5.0. These tasks include, for example, creating certificate requests and managing the certificate life cycle.
  - **Permissions Wizard** Simplifies editing and configuring Web site access, such as assigning access policies to virtual directories and files. The Permissions Wizard can also reflect these Web access policies to NTFS file system permissions.
  - **CTL Wizard** Configures certificate trust lists (CTLs). A CTL is a list of trusted certification authorities for a particular directory. CTLs are especially useful for ISPs who have several Web sites on their server and who need a different list of approved certification authorities for each site.
- **Kerberos v5 Authentication** Passes authentication credentials among networked computers that are running Microsoft® Windows®. IIS 5.0 is fully integrated with the Kerberos v5 authentication model implemented in Windows 2000 Server.
- **Certificate Storage** Stores, backs up, and configures server certificates through a single point of entry. IIS certificate storage is now integrated with Microsoft CryptoAPI (CAPI) storage, which is provided with Windows 2000.
- **Fortezza** Supports Fortezza, the U.S. government security standard (http://www.armadillo.huntsville.al.us/). This standard satisfies the Defense Messaging System security architecture, by supplying a cryptographic mechanism that features message confidentiality, integrity, authentication, and access control to messages, components, and systems.

Administration

This section introduces new administrative features to help you manage an IIS 5.0 installation more effectively. For details and examples, see "Administering an ISP Installation" in this book.

- **Restarting IIS** Restarts your Internet services without requiring you to restart your computer.
- **Process Accounting** Reports how Web sites use CPU resources on the server. This information helps determine which sites are using disproportionately high CPU resources and which sites have malfunctioning scripts or Common Gateway Interface (CGI) processes.
- **Process Throttling** Limits the percentage of time the CPU spends processing out-of-process scripts in Active Server Pages (ASP), Internet Server Application Programming Interface (ISAPI), and CGI applications for individual Web sites. In addition, this feature can stop and restart malfunctioning processes.
- **Improved Custom Error Messages** Allow you to send informative messages to clients when HTTP errors occur on their Web sites. This feature does detailed error processing of scripts in ASP pages through the 500–100.asp custom error message. You can use the custom errors that IIS 5.0 provides, or create your own.
- **Web-based Administration Tools** Allow remote management of your server from almost any browser on any platform. With IIS 5.0, you can set up administration accounts (called *Operators*) with limited administration privileges on Web sites, in order to help distribute administrative tasks.
- **Terminal Services** Lets you remotely administer Windows services (such as IIS) through Microsoft® Management Console (MMC). A feature of Windows 2000 Server, Terminal Services can connect to Windows servers by dialing-up or through Point-to-Point Tunneling Protocol (PPTP). Terminal Services client software must also be installed on client computers.

Programmability

This section acquaints you with the new features that improve IIS programmability. For more information, see "Administering an ISP Installation" in this book, and the IIS 5.0 online product documentation.

- **Application Protection** IIS 5.0 offers greater protection and increased reliability for your Web applications. By default, IIS 5.0 will run all of your applications in a common (or pooled) process that is separate from core IIS 5.0 processes. In addition, you can still isolate mission-critical applications that should be run outside of both core IIS 5.0 and pooled processes.

- **New Features in ASP** Enhance performance and streamline your server-side scripts:
  - New flow control methods
  - Improved error handling
  - Scriptless ASP
  - Performance-enhanced objects
  - Extensible Markup Language (XML) integration
  - Microsoft® Windows® Script Components
  - A new way to determine browser capabilities
  - ASP self-tuning
  - Server-side include with SRC attribute
  - Encoded scripts in ASP pages
- **ADSI 2.0** Adds custom objects, properties, and methods to the existing Microsoft® Active Directory Service Interfaces™ (ADSI) provider, giving you even more flexibility in configuring your sites.

For detailed descriptions of the new features, see the "Features" topic in the IIS 5.0 online product documentation.

Internet Standards

This section discusses the new Internet standards that are shipping with IIS 5.0. For more information, see the IIS 5.0 online product documentation.

- **Web Distributed Authoring and Versioning (WebDAV)** Lets remote authors edit, move, or delete files, file properties, directories, and directory properties on your server over an HTTP connection.
- **FTP Restart** Resumes downloading a file through File Transfer Protocol (FTP) at the point where the data transfer was interrupted.
- **HTTP Compression** Transmits pages faster between the Web server and compression-enabled clients. Compresses and caches static files, and compresses dynamically generated files on demand.

IIS 5.0 Architecture

The next sections describe how the components of IIS 5.0 work together. They begin with an overview of IIS 5.0 architecture as a whole, followed by discussions of its administrative and programmability architectures. They end with a discussion of how other services of Windows 2000 Server and other Microsoft products can be integrated into an installation to enhance IIS 5.0.

Architecture Overview

IIS 5.0 is a service of Windows 2000 Server, which means that it is designed to work closely with many other services that run on Windows 2000 Server. Figure 1.1 shows the relationship between IIS 5.0 and other services you can install on Windows 2000 Server.

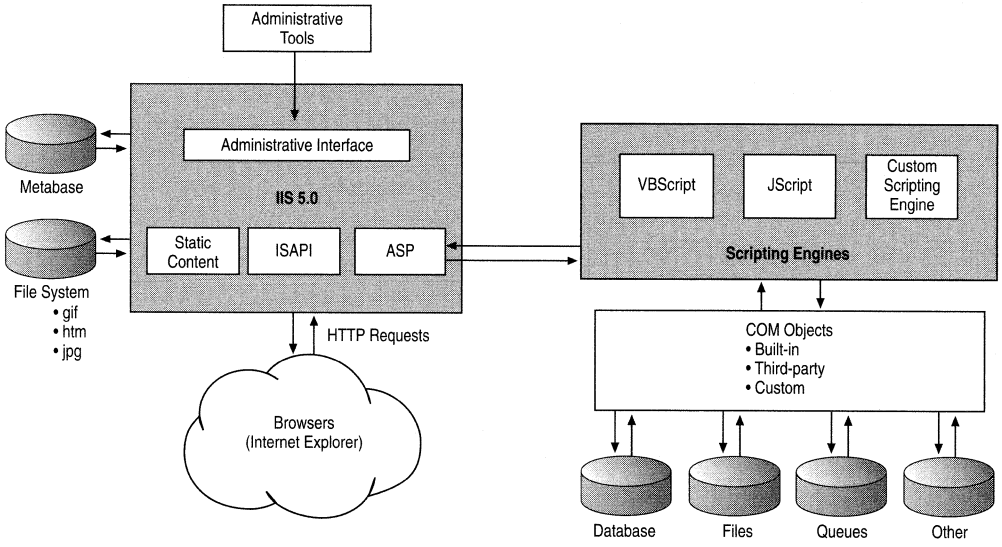


Figure 1.1 IIS 5.0 Architecture

The standard Internet services (Web and FTP servers) reside in a process called Inetinfo. In addition to these Internet services, this process contains the shared thread pool, cache, and logging services of IIS 5.0.

By sharing the same security model (user accounts) as Windows 2000 Server, IIS 5.0 eliminates the need for additional user-account administration. IIS 5.0 administration also borrows existing Windows 2000 Server tools such as System Monitor, Event Viewer, and MMC to conduct similar administrative procedures.

The following sections describe the Windows 2000 tools that are tightly integrated with IIS 5.0.

Microsoft Management Console

MMC hosts programs, called *snap-ins*, from which you can manage your servers. MMC offers a common framework in which various snap-ins can run, so that you can manage several services with a single interface. In addition to integrating and standardizing administrative tools, MMC also lets you customize the console. By picking and choosing specific snap-ins, you can create management consoles that include only the administrative tools you need.

For example, the IIS 5.0 MMC hosts Internet Services Manager as a snap-in from which you can administer Web sites on the computer where IIS 5.0 is installed. As another example, Microsoft® Component Services includes an MMC snap-in from which you can administer its transaction packages.

Active Server Pages and Microsoft Script Debugger

ASP is a server-side application environment that allows you to create dynamic Web sites and powerful Web applications. ASP pages can contain HTML tags, text, and script commands. The script commands execute on the server and return HTML pages to the requesting browser. ASP pages can call Microsoft® Component Object Model (COM) components to perform tasks, such as connecting to a database or performing a business calculation. With ASP, you can add interactive content to your Web pages or build entire Web applications that use HTML pages as the user interface.

Microsoft® Script Debugger is designed to help you quickly locate bugs and interactively test your server-side scripts in ASP pages. Script Debugger, which works with Microsoft® Internet Explorer version 3.0 or later, includes just-in-time (JIT) debugging. When a run-time error interrupts execution of your script, the Script Debugger automatically starts, displays the .asp file with a statement pointer indicating the line that caused the error, and generates an error message. With this type of debugging, your computer suspends further execution of the program. You must correct the error with an editing program and save your changes before you can resume running the script. Figure 1.2 shows a code sample, in which Script Debugger has highlighted a line containing an error.

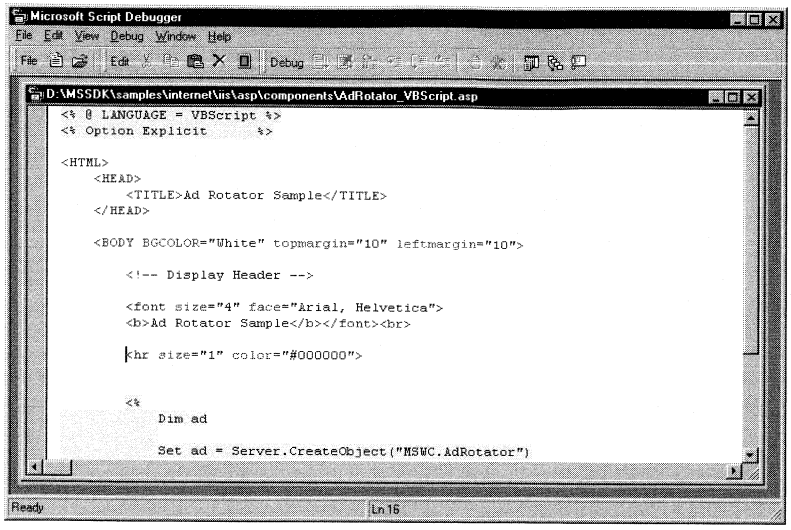


Figure 1.2 Microsoft Script Debugger

Indexing Service

Indexing Service indexes files and file properties on your disks and stores the information in a catalog. It searches the complete text and properties of documents stored on an Internet or intranet Web site. Using samples installed with Indexing Service, a Web site administrator can quickly create Web query forms and make them available to users. Users can then generate queries by filling in the fields of the query form. The Web server forwards this query form to the query engine, which finds the pertinent documents and returns the results, in the form of a Web page, to the client.

In addition to indexing Web pages in HTML format, Indexing Service can search documents formatted by applications such as Microsoft® Word and Microsoft® Excel, so that you don't have to convert them to HTML.

Certificate Services

Microsoft® Certificate Services is a general-purpose, customizable server application that issues, revokes, and renews digital certificates. These certificates, which are generated in standard X.509 version 3 format, are used for public-key cryptography applications such as server and client authentication under the SSL or Private Communication Technology (PCT) protocols. With Certificate Services, organizations can perform authentication on a corporate intranet or across the Internet.

Message Queuing

Message Queuing makes it possible for application programs to send and receive messages quickly, reliably, and asynchronously. It features Microsoft® ActiveX® support, security controls, administration tools, and integration with other strategic Microsoft features such as IIS 5.0, Component Services, and Certificate Services. As a result, Message Queuing is the queuing product of choice for applications running on Microsoft® Windows® 95 and Windows® 98, as well as on Windows 2000 Server. Message Queuing is also interoperable with other important platforms and products through the Message Queuing connector.

Database Access Component

The Microsoft® Database Access component uses Microsoft® ActiveX® Data Objects (ADO) to access information stored in a database or other tabular data structure. Data-driven client/server applications deployed over the Web or an intranet can use this component to integrate information from a variety of sources, including both relational (SQL) and non-relational database management systems (DBMSs). The Database Access component consists of the following, all of which are released, documented, and supported together:

- ADO and Microsoft® Remote Data Service (RDS)
- Open Database Connectivity (ODBC)
- The Microsoft® OLE DB provider for ODBC

ADO can help you write applications that let you access and manipulate data in a database server through an OLE DB provider. The primary benefits of ADO are ease of use, high speed, low-memory overhead, and a small disk footprint. ADO supports key features for building client/server and Web applications.

ADO also features RDS, a high-performance client-side data caching technology that brings database connectivity to Web applications. You can use RDS to build intelligent Web applications that let you access and update data from any OLE DB provider, including ODBC-compliant DBMSs. Because you can implement RDS with familiar technology—off-the-shelf visual controls, HTML, and Microsoft® Visual Basic® Scripting Edition (VBScript)—RDS integrates seamlessly with existing Microsoft® Visual Basic® applications, so that you can transport them to the Web.

The Database Access component also includes ODBC and the OLE DB provider for ODBC. Used in conjunction with an appropriate ODBC driver, these components provide access to several popular DBMSs, including Microsoft® SQL Server™, Oracle databases, Microsoft® Access, and several other desktop databases.

Administrative Architecture

This section describes how IIS 5.0 is built to simplify administrative tasks and allow you the flexibility of administering a site in several ways. To do this, IIS 5.0 offers a comprehensive set of tools, which allow you to manage a Web server and its components, as well as an independent Web site. In addition to these tools, customers can create their own custom interfaces using IIS Administration Objects (described in the next section), which ship with IIS 5.0.

Figure 1.3 shows the administrative tools provided with IIS 5.0 and how they interact with IIS Administration Objects.

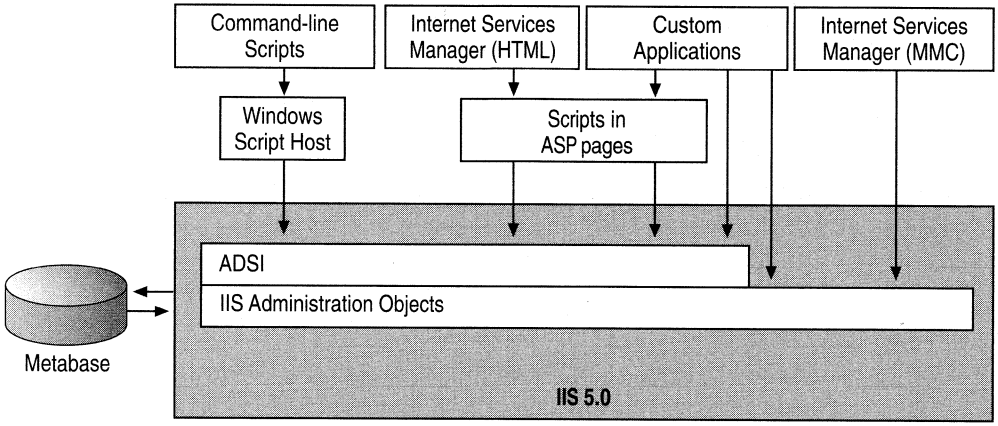


Figure 1.3 Administrative Architecture of IIS 5.0

IIS Administration Objects

IIS Administration Objects (IIS Admin Objects) are fully distributable COM objects with methods that let an application manipulate IIS 5.0 configuration keys and data in the memory-resident metabase. You can use IIS Admin Objects to write applications, such as server administration or Web authoring tools, that check and update the server's configuration by manipulating keys and data in the metabase. You can also use IIS Admin Objects to store your IIS 5.0–related custom application configuration data in the metabase (for faster access) without filling up the Windows system registry.

IIS Admin Objects are also programmable COM objects that a script in an ASP page or custom application can call to *change* IIS 5.0 configuration values stored in the IIS 5.0 metabase. For example, file and directory access permissions used by IIS 5.0 are stored in the metabase. You can efficiently set these permissions for one or many files and directories with a simple script in an ASP page. The Internet Services Manager snap-in to MMC and the browser-based version of this snap-in (discussed in the next section), the Microsoft® Windows® Script Host (WSH), and custom administration applications all use IIS Admin Objects to manage IIS 5.0.

Internet Services Manager

With Internet Services Manager, a snap-in for MMC, you can manage many IIS 5.0 Web sites from a single location anywhere on the Internet. With this snap-in, you can create Web sites and virtual directories. You can also set levels of permission, fine-tune logging, and enable throttling.

IIS 5.0 also includes a browser-based version of the IIS 5.0 snap-in, Internet Services Manager (HTML), shown in Figure 1.4. With this browser-based tool, you can configure IIS 5.0 from almost any computer on the Internet or on a private intranet.

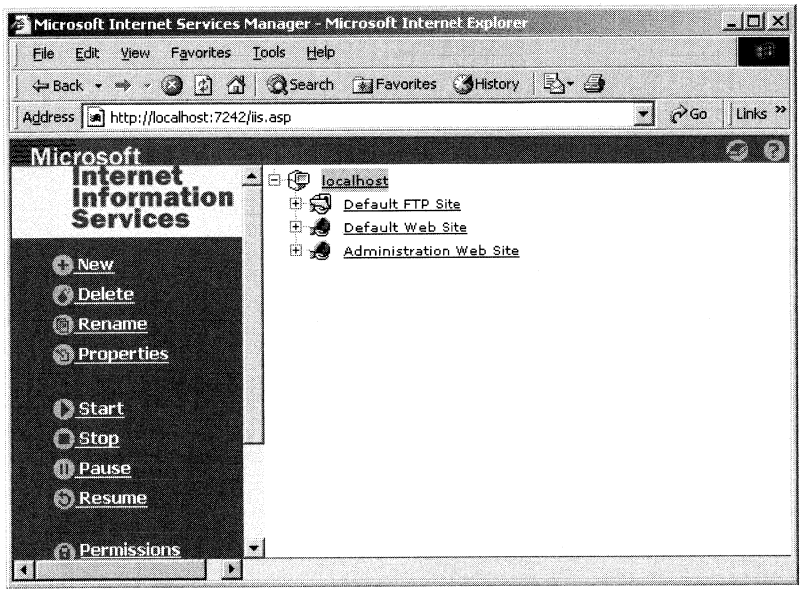


Figure 1.4 Internet Services Manager (HTML)

Built-in and Customized Scripts

IIS Admin Objects make programmatic administration as straightforward as possible. They are based on ADSI, and together they let you automate many administrative tasks. IIS 5.0 comes with several built-in administration scripts installed by default in the Inetpub\AdminScripts directory. But you can also create your own customized scripts to handle more complex tasks. ADSI can be easily accessed and manipulated by any language that supports automation, such as VBScript or Microsoft® JScript®, Visual Basic®, Java, or C++. For examples and information about creating your own custom scripts, see "Administering an ISP Installation" in this book.

IIS Admin Objects and ADSI work in conjunction with WSH, a language-independent scripting environment for 32-bit Windows platforms. Microsoft supplies both VBScript and JScript scripting engines with WSH. For other languages such as Perl, you can buy ActiveX scripting engines through third-party companies.

Programmability Architecture

Programmability architecture is based on a hierarchy of how IIS 5.0 processes incoming requests. Once you understand how the hierarchy works, you can design efficient applications to capitalize on the features IIS 5.0 has to offer.

Web applications are maturing into mission-critical, line-of-business applications that demand reliability and availability for all customers. Before the Web, most applications were written and executed on stand-alone computers as single-user applications, and most shared server code was written and executed within databases. Web applications are deployed in a distributed, disconnected environment. They often run on many different servers and access information from many different data stores. IIS 5.0 adds the necessary technologies to the Windows 2000 Server platform, so that organizations can develop and deploy reliable and scalable Web applications for multiple users.

Figure 1.5 illustrates the programmability architecture of IIS 5.0 and the components described in this section. The figure starts with the most difficult way to develop Web applications (with CGI on the left) and moves to the easiest (on the right), which takes advantage of all the features in the IIS 5.0 hierarchy.

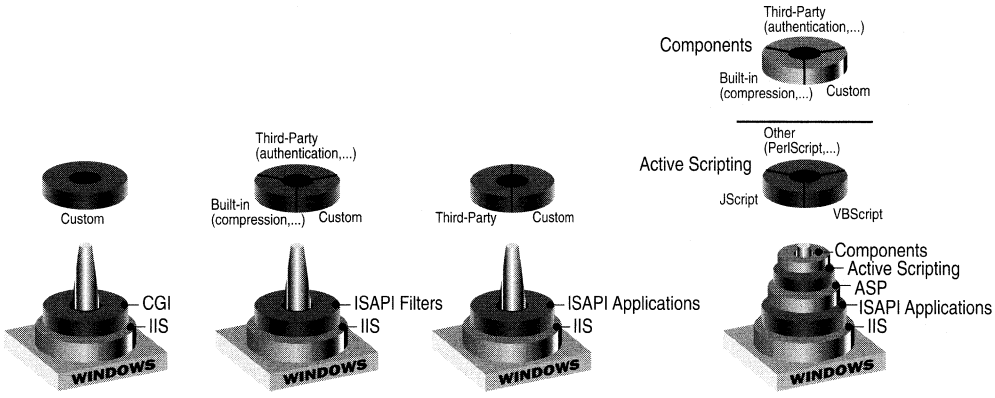


Figure 1.5 Programmability Architecture

The first three hierarchies (from left to right) shown in Figure 1.5 are self-limiting. When setting up applications to run on a Web server, it is recommended that you choose the hierarchy farthest to the right, and use built-in applications whenever possible.

Programmability is configured around three options for Web applications. Each hierarchy in Figure 1.5 offers one or more of these options. You should choose them in the following order (depicted from right to left):

1. **Built-in** First, try to find applications that come with IIS 5.0. For example, if you need to add compression to a site, install the built-in ISAPI filter or COM component.
2. **Third-Party** Next, if you want an application that performs a specific task, but can't find one that comes with IIS 5.0, buy from a third-party vendor. For example, buy an ISAPI filter for authentication.
3. **Custom** Finally, if you need a type of application not already built into IIS 5.0 or already created by a third party, you need to customize your own. This method (on the left) is obviously the most difficult and should be your last resort. In this category the least recommended method is to build CGI applications, for reasons explained later in this chapter.

Developing Web applications involves many of the same complexities as developing multiuser server applications. For instance, when creating a multiuser application, developers must invest time building complex routines for managing server process pools, thread pools, database connections, user context, and transactions usually associated with server applications. IIS 5.0 as well as features and products made to run with it on the Windows operating system eliminate much of this complexity, thanks to built-in server technologies. Along with Windows 2000 Server, these technologies give developers a platform for developing Web applications.

For more information about developing Web applications, see "Developing Web Applications" and "Data Access and Transactions" in this book.

Common Gateway Interface

IIS 5.0 fully supports both scripts and executable programs written for the CGI specification. When you or your customers create executable CGI programs on a Web server, remote users can start these programs by filling out an HTML form or by simply requesting a URL from the server. Arguments following the question mark in the URL are passed to the CGI application as an environment string, which is then parsed and acted upon. CGI applications run out of process on the server, which means each request creates its own process. This architecture makes CGI applications slower than other types, but because they run in their own process, any problems are less likely to affect the operation of the server.

ISAPI Filters

ISAPI is an Internet API for extending IIS 5.0 and other HTTP servers that support its interface. ISAPI filters are dynamic-link libraries (DLLs) that allow preprocessing of requests and postprocessing of responses, which in turn allows for site-specific handling of HTTP requests and responses. These filters can also be run out of process, in order to improve reliability and to consume fewer resources than CGI applications.

IIS 5.0 contains several built-in ISAPI filters, such as one for compression. You can also buy and install third-party filters, such as an authentication filter, to do tasks not covered by the built-in filters. For information about installing filters, see the "Installing ISAPI Filters" topic in the IIS 5.0 online product documentation.

If you don't find a built-in filter or one that you can buy, you can build a custom filter. For example, you can write an ISAPI DLL to intercept specific server events and perform appropriate actions. This functionality is especially useful in implementing server security. For more information, see the "ISAPI Filters Overview" topic in the IIS 5.0 section of the SDK documentation on MSDN.

You will find several sample ISAPI filters on the *Microsoft® Windows® 2000 Server Resource Kit* companion CD.

ISAPI Extensions

ISAPI extensions are multithreaded DLLs that can be loaded into the same memory space (in-process) occupied by the Web service, and can perform server-side tasks as an interface between the user and IIS 5.0. ISAPI extensions have a strong performance advantage over CGI applications for several reasons:

- ISAPI extensions can be configured so all of them run in a single process, or so they run in the same memory space as the Web service.
- Instead of loading an executable for each request, ISAPI uses thread-safe DLLs that are loaded only once.
- ISAPI uses Microsoft® Win32®-based APIs to communicate with the Web service, which are much faster than CGI methods.

IIS 5.0 does not have any built-in ISAPI extensions, so you either have to buy third-party extensions or customize your own. ISAPI extensions developed for other Web servers are generally easy to port to IIS 5.0. In addition, it is sometimes advantageous to rewrite existing CGI applications as ISAPI extensions to improve their performance. For more information, see "Migrating a Web Server to IIS 5.0" in this book. For information about designing ISAPI extensions, see the "Designing High-Performance ISAPI Applications" topic in the IIS 5.0 online product documentation.

Active Server Pages

ASP offers an open, server-application environment in which you can combine HTML, server-side scripts, and reusable COM server components to create dynamic and powerful Web-based business solutions. After the server-side script on an ASP page runs, the results are returned to the client browser in the form of a standard HTML document.

IIS 5.0 natively supports scripts in ASP pages written in both VBScript and JScript. However, you can write ASP applications in any scripting language, as long as you install a scripting engine that conforms to the Active Scripting standard discussed later in this chapter. Scripts in ASP pages have access to objects that make development quick and easy, including **Application**, **Session**, **Request**, **Response**, and **Server** objects.

ASP also supports COM components, which allow you to reuse business logic in other applications. ASP is supported by a number of Web servers, and existing ASP applications can be easily ported to IIS 5.0. In addition, it is often a good idea to rewrite CGI applications as ASP pages, particularly those with functionality that you can quickly and easily reproduce by using ASP built-in objects. For more information, see "Migrating a Web Server to IIS 5.0" in this book.

COM Components

ActiveX is a technology built on COM that allows the developer to create objects or controls that "activate" content on the Web. With tools such as Microsoft® Visual C++®, Visual Basic, or Microsoft® Visual J++®, you can develop COM components and embed them in a Web page, adding a higher level of interactivity to the page. COM components can be run on a server, on a client, or on both. The IIS 5.0 online product documentation, for example, uses Microsoft® ActiveX® control called HTML Help, located in the left-hand frame of the browser, for its table of contents. When interfacing with OLE DB, ADO, or other database-access methods to retrieve information stored in an Access or SQL Server database, you can write COM components in any COM-compliant language, such as Visual Basic, Visual C++, or Visual J++.

Active Scripting

Active Scripting is a technology created by Microsoft to help developers take advantage of existing scripting languages and to benefit from their use through a COM interface. Microsoft ships two languages that take advantage of Active Scripting—VBScript and JScript. However, the scripting interface is open, allowing third parties to supply their own languages—like Perl—to any application that implements scripting.

Component Services

Component Services is a transaction processing system for developing, deploying, and managing distributed server applications. Simply put, a transaction is any business operation in your daily life, such as exchanging money for goods or services.

In software, a transaction is an operation (initiated by an application) that succeeds or fails as a whole, even if the operation involves many steps (for example, ordering, checking inventory, and billing). Transaction processing is crucial for distributed business applications that require accuracy, data consistency, and security.

The following list shows examples of some of the built-in and installable Component Services applications:

- Transaction Management
- Queued Components
- Object Pooling
- Just-In-Time Activation
- In-Memory Database
- Thread Management
- Role-Based Security
- Ad Rotator

A transaction changes a set of data from one state to another. For example, if you withdraw money from your bank account, your balance changes to reflect the transaction. For a transaction to work correctly, it must have what are known as the ACID (Atomicity, Consistency, Isolation, and Durability) properties:

- **Atomicity** All changes from objects involved in the transaction are committed as one unit. All changes are either committed or rolled back to their original state.
- **Consistency** Data must change to reflect the transaction, like a bank balance changing to reflect a withdrawal.
- **Isolation** Concurrent transactions are unaware of each other's partial and uncommitted results. Work that is completed by concurrent transactions can be thought of as occurring in a serial manner, one after the other. Otherwise, the transactions might create inconsistencies.
- **Durability** Committed updates to managed resources (such as database records) will survive communication, process, and server system failures. Transactional logging lets you recover the original state after failures.

All of these properties ensure that a transaction does not create problematic changes to data between the time the transaction begins and the time it must commit. Also, these properties simplify cleanup and error handling when updating databases and other resources.

With Component Services you can work with transactions effectively, and even package components within transactions. You can develop a transactional application for a single user, and then use simple scripting commands to scale it for use in a production environment. Component Services components are activated when needed and deactivated when not, thereby conserving server resources and increasing the number of users who can run your application concurrently. Component Services applications can also be run in separate memory spaces so that their operational status will not affect other applications, a function called *process isolation*.

Component Services is much more than a transaction-management server. It is also an object manager for distributed network objects and environments. In fact, it defines a programming model, and provides a run-time environment and a graphical administration tool for managing enterprise applications. Specifically, it can do the following:

- Process distributed transactions
- Automatically manage processes and threads
- Manage object instances
- Control object creation and use through a distributed security service
- Handle system administration and manage components through a graphical interface

Figure 1.6 shows the graphical interface of the Component Services snap-in for MMC and displays installed packages. Through this interface, you can add or delete packages or configure Component Services as needed.

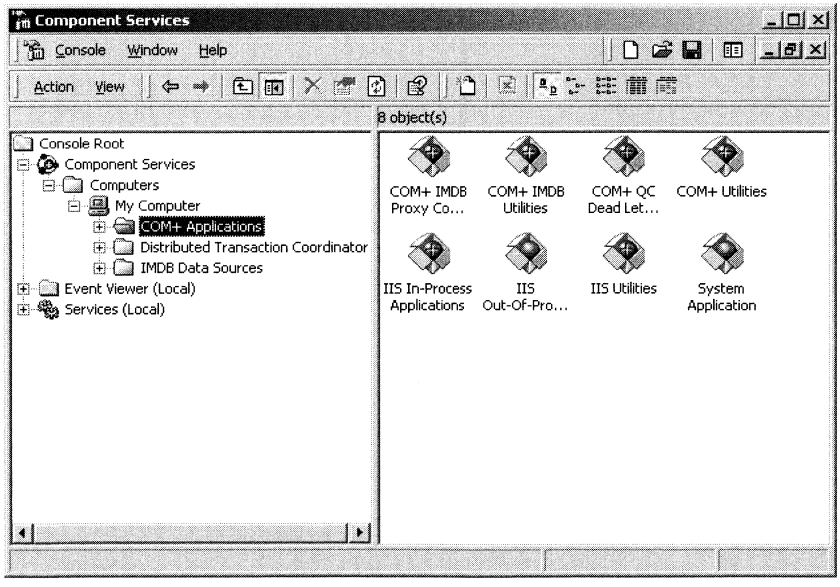


Figure 1.6 Component Services Snap-in for MMC

Publishing on Web Sites

This section introduces the general publishing solution offered by Windows 2000 Server through legacy data and the following technologies:

- WebDAV, a new extension to HTTP/1.1
- Microsoft® FrontPage® Server Extensions
- FTP

WebDAV

WebDAV is an extension of the HTTP 1.1 standard for exposing a hierarchical file storage media, such as a file system, over an HTTP connection. With WebDAV you can control how remote authors access resources on the file system by letting them edit, move, search, or delete files and directories, and their properties, on a Web server. You can configure a WebDAV virtual directory's permissions to:

- Search for directories, files, and their properties.
- Create, modify, and delete directories, files, and their properties.
- Create, modify, delete, and browse directories and their properties.
- Store and retrieve custom properties for files and directories.
- Lock files so that multiple users can read a file concurrently, but only one individual can modify it.

For details about setting WebDAV permissions, see the "Setting Web Server Permissions" topic in the IIS 5.0 online product documentation.

The advantage of using WebDAV is its interoperability. For example, a client can connect to a WebDAV server from a Macintosh or UNIX client computer. Uniform Naming Convention (UNC) connections, on the other hand, do not offer this flexibility.

FrontPage Server Extensions

FrontPage Server Extensions are a set of programs that you can install with the Windows 2000 operating system on a Web server. You can add them on to the IIS 5.0 snap-in and thus simplify administration. FrontPage Server Extensions can do the following in terms of *FrontPage webs* (projects containing all the pages, images, and other files that make up a Web site):

- Author FrontPage webs

  Example: When an author moves a page from one folder to another in a FrontPage web, the Server Extensions automatically update all hyperlinks to that page from every other page and from every Microsoft® Office document in the FrontPage web. These updated hyperlinks are then placed directly on the Web server.

  For a full description of FrontPage webs, see the *Microsoft® FrontPage® 2000 Server Extensions Resource Kit* at http://officeupdate.microsoft.com/frontpage/wpp/serk.

- Administer FrontPage webs

  Example: A FrontPage web administrator can specify which users can administer, author, or browse a FrontPage web.

- Maintain browse-time functionality

  Example: When users of a FrontPage web participate in a discussion group, the Server Extensions maintain an index of hyperlinks to articles in the discussion, separate discussion threads, tables of contents, and search forms to locate pages of interest.

The FrontPage client and Server Extensions work together to minimize costly file transfers over the Internet, a definite advantage in terms of publishing on a Web site. When the Microsoft® FrontPage® Explorer opens a FrontPage web from a Web server containing the Server Extensions, information about the FrontPage web (such as its hyperlink map) is downloaded to the client machine so that the FrontPage Explorer can display the information. However, the full set of pages and other files that make up the FrontPage web remain on the Web server. A page is only downloaded over the Internet when it is opened for editing in the Microsoft® FrontPage® Editor. This is a very efficient mechanism: An entire Web site can be changed directly on a Web server at the cost of downloading and editing a single file.

FTP

Finally, you can publish to Web servers with FTP. Integrated into the Windows operating system as an Internet service, FTP publishes information on a Web server through a standard FTP client. Depending on the client, you can operate FTP through the command-line or through a graphical user interface (GUI)–based interface.

For more information about FTP, see "Administering an ISP Installation" in this book.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server.

http://www.microsoft.com/intranet/
The intranet area of Microsoft® TechNet. You can download white papers, FAQs, case studies, and free intranet solutions written by Microsoft® Solution Providers.

http://msdn.microsoft.com/workshop/
Microsoft's MSDN Online Web Workshop home page is a useful resource for Webmasters and Web application developers.

http://msdn.microsoft.com/workshop/server/
The Server Technologies area of the MSDN Online Web Workshop, which includes links to information about ASP.

http://www.learnASP.com
A good resource for ASP. The site contains ASP-related articles, FAQs, tutorials, tools, and free ASP component downloads.

http://mspress.microsoft.com/
The Microsoft Press® Web site. Microsoft Press publishes a number of books and training materials about Microsoft's products and related technologies.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 2

Managing the Migration Process

Internet Information Services (IIS) 5.0 simplifies the setup and administration of Web services and provides many advantages over competing Web servers and previous versions of IIS. With IIS 5.0, Microsoft has emphasized a few key areas, notably security and manageability. New methods of publishing are implemented in this version, as well as updates and modifications to the Active Server Pages (ASP) model that help to improve application processing capacity and efficiency. These new and enhanced features give you compelling reasons to migrate or upgrade a Web server to IIS 5.0.

This is the first of two chapters about migrating to IIS 5.0. This chapter provides general, conceptual information about planning and managing the migration process when it involves multiple servers on an enterprise network. The next chapter, "Migrating a Web Server to IIS 5.0," provides specific "how-to" information about migrating an individual Web server.

This first section of this chapter gives an overview of the migration process, using the Microsoft® Solutions Framework (MSF) approach to deploying successful and cost-effective business solutions. It divides a Web server migration project into four phases: Envisioning, Planning, Developing, and Deploying. Subsequent sections provide details on key activities and milestones for each phase. The "Additional Resources" section provides references to more in-depth information about Web server migration and other useful tools. Forms and templates to use in planning a migration project are included on the *Microsoft® Windows® 2000 Server Resource Kit* companion CD.

In This Chapter

Migration Process Overview

Envisioning

Planning

Developing

Deploying

Additional Resources

Migration Process Overview

The approach to migration used in this chapter is based on the Microsoft Solutions Framework (MSF) process model for Infrastructure Deployment, developed by Microsoft to assist customers in deploying business solutions. MSF is a flexible, interrelated series of models that can guide an organization through assembling the resources, people, and techniques needed to bring technology infrastructure in line with business objectives. You can use MSF together with your own tools and techniques to plan and manage a migration project. For more information about MSF, see http://www.microsoft.com/msf/.

The following graphic depicts the MSF process model, consisting of envisioning, planning, developing, and deploying, along with related milestones. The remainder of this chapter describes the migration process within the context of this model.

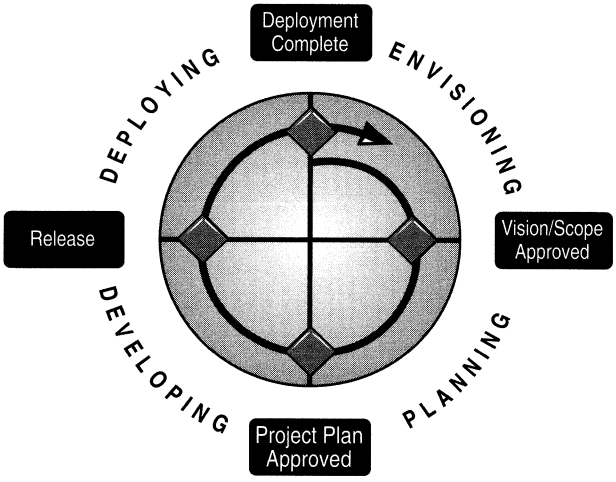


Figure 2.1 The MSF Process Model for Infrastructure Deployment

Table 2.1 describes how IIS 5.0 migration activities fit into the MSF process model, as depicted in the preceding figure. Although listed sequentially, many of the activities are performed concurrently, particularly within a phase.

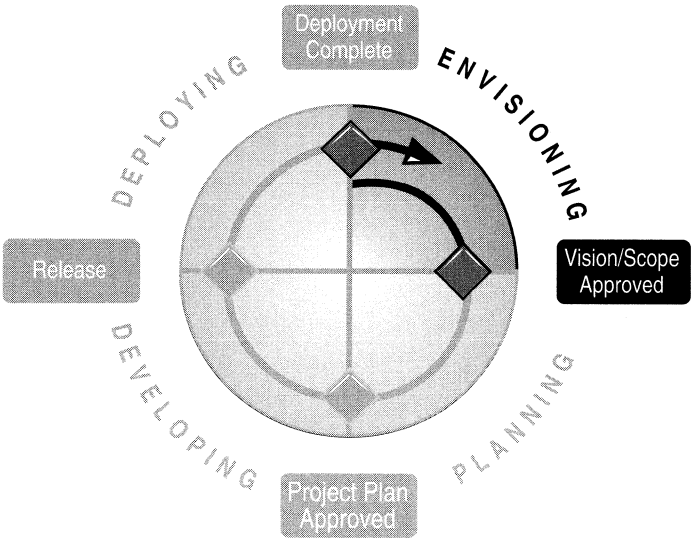
Table 2.1 The MSF Process Model and Migration Activities

MSF Phase	Milestone	Migration Activity
1. Envisioning: defining the goals and limits of the project	Vision and Scope Document Approved	A. Define the project by identifying goals, scope, constraints, and assumptions. B. Create a requirements definition that describes what the new Web services must do. C. Develop a conceptual design for services. D. Assess high-level project risks. E. Define the structure of the project team.
2. Planning: writing the Functional Specification and Project Plan	Project Plan Approved	A. Gather information about current Web services. B. Define and design the new service offering in a functional specification. C. Assess resource needs to complete the project. D. Build a master project plan. E. Draft a project schedule.
3. Developing: developing, testing, and building out the system	Release	A. Validate the physical design by simulating the server environment and performing unit, integration, and application testing. B. Build out the system, configuring and locating the production Web servers that will be used on your network. C. Begin training administrators and key users. D. Conduct pilot testing and introduce the new services to a defined set of users on a small- and medium-scale basis.
4. Deploying: making the new services available to end users	Deployment Complete	A. Finish training administrators and all users. B. Roll out the new system, evaluate performance, and correct any problems. C. Monitor the system, and plan improvements and enhancements.

Every migration project varies in size, scope, and objectives. While this chapter provides an overview of the basic tasks involved, you may require additional assistance, especially when planning and managing a large-scale migration involving multiple geographic locations and numerous servers and users. If so, you might want to consider contacting a Microsoft Solution Provider or a Microsoft Consultant. For more information about obtaining this help, see http://www.microsoft.com/msf/.

For help with planning and deploying Microsoft® Windows® 2000 Server solutions, see http://www.microsoft.com/windows/server/default.asp.

1. Envisioning



During the Envisioning phase, the project team creates a Vision and Scope document to define the following:

- Project vision, scope (given the schedule and constraints), and assumptions
- Project requirements
- Conceptual design for services
- High-level project risks
- Structure of the project team

The Envisioning phase focuses the team on creating solid business value. It keeps your organization from investing significant effort on minor needs or from improving bad processes rather than creating good ones. The best results are based on open thinking about how the proposed solution may address not only the most visible current need, but also the underlying causes of that need. It is also important to consider similar issues in other departments, as well as issues that your business might face in the future.

The main milestone for the Envisioning phase is the project team's approval of the Vision and Scope document. Once this is completed, you'll be ready to move to the next phase, Planning.

To help you write a Vision and Scope document, you can use the "Vision and Scope Template" found in the VisScope.doc file included with IIS Tools. To install IIS Tools, run Setup from the Resource Kit companion CD and choose the **IIS Tools** option.

Define the Project

The first section of the Vision and Scope document should provide clear direction to the team by defining project vision, scope (given the schedule and constraints), and assumptions. Asking questions such as the following will help you in this effort:

- **Vision** At a high level, what final outcome or result do we envision for the project? For example, a vision statement might be, "To leverage information technology through fast and cost-effective development and deployment of crucial line-of-business applications." The following is an example of a vision statement for a Web server migration project, using a fictitious company, Contoso Pharmaceuticals:

  > Contoso Pharmaceuticals will replace its current UNIX Apache Web server environment with Windows 2000 Server and IIS 5.0, a more efficient and flexible solution that will maximize competitiveness in our industry while reducing operational and administrative costs. The company will implement a global Windows 2000 Server domain model and will begin a scheduled deployment program to 20,000 worldwide users at 100 locations by the third quarter of 1999. It will start an enterprise-wide rollout within three months and will be user-complete within 18 months, or the first quarter of 2001.
  >
  > Implementation will require a conversion and coexistence infrastructure in order to seamlessly move users to the new platform. To accomplish this, the company will use Windows 2000 Server integration tools and third-party UNIX conversion tools.

  A vision statement can also include a conceptual design drawing, as described in "Develop a Conceptual Design" later in this chapter.

- **Scope** What functionality can we reasonably implement during this project, and what should be postponed to a later date? Project variables—resources (people and money), schedule (time), and features (the solution)—exist in a triangular relationship, as depicted in Figure 2.2. Setting project scope requires balancing these variables. This could be accomplished by trading desired, but inessential, features for a shortened project schedule or reduced resource requirements.

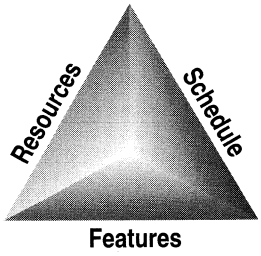


Figure 2.2 Balancing Project Variables

- **Assumptions** What assumptions are implicit in the project plan? Some assumptions might include that a valid Windows 2000 Server domain design has been implemented, or that qualified staff is available.

Create a Requirements Definition

It's important to clearly define the requirements for the new Web service in your Vision and Scope document. Your primary reason for migrating to IIS 5.0 could be as simple as adding Web services to your corporate network, or as far-reaching as automating workflow processes or providing online transaction processing (OLTP). You might also have a number of secondary goals. Although you might not have the resources to realize every goal at this time, it is worthwhile to list the more important ones to include in longer-range planning. Table 2.2 lists some typical requirements for a migration project.

Table 2.2 Project Requirements and Deliverables

Requirement	Deliverable
Improve information sharing	Integrate the corporate intranet with Microsoft tools and technologies. IIS 5.0 fully integrates with other Microsoft business products, such as Microsoft® Word, Microsoft® Excel, and Microsoft® SQL Server™. In addition, IIS 5.0 is fully compliant with Microsoft® ODBC and Microsoft® OLE DB, and can connect to more than 55 different databases. Use this built-in functionality to access, link, or exchange data with these products, without the need for additional software.
Provide advanced online business services	Use IIS 5.0 in combination with other Microsoft products, such as Microsoft® Site Server, to conduct transaction-based services, central management, content indexing, and many other advanced services. You're ensured all system components have been designed, tested, and optimized to interoperate. For large Commercial Solution Providers, Microsoft® Commercial Internet System (MCIS) provides a fully integrated solution.
Increase security	Use IIS 5.0 security, which integrates Windows 2000 Server security policies and user databases, to reduce the number of security holes created by Web servers running on top of nonintegrated or incompatible security systems.
Reduce system support costs	Reduce support costs and risks associated with use of noncommercial software. IIS 5.0 provides enhanced local and remote Web administration at a significantly lower cost than comparable UNIX solutions. Also, take advantage of the lower average cost of hardware for Microsoft® Windows®-based systems.
Reduce application development costs	Use ASP technology, with its many features that make application development quick and easy, including IIS 5.0 built-in script debugging.
Improve application and server performance	Take advantage of the efficient performance of ASP-based applications and Internet Server Application Programming Interface (ISAPI) extensions. You'll notice improved server performance because IIS 5.0 can run ASP-based applications and ISAPI extensions. This allows for more efficient use of resources than is possible on most other Web servers.

Develop a Conceptual Design

From the requirements definition, you can develop a conceptual design that reflects your goals and requirements and that takes into account the current network and server environment. This is usually a high-level drawing, included in your Vision and Scope document, that depicts an optimal solution.

Figure 2.3 is an example of a conceptual design.

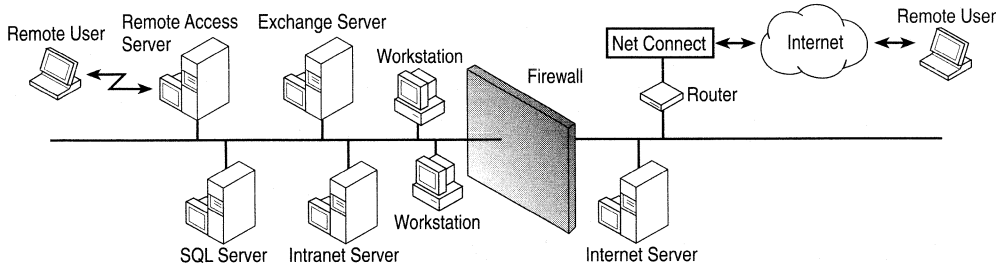


Figure 2.3 Example of Conceptual Design

This figure shows a corporate system that includes an intranet and a corporate Internet Web site. It supports internal and Internet e-mail through Microsoft® Exchange Server. A firewall protects sensitive company information and the company intranet against outside intrusion.

Assess Risk

Migrating a Web server can involve some risk. Analyzing risks before you begin a project, and implementing methods to mitigate them, can reduce any undesirable impact on your business. Table 2.3 lists some high-level project risks and mitigation strategies. You should include this type of assessment in your Vision and Scope document. Your list will probably include some risks that are more specific to the parameters of your project.

Table 2.3 Project Risks and Mitigation Strategies

Risk	Mitigation Strategy
Service downtime	Upgrade or migrate to a different physical server, make sure servers pass all tests prior to full deployment, and test and debug all applications prior to full deployment.
Applications not functioning correctly after migration	Make sure skilled resources are available to perform migration work, especially for mission-critical applications. Also, test and debug all applications on a computer that is running Windows 2000 Server and IIS 5.0. Prior to deployment, perform stress testing that simulates the actual usage of the applications, in order to uncover any performance problems.
Project not completed on schedule	During the planning process, be generous with estimates of time necessary for the expected work, and then allocate backup resources in the event that a task takes longer than anticipated. A 20 to 30 percent buffer is recommended.
Expenditures exceed budget	Estimate costs liberally, and allow for unexpected expenses; for example, hardware might malfunction and need to be replaced or software might need to be upgraded. Adding 20 to 30 percent to your base cost estimate will give you a reasonable cushion.
Applications or tools don't run on Windows 2000 Server or IIS 5.0	Make a careful inventory of applications and tools currently in use and, prior to system implementation, replace any that are incompatible. For more information about this inventory, see "Gather Information" later in this chapter.
Resistance to change by system users	Try to give individuals who will be using the new system a sense of ownership by involving them in its development. For example, provide information about the project in advance, and request input during the planning process. Give regular progress reports, and make sure users receive adequate training and support on the new technologies.

To evaluate risks for your project, you can use the "Risk Assessment Form" found in the RiskAsmt.doc file on the Resource Kit companion CD.

Define the Project Team

In the MSF model, a team of peers plans and implements a project. Each member is responsible for a functional area of the project, but overall project responsibility rests with the team as a whole. The project team definition of the Vision and Scope document identifies functional areas to be addressed by team members and assigns individual roles and responsibilities. For a small project, a single team member can be responsible for one or more functional areas. On larger projects, a separate team might be assigned to each functional area. Table 2.4 provides a description of the roles and responsibilities typically necessary in a migration project.

Table 2.4 Migration Project Roles and Responsibilities

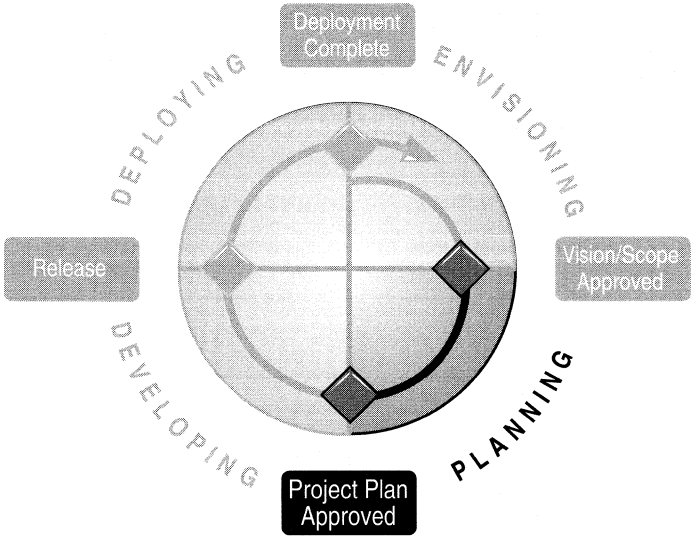
Role	Responsibility
Product Management	Product Management articulates a vision for the service and compiles requirements. This person or team is part of the project team and represents (or defends) end-user interests throughout the project. This role is not necessarily technical.
Program Management	Program Management drives the critical decisions necessary to release the right service at the right time, and coordinates the decision-making process in order to deliver the service in a manner consistent with organizational standards and interoperability goals. This person or team takes a technical role in the products and also coordinates the day-to-day activities of the rollout, providing technical guidance to the team and reporting progress to the Executive Sponsor, a high-level manager who might not necessarily be part of the team. The individual or team comprising Program Management is typically involved full-time and should have project management experience.
Development	Development builds or implements a system that is fully compliant with the Functional Specification, as described in "Define the New Service Offering" later in this chapter. This person or team has several responsibilities in a Web server migration project: • Developing and designing the system services and base configuration of the system • Creating profiles, system policies, and the overall system user interface • Designing, testing, implementing, and supporting the system • Selecting, evaluating, migrating, implementing, and supporting Web applications

Continued

Table 2.4 Migration Project Roles and Responsibilities *(continued)*

Role	Responsibility
Test	Test exercises the user interface, applications, and integration of new software into existing systems, ensuring that all issues are known before the release of the service. The test person or team is responsible for developing procedures and guidelines for testing and evaluating all applications in conjunction with new hardware and software systems. This role is responsible for writing the test suites and ensuring project goals have been met.
User Education	User Education improves the user experience through training and support systems. This person or team is responsible for ensuring that the user education process and documents are completed, including all documentation relative to this installation. This role also creates a knowledge base for support and evaluates the various options, before selecting the best ones for training and education programs.
Logistics	Logistics ensures a smooth rollout, installation, and migration of the system to the operations and support groups. This person or role is responsible for planning the deployment of technology.
Executive Sponsor	This is a high-level management official (at the Director, Vice President, or Corporate Information Officer level) who has a great deal of authority and who can support your efforts by providing assistance throughout the project. This individual will not be involved full-time. The Executive Sponsor is not necessarily a team member, but serves as an "external influence" on the team.

2. Planning



Planning, the second phase of the MSF process model, is perhaps the most important activity in a migration project. Careful planning can make the difference between a smooth transition to IIS 5.0 and a rocky one. During this phase, the project team defines:

- **What** solution the project team will deliver. The solution is defined in the form of a functional specification document.
- **How** the solution will be developed, tested, and deployed, in the form of a master project plan.
- **When** development of the solution will begin and when its deployment will be completed, in the form of a project schedule.

As part of this process, you will gather information about current services, define the new service offering, as well as assess resource needs and time requirements for carrying out the project. These activities will yield the information you need to build a functional specification, as well as a master project plan, and a project schedule. The project team's acceptance of the project plan is the milestone signifying completion of the Planning phase.

Team Roles during the Planning Phase

During the previous phase—Envisioning—Product Management had a focal role. Now team focus shifts to Program Management, where it will remain during the Planning and Developing phases. Table 2.5 shows team roles and responsibilities during the Planning phase:

Table 2.5 Team Roles During the Planning Phase

Role	Responsibility
Product Management	Conceptual design, business and user needs analysis, budget
Program Management	The Functional Specification, project plan, and schedule
Development	Technology evaluation, physical design, development plan, and schedule
Test	Design evaluation, test plan, and schedule
User Education	User education, communications, usability test plan, and schedule
Logistics	Design evaluation, technical education plan, and schedule, as well as logistics plan and schedule

Gather Information

In developing a plan for new services, it's important to have a complete picture of the current environment for Web services, in terms of the technical resources in use, their locations, and the users that draw upon them. This will help you decide how to make the best use of resources and minimize disruption of important services.

Server and Network Environment

Drawing from a survey of your current Web servers, along with their functions, utilization level, and relationships with other servers on the network, you can determine which services to migrate to IIS 5.0 and how to integrate them in the service environment. In addition, a network environment survey can help you characterize the current network and its resources. It will tell you, for example, the names and locations of Web servers, proxies, and gateways, numbers of local and remote users, and access points. You can find sample forms to use for conducting this survey, titled "Organization Environment Survey" (OrgEnvir.doc) and "Technical Environment Survey" (TechEnvi.doc), on the Resource Kit companion CD.

You might also want to perform a traffic analysis for estimating capacity requirements. For more information about doing this, see "Capacity Planning" in this book.

Tools and Utilities in Use

You'll also need to develop a list of the tools and utilities currently in use and, with the vendor or creator, verify their compatibility with Windows 2000 Server and IIS 5.0. These tools might include administrative, programming, Web publishing, and client-side applications, such as Internet browsers.

Some of these tools might no longer be required after a migration; for example, if you use Microsoft® FrontPage® Web site creation and management tools, you can eliminate many other publishing tools, even if you're running other Web servers in addition to IIS 5.0. To use all the features of FrontPage, you also need to install the appropriate Microsoft® FrontPage® Server Extensions, which are available for most popular Web servers, including those servers running on UNIX. For more information about FrontPage Server Extensions, see http://www.microsoft.com/frontpage/.

Chances are that users will want to continue using at least some of their existing tools. If you're migrating from a UNIX-based server, this could mean obtaining the Windows 2000 Server counterparts to UNIX tools. For more information about acquiring these, see the resources listed at the end of "Migrating a Web Server to IIS 5.0" in this book.

You can use a checklist to help generate a list of the Windows versions of UNIX tools and utilities that you'll need to obtain when migrating from a UNIX-based Web server to IIS 5.0. Some are available from third-party sources and from public FTP sites. You will need to port or rewrite any custom tools. Here are some items that might be on your list:

- IIS Migration Wizard, if migrating from Netscape or Apache
- Web content migration tools, such as FrontPage
- UNIX-to-Microsoft® MS-DOS–based file copy and conversion tools, such as Mtool
- Test scripts
- Test tools, such as Web Capacity Analysis Tool (WCAT), the Web Application Stress Tool, and the IIS 5.0 debugger
- Shell scripts used by applications during development or execution
- Counterparts to UNIX daemons, as well as other utilities that perform application functions
- Script interpreters, such as Perl, REXX, and TCL

In addition, you will need to generate a list of application programming tools in use, such as NSAPI, Common Gateway Interface (CGI), Perl, C++, and Java. Next, decide if you will continue supporting them. Otherwise, make the transition to ISAPI or ASP, as described in "Migrating a Web Server to IIS 5.0" in this book. To help you with this list, you can use the "Tools Checklist" located in the ToolsChk.doc file on the Resource Kit companion CD.

Users

It's also helpful to make a diagram that depicts all system users, their locations, and the services they access. If your system serves a business or nonprofit group, you might find it useful to have both an organizational chart and an information flowchart to characterize where and how users currently access Web services. If you offer services as an Internet service provider (ISP), you need information about Web developers and general users accessing your system. To this diagram, you can add any additional capabilities that will result from the migration to IIS 5.0 as well as indicate any changes to these services. Figure 2.4 shows the information flow as external and internal users access the services and facilities of a commercial ISP.

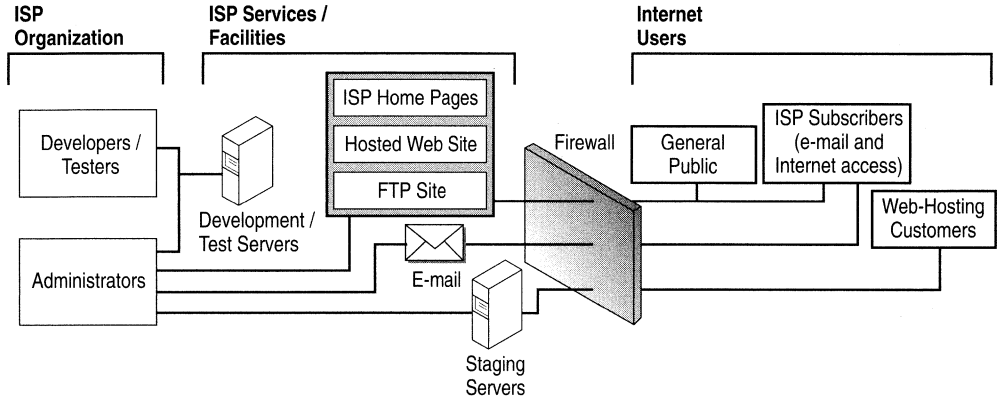


Figure 2.4 System Users of an ISP

Standards

Another area to assess is system operations standards, which usually take the form of policies and procedures. You might want to review and update your existing standards with reference to IIS 5.0. To help conduct a standards review, you can use the "Standards Review Form" found in the Standard.doc file on the Resource Kit companion CD.

Table 2.6 shows some issues to consider when reviewing standards and policies:

Table 2.6 Policy and Procedure Review

Policy/Procedure	More Information	Issues
Content Publishing	IIS 5.0 online product documentation	General guidelines on issues such as consistency in page formatting and style. Instructions on how to publish a Web page served by IIS 5.0, and directory access policies for Web Distributed Authoring and Versioning (WebDAV).
Disaster Recovery	"Capacity Planning" in this book	Recovery plans. Off-site or on-site spare server location. Spare server availability. Rehearsal provisions.
Internet Services	"Security" in this book	The person to contact for each service. Access control.
Maintenance	"Monitoring and Tuning Your Server" in this book	Preventative maintenance procedures. System monitoring policy and tools.
Netiquette		Guidelines for appropriate network and e-mail use.
Network	*Microsoft® Windows® 2000 Server Resource Kit TCP/IP Core Networking Guide*	Escalation procedures for downed links. Contacts for network services.
Redundant Servers	"Administering an ISP Installation" in this book	Fault-tolerant servers. Load-balancing servers.
Security	"Security" in this book	Password length and expiration period. Logon policies and auditing. Intruder prevention processes. Ownership/responsibility for user accounts. Methods for key encryption.

Define the New Service Offering

After gathering the necessary information about the network, tools in use, system users, and standards, the next step is to define the new service offering in more detail. This involves translating the requirements of the migration and conceptual design into the Functional Specification, which describes one or more solutions.

Functional Specification

The Functional Specification provides a detailed description of the IIS 5.0 solution. The team describes all system functionality at a high level and then works out details during the design process to complete the document. The immediate goal is to give the Functional Specification enough detail for Development, Testing, User Education, and Logistics Management to begin laying out project plans for their portion of the work. To help you begin this process, you can use the "Functional Specification Template" found in the FuncSpec.doc file on the Resource Kit companion CD.

The Solution

In addition to basic IIS 5.0 functionality, the solution you define might use related products that provide advanced services, such as SQL Server and Site Server. Fortunately, Microsoft applications were built to work together, making it easy to implement solutions. For example, you can use FrontPage wizards to insert a data call in a Web page so it interacts with SQL Server and an existing database. For more information about database applications, see "Developing Web Applications" in this book. For examples of IIS 5.0 solutions, see http://www.microsoft.com/backstage/.

You might plan to migrate to IIS 5.0 because of a specific requirement supported by a particular capability IIS 5.0 provides. However, the Planning phase is also an ideal time to assess any additional features of IIS 5.0 that could enhance your existing services. Rather than exactly replicating your current services, and substituting IIS 5.0 features in place of existing ones, you might want to consider adding new capabilities. In addition, you may need to consider retiring gopher services, which aren't supported by IIS 5.0, or making provisions to continue providing them through another server. Earlier in this chapter, Table 2.2, "Project Requirements and Deliverables" provided some examples of the deliverables that an IIS 5.0 migration project could encompass.

To compare IIS 5.0 features with those of other Web servers, follow the links to the Web server comparison page from the Windows 2000 Server Web site, located at http://www.microsoft.com/ntserver/.

Interoperation Plan

For various reasons, you might decide to make the transition to IIS 5.0 in phases, migrating only one Web server or group of servers at a time. This process can result in a mixture of platforms and Web servers operating on the same network for a period of time. If your planned server environment will be mixed, you need to develop a strategy that allows the different servers to interoperate, or coexist, on your network; the servers must also work in concert to provide Web services. This entails considering the best way to integrate the different components of the IIS 5.0 solution into your existing *service environment*, which is the combination of physical servers, network connections, software, and services or functions.

This strategy can be incorporated in an interoperation plan. Depending on your particular environment and needs, your plan might address the following issues:

- Enabling file sharing between UNIX- and Windows-based platforms
- Establishing common methods and policies for administering multiple Web server technologies
- Implementing common security methods and password synchronization

This chapter provides references to helpful information in these areas. In addition, Microsoft as well as third-party vendors offer a variety of tools and solutions for implementing a mixed server environment. For more information about tools, and for URLs to interoperation resources, see "Migrating a Web Server to IIS 5.0" in this book.

Detailed Design

Another key component of the Functional Specification is a detailed solution design. For the detailed design, you apply real-world technology constraints to the conceptual model, including implementation and performance considerations. This is a good time to reevaluate and refine your preliminary resource, cost, and schedule estimates.

The detailed design should include interfaces to the existing system, as well as provisions for meeting projected future needs in terms of physical servers and network bandwidth. Future capacity needs can be affected by growth in overall system usage as well as by user demand for increasingly rich content. For more information about this subject, see "Capacity Planning" in this book. In that chapter, the detailed design for services at http://www.microsoft.com is illustrated in Figure 4.9, "The microsoft.com Web Site." For developing the detailed design, you can use the "Design Template" found in the Design.doc file on the Resource Kit companion CD.

Solution Prototype

At this point, it's wise to perform proof-of-concept testing, in order to validate the technical features of your proposed solution. You don't need to test the complete solution, but you should set up a prototype of the production environment to demonstrate that your design adequately defines the required capabilities of the system. Prototyping allows predevelopment testing from many perspectives, especially usability, and helps create a better understanding of user interaction. The information gained from prototyping will also help you improve the Functional Specification.

Assess Resource Needs

Another important step is to identify the material and staff resources needed for the project. You must consider any software and hardware that you need to purchase, and whether you want to train current employees or hire consultants to fill gaps in expertise. From this assessment, you can develop a resource plan, cost estimate, and budget. There are several areas to consider when assessing resource needs, such as staff, tools, software, and hardware.

Staff

To implement a migration to IIS 5.0, the project staff needs to have the following skills:

- Knowledge of Windows 2000 Server networking and administration, including domain design, Web services, and security setup.
- Understanding of general internetworking concepts and Transmission Control Protocol/Internet Protocol (TCP/IP) technologies.
- Familiarity with Microsoft Internet technologies, including ASP.
- For migrations involving interoperation between UNIX and Windows 2000 Server, a firm grasp of the relevant concepts, and awareness of the available tools.
- For migrations from UNIX Web services, application developers should have the ability to port applications and utilities from UNIX to Windows 2000 Server.

These skills could already exist within your company. On the other hand, you might decide to hire a consulting firm to round out your capabilities or even to handle the entire migration effort. If your project is small, you might find all the talent you need in one or two experienced members of your staff.

Migration Tools

Migration tools are an important resource for expediting migration tasks. Listed here are a few you might want to have on hand.

IIS Migration Wizard

The IIS Migration Wizard, included on the Resource Kit companion CD, migrates server configuration settings to IIS 5.0 from Netscape Enterprise Server, Apache HTTP Server, and IIS 4.0, and replicates settings from one IIS 5.0 server to another. The wizard is described in "Migrating a Web Server to IIS 5.0" in this book.

Content Replication

For moving content to the new IIS 5.0 server from another Windows computer, you might want to use a replication tool such as Microsoft® Content Deployment, which is included with Site Server.

You might also consider acquiring FrontPage for migrating Web sites to IIS 5.0. By using the Microsoft® FrontPage Import Web Wizard, you can convert a Web site to a full-featured FrontPage web. When you do this, the wizard imports your pages, images, and files, while preserving the web's structure and hyperlinks. For more information about FrontPage, see http://www.microsoft.com/frontpage/.

Cross-Platform Administration

Microsoft has developed a set of utilities for configuring and administering servers in an environment that encompasses both UNIX-based and Windows-based computers. You can find more information about these and other tools in "Migrating a Web Server to IIS 5.0" in this book.

Other Tools

Other tools for converting CGI applications, debugging script, transferring data, testing, and more are listed in the "Additional Resources" section in this chapter and in "Migrating a Web Server to IIS 5.0" in this book.

Server Software

To migrate to IIS 5.0, the only server software you need is Windows 2000 Server, which includes integrated Web (HTTP), File Transfer Protocol (FTP), Internet mail, and local newsgroup (NNTP) services. For large installations, you might also want to consider obtaining Site Server or, for very large applications, Microsoft® Commercial Internet System (MCIS). Information about these products is available at http://www.microsoft.com/products/os.htm.

Hardware

The complexity of the migration and the number of servers involved dictates the hardware needed to create a development environment and test lab, as well as to deploy the new system. The following paragraphs discuss some considerations involved in planning hardware resources.

Development and Test Environment

A working development environment allows for proper development and testing of the solution so that it has no negative impact on production systems. You'll need to have sufficient development computers available for migrating and porting applications and for performing application-specific testing.

In addition, if your organization does not already have a suitable test lab in place, the team must build one. The test lab should contain servers that are representative of the new user environment. For example, if the new environment will be mixed, including computers that are running UNIX or Macintosh® operating systems along with computers that are running Windows 2000 Server, the test lab should reflect this.

The test lab consists of a unit lab and an integration lab. The unit lab is used for the initial preliminary evaluation and review of features, functions, and application software behavior. After applications are ported or developed, and then are evaluated on the development computer, they are tested in the unit lab.

Once applications are tested and approved at the unit lab, they are staged in the integration lab, so that users and other assigned teams can begin the evaluation process.

The integration lab should be made available for other project teams and users at different times during deployment. This gives design and testing teams an opportunity to review and address any immediate problems or issues found, without taking much time away from their overall project responsibilities.

The unit and integration lab environments should be identical in order to support evaluation of cross-router and cross-bridge connectivity. Two routers can divide the labs to simulate a wide area network (WAN). This way, design and testing teams can replicate production environments throughout the enterprise and eliminate potential problems during the testing process.

Depending on your user environment, each lab might be a pure Windows 2000 environment, or it might include both Windows 2000 and another network operating system. Desktop operating systems could include Microsoft® Windows® 95, Microsoft® Windows NT® 3.51 or Microsoft® Windows NT® 4.0, Windows 2000, Macintosh, and even some UNIX-based computers. Be sure to make each environment identical to the desktop configuration when Windows 2000 Server is deployed.

Figure 2.5 shows a unit and integration lab with a WAN link simulated by using two routers.

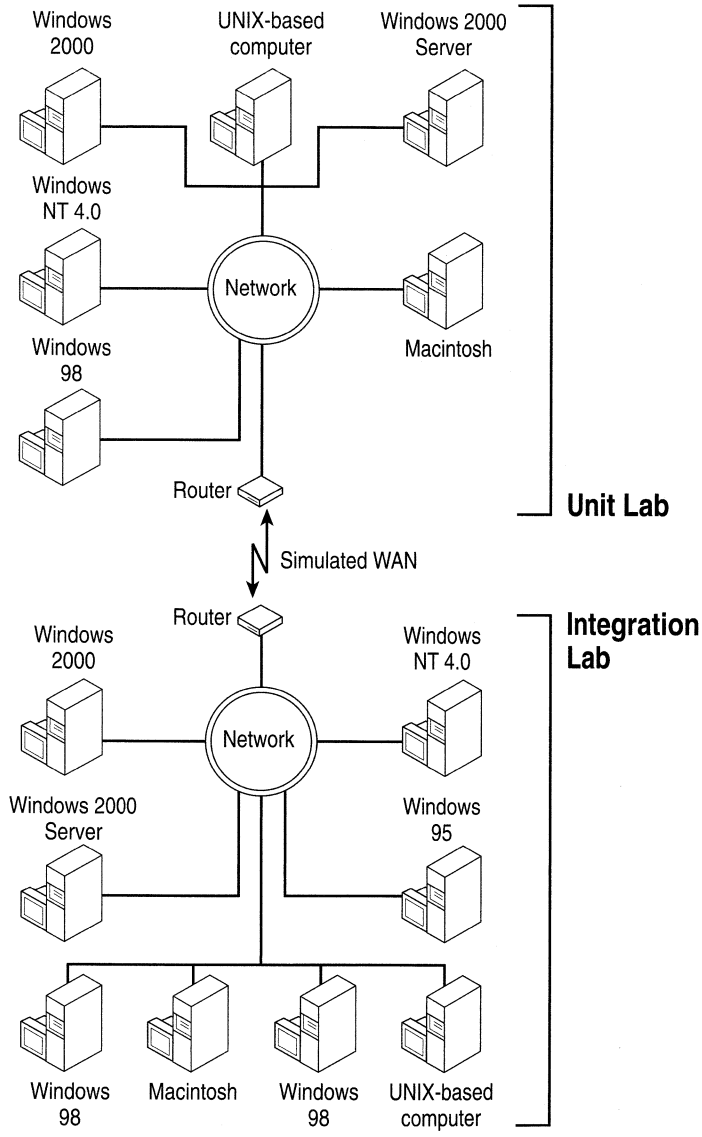


Figure 2.5 Example of a Test Lab Setup

Deployment Environment

It's a very good idea to set up a dedicated physical server for deployment, leaving your production Web server running until the new server is fully tested and debugged. This is especially true if you're migrating both your Web server and your operating system, such as when moving to Windows 2000 Server with IIS 5.0 from a UNIX computer running Apache; however, your job will be easier if you do this in all cases. This way you can set up users, groups, and permissions, and transfer all of your files, applications, and configuration settings without running the risk of losing your data in the event of a mishap. In addition, by doing this you can also thoroughly test the server and applications prior to deployment, which is highly recommended for mission-critical servers and applications.

For details on setting up a dedicated physical server for deployment, and for more information about assessing your deployment hardware needs, see "Migrating a Web Server to IIS 5.0" in this book.

Build the Master Project Plan

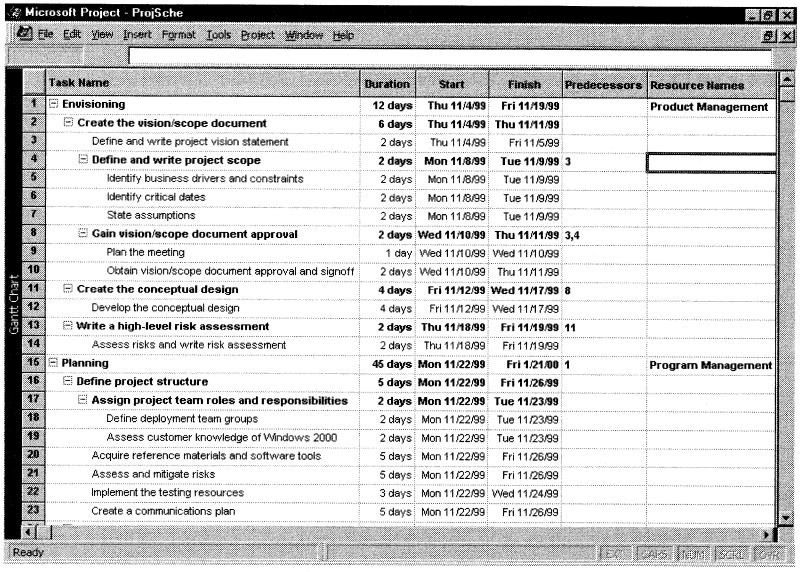
The Master Project Plan, which is basically a blueprint to follow throughout the project, includes all of the detailed plans contributed by each functional area of the project: Product Management, Program Management, Development, Test, User Education, and Logistics. Developing this plan prompts the team to consider all of the resources that must be assembled in advance, and alerts the team to potential pitfalls to avoid. A good plan steers the migration process, helping keep staff on schedule and on task, and the project within budget.

For developing your project plan, you can use the "Project Plan Template" in the ProjPlan.doc file on the Resource Kit companion CD.

Draft the Project Schedule

Your project schedule should include all of the detailed project schedules, including the release date. The team determines the release date after negotiating the Functional Specification draft and reviewing the Master Project Plan draft. It might be necessary for the team to modify some of the Functional Specification or Master Project Plan to meet a required release date. Although features, resources, and release date can vary, a fixed release date will help the team prioritize features, assess risks, and plan adequately. The key to project success is finding the right balance between resources, deployment date, and features.

Figure 2.6 shows a sample schedule. You can use the "Sample Project Schedule," in the ProjSche.mpp file on the Resource Kit companion CD, as a template for developing your schedule.



	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Envisioning**	**12 days**	**Thu 11/4/99**	**Fri 11/19/99**		**Product Management**
2	**Create the vision/scope document**	**6 days**	**Thu 11/4/99**	**Thu 11/11/99**		
3	Define and write project vision statement	2 days	Thu 11/4/99	Fri 11/5/99		
4	**Define and write project scope**	**2 days**	**Mon 11/8/99**	**Tue 11/9/99**	3	
5	Identify business drivers and constraints	2 days	Mon 11/8/99	Tue 11/9/99		
6	Identify critical dates	2 days	Mon 11/8/99	Tue 11/9/99		
7	State assumptions	2 days	Mon 11/8/99	Tue 11/9/99		
8	**Gain vision/scope document approval**	**2 days**	**Wed 11/10/99**	**Thu 11/11/99**	3,4	
9	Plan the meeting	1 day	Wed 11/10/99	Wed 11/10/99		
10	Obtain vision/scope document approval and signoff	2 days	Wed 11/10/99	Thu 11/11/99		
11	**Create the conceptual design**	**4 days**	**Fri 11/12/99**	**Wed 11/17/99**	8	
12	Develop the conceptual design	4 days	Fri 11/12/99	Wed 11/17/99		
13	**Write a high-level risk assessment**	**2 days**	**Thu 11/18/99**	**Fri 11/19/99**	11	
14	Assess risks and write risk assessment	2 days	Thu 11/18/99	Fri 11/19/99		
15	**Planning**	**45 days**	**Mon 11/22/99**	**Fri 1/21/00**	1	**Program Management**
16	**Define project structure**	**5 days**	**Mon 11/22/99**	**Fri 11/26/99**		
17	**Assign project team roles and responsibilities**	**2 days**	**Mon 11/22/99**	**Tue 11/23/99**		
18	Define deployment team groups	2 days	Mon 11/22/99	Tue 11/23/99		
19	Assess customer knowledge of Windows 2000	2 days	Mon 11/22/99	Tue 11/23/99		
20	Acquire reference materials and software tools	5 days	Mon 11/22/99	Fri 11/26/99		
21	Assess and mitigate risks	5 days	Mon 11/22/99	Fri 11/26/99		
22	Implement the testing resources	3 days	Mon 11/22/99	Wed 11/24/99		
23	Create a communications plan	5 days	Mon 11/22/99	Fri 11/26/99		

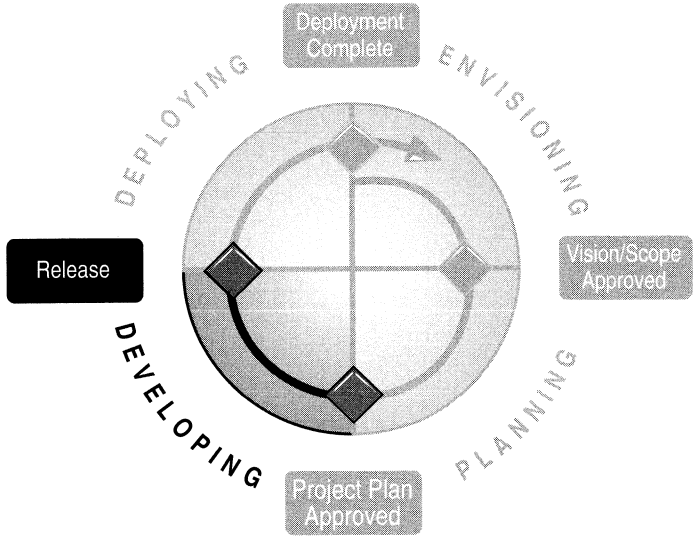
Figure 2.6 Sample Project Schedule

Check Your Assumptions

Before you begin putting your plans into action, develop a list of any assumptions, constraints, and dependencies on which the plans rely. Then check your assumptions to avoid costly delays later on. Examples of assumptions you'd want to verify are:

- A valid Windows 2000 Server network is in place.
- The project team knows how to install and configure Windows 2000 Server and IIS 5.0.
- Computer hardware meets the requirements listed at http://www.microsoft.com/products/.
- Required Windows 2000 Server counterparts to UNIX software are available.

3. Developing



An approved functional specification and associated project plan provide the baseline for focused development to begin. During the Developing phase, the Development team creates and validates the detailed design through testing and debugging. The system is built out, training of administrators and key users begins, and, finally, pilot testing is conducted.

The main milestone for the Developing phase is Release. At the Release milestone, the team assesses product functionality and verifies that rollout and support plans are in place. All new development is complete, and deferred functionality is documented for the next release.

Team Roles During the Developing Phase

During this phase, Program Management continues to take the lead. Table 2.7 shows the responsibilities of the project team during this phase:

Table 2.7 Team Roles During the Developing Phase

Role	Responsibility
Product Management	Developing and delivering communications to the user community outside the project team
Program Management	Functional specification management, project tracking, updated plans
Development	Solution and documentation
Test	Solution test, issue identification, documentation testing, updated test plans
User Education	Training, updated training plan, user communications
Logistics	Rollout checklists, updated rollout and pilot plans, site preparation checklists

Validate the Design

When the detailed design is complete, you're ready to validate it. Validating the design consists of performing unit, integration, and application testing to ensure the proposed system functions as expected and required. This testing is usually conducted in a lab environment similar to the one depicted in "Assess Resource Needs" earlier in this chapter.

Testing can be divided into four basic categories:

- **Unit Testing** Performed in the unit test lab, unit testing validates functionality of the Web server independent of other computers and systems.
- **Integration Testing** Performed in the integration test lab, integration testing ensures functionality, performance, and security of the Web server in the context of a network, and the platforms and servers with which the server will operate.
- **Application Testing** To verify Web application functionality and performance, application testing progresses from a development computer for basic functionality testing, to the unit lab, and finally to the integration lab to test its functionality in context.
- **Pilot Testing** Conducted by typical system users, usually on their own workstations, pilot testing demonstrates the usability of the Web services.

The testing process is iterative rather than linear. For example, you might discover a problem during an integration test, correct it, test the solution, and then go back to unit testing to find out if the correction has changed your results. In addition, it's frequently possible to conduct two or more procedures concurrently in order to complete testing at an earlier date.

For more information about testing and debugging applications, see "Developing Web Applications" in this book. In addition, see test planning information at http://www.microsoft.com/msf/.

For more information about improving server performance, see "Monitoring and Tuning Your Server" in this book. To develop your own test plans, you can use the "Sample Test Plan" found in the Testplan.doc file on the Resource Kit companion CD.

Testing at microsoft.com

The microsoft.com Web site provides a good example of the testing process for new servers and applications. Prior to adding a new IIS 5.0 production Web server, the microsoft.com team monitors and tunes a replica of the new server in a lab environment for a period of three months. Team members use a software and hardware configuration exactly replicating the production Web server that will exist on the microsoft.com Web site. Next, they move the server to the Microsoft intranet for a two-week period of further evaluation and tuning. After the tuning period, they replicate it to a nonmission-critical server on microsoft.com, running it in debug mode for one or two days to resolve any new problems that arise. Finally, they replicate the server to the actual production hardware, and then make it available site-wide. During the entire test process, the team monitors server performance continuously by using the HTTP Monitoring Tool, and uses the optimization techniques described in a white paper you can read at http://www.microsoft.com/ms.htm.

Another important concern for the microsoft.com team is ASP performance. To ensure good user experiences, the test team inspects all new ASP pages before they're published on the Microsoft Web site, noting the number of elements within each ASP page that have the potential to block performance, including objects such as Microsoft® ActiveX® Data Objects (ADO) and the ASP **FileSystemObject**. The test team then uses WCAT to test all pages that contain more than a certain threshold number of potentially troublesome elements. The team scores each page on Requests per Second, Maximum Response Time, and Average Response Time. If a page receives a poor score, the test team requires the owner to make improvements that will enhance performance.

Unit Testing

Unit testing involves the individual server outside the context of the network as well as other servers and operating systems with which it may coexist. If your migration involves multiple servers, you should migrate and test one server initially and ensure that it functions properly before proceeding to the others. This learning process will help you improve your approach for migrating subsequent servers. Table 2.8 describes some of the more important items to check during unit testing. You might want to include additional items in your own test process.

Table 2.8 Unit Tests

Unit Test	Description
User and Administrator accounts	Verify that user and group authentication information is accurate and complete. Verify that the correct user database is referenced by IIS 5.0 (local or domain).
Permissions	Verify file and directory permissions. Check access to files using different user accounts. Run ported applications and server extensions, such as CGI scripts and executables or ISAPI dynamic-link libraries (DLLs), to exercise Component Services settings and Execute/Script permissions.
File names and paths	Check for file-name conflicts and verify that file names and paths are correct. Verify that Windows conventions are used within migrated files, including referenced file names and paths, as discussed in "Migrating a Web Server to IIS 5.0" in this book.
Hyperlinks and page formatting	Run as http://localhost/ and verify hyperlinks. Also check for corrupt HTML that results in improper page formatting. Be sure to include ASP in this testing, if you are using it.
Applications	Verify that pooled, out-of-process, and in-process applications run correctly, as well as any applications that rely on a third-party script interpreter, such as Perl. Test any ported CGI or server-extending applications (EXE and DLL) to exercise Component Services settings and Execute/Script permissions.

Integration Testing

Once the server has passed unit testing, you're ready to evaluate it in a larger context of a network or server group during integration testing. Table 2.9 describes some common integration tests. You might want to include additional items in your own test process.

Table 2.9 Integration Tests

Integration Test	Description
Network identification	Verify that the server is correctly identified on the network.
Application integration	Test ASP applications that access backend databases or other remote objects, to verify that they function as expected and that permissions and script settings (such as time-out) are set correctly.
Stress, or load testing	Measure Web site performance, working with a replica of the site in a lab environment with multiple clients to simulate load on the servers. WCAT, a useful tool for this simulation, is provided on the Resource Kit companion CD. You can also check how individual Web sites are using the CPU by using IIS 5.0 process accounting, as described in the IIS 5.0 online product documentation.
Server availability	Measure availability of the server on the network by using the HTTP Monitoring Tool, which is included on the Resource Kit companion CD. It is described in a white paper you can read at http://msdn.microsoft.com/workshop/server/.
Performance monitoring and tuning	Monitor server performance by using the HTTP Monitoring Tool, as described in a white paper you can read at http://msdn.microsoft.com/workshop/server/. See also "Monitoring and Tuning Your Server" in this book and the "Calculating Connection Performance" topic in the IIS 5.0 online product documentation.
Security functionality	Test the various possible iterations of the system to verify that security performs as expected in each scenario. You can generate tests to exercise these system variations from a matrix that includes: • Each secured system component, such as Microsoft® Internet Explorer 5, IIS 5.0, and SQL Server. • Variants in the security implementation of each component; for example, browser Secure Sockets Layer (SSL) security (48 bit, 128 bit, or Server-Gated Cryptography [SGC]), IIS 5.0 authentication (none, Basic, or integrated Windows authentication), and so forth. • Communication protocols between system components, such as named pipes and TCP/IP sockets.
Security against penetration	Test security against intrusion, as described in "Security" in this book.

Application Testing

Prior to deployment, each application you migrate or port to IIS 5.0 should be tested on a server that's running Windows 2000 Server with IIS 5.0. It's best to begin this process well in advance of the final deadline for completion since Web applications can present unexpected problems.

In addition to testing for basic functionality, it's also important to test an application for performance. A poorly coded Web page can increase server response time and decrease the number of users served. Table 2.10 includes a few examples of application tests. Usually, code review is conducted on a development computer, then stress and performance testing move to the test lab. For more information about testing and debugging applications, see "Developing Web Applications" in this book.

Table 2.10 Application Tests

Application Test	Description
Code review	Check hyperlink references, keywords, and programming style. Make sure that any UNIX conventions are changed to Windows conventions, as described in "Migrating a Web Server to IIS 5.0" in this book. For ASP optimization tips, see "ASP Best Practices." For tips on reviewing ASP code, see http://msdn.microsoft.com/workshop/server/.
Load, or stress testing	Test the number of concurrent users the application can support. Verify that CPU and memory usage is acceptable under high loads. You can use the Web Application Stress Tool, included on the Resource Kit companion CD, for stress testing multitier ASP applications.
Performance testing	Test application performance under a variety of conditions. Test overall performance impact of the application on the server.
Application logic	Run as http://localhost/ and check for proper operation of application logic.

Build out the System

Once the detailed design has been tested and validated, you're ready to build out the system, configuring and locating the production Web servers that will be used on your network. When the Release milestone for the Developing phase has been achieved, you will be ready to deploy the new servers, as described in "Deploying" later in this chapter.

Begin Training

As soon as possible, begin training administrative staff, support personnel, and key users. Any staff members who will be working with IIS 5.0 during or after the migration will benefit from taking a Microsoft-certified course in IIS 5.0. In addition to training the individuals who will be performing the migration and administering servers, you'll probably want to provide end-user training and support. Web developers will also benefit from training on subjects such as creating ASP-based applications and using FrontPage to publish Web pages.

For more information about available Microsoft training, see http://www.microsoft.com/train_cert/.

For information about other books and for training materials on Microsoft products, see http://mspress.microsoft.com/.

Conduct Pilot Testing

Pilot testing involves having a group of end users try the system prior to its full deployment in order to give feedback on IIS 5.0 features and functions. The level of pilot testing you want to perform depends on the size and scope of your migration project. For larger projects, a formal, carefully planned pilot is essential. For any size project, it's good to have selected end users test the system prior to full deployment.

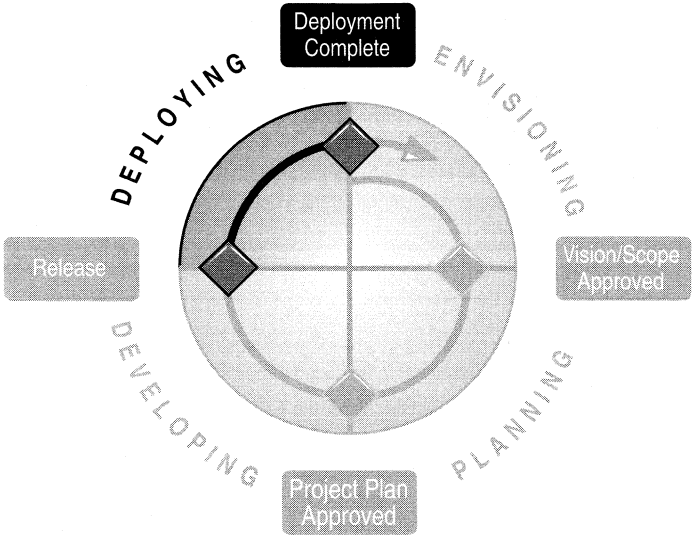
During the initial, pre-pilot phase you can select a small group of technically savvy users to try out the technology. You probably won't want to provide support during this phase, because it can draw resources away from system development and burden your schedule.

After making adjustments based on input from the pre-pilot users, you can begin the actual pilot. During this phase, a larger group of users should review and fully use system features. These users should be at about the same technical level as your system users in general. During this phase, you should plan to provide support for all issues, errors, or problems that users report. When you make any corrections, be sure to thoroughly retest the system.

The following are items to include in planning a pilot test:

- Adequate training for participants
- A rollout plan for deploying the servers and preparing systems for the pilot
- Documentation of the installation process so you can improve it as more is learned
- A mechanism, such as a Web site or e-mail alias, for users to provide constant feedback to the design and testing teams
- Evaluation criteria for the pilot, including information about the number of users who were dissatisfied, the number of problems reported, the number of support calls and requests, and the resolution rate for problems

4. Deploying



The final phase of the migration process, in the MSF process model, is Deploying. During this phase, the team must transition from creating elegant development solutions to complying with the rigid operational requirements of thorough and complete testing. The goal of this phase is to make the new services available to the target audience and users. This requires completing user and administrator training, rolling out and monitoring the system, and fixing bugs.

The milestone for this phase is Deployment Complete, when the system can be formally turned over to Operations and Support, for maintenance.

Team Roles During the Deploying Phase

During this phase, a subtle shift occurs in the dynamics of the project team. Here, the Logistics Manager will step to center stage. The Logistics Manager will often share the focus of the project with User Education as the systems are actively deployed, as users receive the appropriate training, and as the operations and support systems are activated. Table 2.11 shows the responsibilities of the project team during this phase:

Table 2.11 Team Roles During the Deploying Phase

Role	Responsibility
Product Management	Promotion, feedback, assessment, sign-off
Program Management	Solution/scope comparison, stabilization management
Development	Problem resolution, escalation support
Test	Performance testing, problem identification
User Education	Training, training schedule management
Logistics	Site deployment management, change approval

Finish Training

Prior to rolling out the new system, you need to be sure that all system administrators, support staff, and users have completed training. In addition, user support systems should be in place.

Roll out the New System

Before making the new system available to all prospective users, you can use the checklist in Table 2.12 to verify that you've completed the minimum set of tasks required for a successful rollout.

Table 2.12 Rollout Checklist

√	Task	Reference
	Migrate server configuration settings.	"Migrating a Web Server to IIS 5.0" in this book.
	Migrate Web and FTP site content.	"Migrating a Web Server to IIS 5.0" in this book.
	Migrate Web applications.	"Migrating a Web Server to IIS 5.0" in this book.
	Prepare the network including implementation of a valid Windows 2000 Server domain and network topology.	Windows 2000 Server online product documentation.
	Install administrative tools and utilities.	"Migrating a Web Server to IIS 5.0" in this book.
	Implement required policies and procedures.	"Standards" earlier in this chapter.
	Install tools, utilities, and applications needed by Web developers and end users.	"Migrating a Web Server to IIS 5.0" in this book.
	Thoroughly test components, system, pages, and applications.	"Validate the Design" earlier in this chapter.
	Provide training on the new system for all administrators, developers, and end users.	"Staff" earlier in this chapter.
	Make technical support available for all users.	"Staff" earlier in this chapter.

Once you've completed the rollout checklist, you're ready to make the new IIS 5.0 server available to users on the network. For a large-scale migration, or one involving mission-critical applications, you might deploy the server in steps, making its services available to progressively larger groups of users over time. For an example of how to do this, see the sidebar "Testing at microsoft.com" earlier in this chapter.

There are several technical approaches to taking a production Web server out of service and bringing the new server onto your network to begin hosting. The method you choose should be based on your network policies, amount of acceptable downtime, and your technical knowledge. Here are two possible methods:

- **If some downtime is acceptable,** power down the existing production Web server and simultaneously start up the new IIS 5.0 server using the IP address and network settings of the existing production Web server. You might experience several minutes of server downtime; however, problems should be minimized if you perform the switch during off-peak hours and inform users several days in advance.
- **If no downtime is acceptable,** assign the new IIS 5.0 server a new IP address, and set the transistor-transistor logic (TTL) units in the Domain Name System (DNS) records so that they will expire in 24 hours. Set up round-robin DNS so requests will be sent sequentially to the production Web server and to the new IIS 5.0 server. Wait for the DNS to propagate. After the first propagation, turn off the round-robin setting on the production Web server, and wait for propagation to occur again. When you're sure the new IP has been propagated, turn off the production Web server. However, exercise caution if you're using tracking or dynamic capabilities on your site, as problems could occur.

  Of course, if you're using a server clustering tool such as Network Load Balancing, all you need to do is assign the new server the appropriate IP address and bring it onto the network, and then take the old production Web server offline.

Monitor the System

It's important to monitor the performance of the new IIS 5.0 server, especially in the first few days after deployment, to ensure your users are not encountering problems when visiting your Web sites. During this time, you'll also want to fine-tune server performance. To track server performance, use either the System Monitor in Microsoft® Windows® 2000 Server, or the HTTP Monitoring Tool, the latter of which is included on the Resource Kit companion CD. Performance degradation can alert you to a serious problem. Some suggested items to monitor are processor use, memory use, disk use, and network availability. For more information about monitoring, see "Monitoring and Tuning Your Server" in this book.

No matter how carefully you've planned and carried out the migration, you're bound to run into a glitch or two following system deployment. There are a number of resources to help you troubleshoot problems that occur with the new IIS 5.0 server. For example, by participating in a related newsgroup, such as the newsgroup sponsored by 15seconds.com, you can get troubleshooting information from other IIS 5.0 users. In addition, Microsoft provides searchable reference information that may be of help at http://support.microsoft.com/support/.

For finding information about debugging applications, try using the search string "application debugging."

Additional Resources

Described below are additional resources that can help you perform a migration. Also listed are Web sites and books that provide additional information about IIS 5.0 and about other features of Windows 2000 Server.

Support Documents

The following support files, which were referred to in this chapter, can be found on the Resource Kit companion CD. They are available in .rtf format as well.

- **Design.doc** Design Template
- **FuncSpec.doc** Functional Specification Template
- **OrgEnvir.doc** Organization Environment Survey
- **ProjPlan.doc** Project Plan Template
- **ProjSche.mpp** Sample Project Schedule
- **RiskAsmt.doc** Risk Assessment Form
- **Standard.doc** Standards Review Form
- **TechEnvi.doc** Technical Environment Survey
- **TestPlan.doc** Sample Test Plan
- **ToolsChk.doc** Tools Checklist
- **VisScope.doc** Vision and Scope Template

IIS 5.0 Migration Tools

IIS Migration Wizard

Included on the Resource Kit companion CD, the IIS Migration Wizard automatically migrates most configuration settings, as well as user and group information, to IIS 5.0 from the following Web servers:

- Netscape Enterprise Server 3.5
- Apache HTTP Server 1.3
- IIS 4.0
- IIS 5.0 (from one computer to another)

For more information about the IIS Migration Wizard and for tables of migrated configuration settings, see "Migrating a Web Server to IIS 5.0" in this book.

> **Note** The IIS Migration Wizard is provided for instructional purposes only and is not supported by Microsoft.

Microsoft Interoperability Lab

For more information about general cross-platform interoperability and coexistence, visit the Microsoft Interoperability Lab at http://mspress.microsoft.com/reslink/common/net/.

Other Migration Tools

http://www.datafocus.com/products/tk/
: Mortice Kern Systems makes the MKS Toolkit, which provides Windows scripting and migration tools. This set of more than 210 UNIX and Windows utilities allows you to create scripts and use familiar UNIX commands on your PC. It also automates tasks like updating pages on a Web site, manipulating files and text, querying a database, and accessing and updating the Windows registry.

http://www.shareware.com/
: At this site you can obtain a tool called Ws_ftp that performs recursive FTP for Windows.

Migration and Integration Resources

Migrating to Windows NT by Steve Heath, 1997, Houston: Digital Press.
: A handbook to ease the transition from another operating system to Windows NT. Discusses the user interface, accessories, networking abilities, and other aspects of Windows NT. Includes chapters dealing with transitioning from UNIX, Macintosh, MS-DOS, and Windows 3.x.

Windows NT & UNIX, Administration, Coexistence, Integration, & Migration by G. Robert Williams et al, 1998, Reading: Addison Wesley Longman, Inc.

Topics covered include porting applications, TCP/IP, Common Object Request Broker (CORBA) and Microsoft® Distributed Component Object Model (DCOM) interoperability. Also discusses various e-mail systems, user interface emulators, Portable Operating System Interface UNIX (POSIX) commands and utilities, and clustering technologies. Includes a quick reference guide to common Windows NT and UNIX commands and utilities.

Windows NT, UNIX, NetWare Migration and Coexistence, A Professional's Guide by Raj Rajagopal, 1998, Perth: CRC Press LLC.

Identifies and classifies the solutions and products that support migration and coexistence between UNIX, Windows, and NetWare. It discusses porting applications as well as developing new applications by using cross-platform development techniques, and also provides guidelines for selecting a solution.

Windows NT & UNIX Integration Guide by David Gunter et al, 1997, Maidenhead: Osborne McGraw-Hill.

The CD-ROM contains freeware, shareware, and demonstrations of commercial products designed to assist Windows NT and UNIX integration efforts. It also contains a selection of FAQs for easy reference.

Planning and Testing Resources

Software Project Survival Guide by Steve McConnell, 1998, Redmond: Microsoft Press.

This book provides a complete set of guidelines for a successful development project.

http://www.construx.com/survivalguide/

This Web site provides helpful information and tools on software project planning, much of which is relevant to a migration project.

Software Testing in the Real World, Improving the Process by Edward Kit, 1995, New York: ACM Press.

This is a general guide to software testing with many useful appendices that include sample development, test plan outlines and reports.

Software Verification and Validation, A Practitioner's Guide by Steven R. Rakitin, 1997, Norwood: Artech House, Inc.

This is a general guide to software testing that includes many useful references to additional resources, information about test tools, and sample verification exercises.

Test Tools

Here are various tools that you can use for testing:

- Web Capacity Analysis Tool (WCAT), found on the Resource Kit companion CD, allows you to stress test the Web server by sending multiple client requests to simulate load on the servers.
- The Web Application Stress Tool, found on the Resource Kit companion CD, is designed to realistically simulate multiple browsers requesting pages from a Web application. It covers the features most needed for stress testing three-tier personalized Active Server Page Web sites running on Microsoft® Windows NT® Server 4.0 and Windows 2000 Server.
- The HTTP Monitoring Tool, found on the Resource Kit companion CD, monitors HTTP activity on the server.
- The IIS 5.0 debugger component, which is built into IIS 5.0, can be used for debugging applications.
- The Microsoft® Component Object Model (COM) application debugger is a component of Windows 2000 Server.

For more information about testing tools, see http://www.microsoft.com/windows/server/default.asp.

General Administration Books and Training

Inside Windows NT, Second Edition by David A. Solomon, 1998, Redmond: Microsoft Press.

Essential Window NT System Administration by AEleen Frisch, 1998, Sebastopol: O'Reilly & Associates, Inc.

http://www.microsoft.com/msf/
This Web site provides information about training on the Microsoft Solutions Framework.

http://www.microsoft.com/ISAPI/train_cert/catalog/findcrse.asp
To find online training resources for Windows 2000 Server, select Windows 2000 Server as the product category.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 3

Migrating a Web Server to IIS 5.0

The previous chapter, "Managing the Migration Process," gave an overview of the activities and planning steps involved in migrating Web services to IIS 5.0. The present chapter takes a "nuts and bolts" approach to the subject, providing information about migrating configuration settings, content, and applications to IIS 5.0 from another type of Web server. In addition to including the basic steps to accomplish this, the chapter explains using the IIS Migration Wizard to replicate configuration settings and content from a Web server running Apache HTTP Server, Netscape Enterprise Server, IIS 4.0, or IIS 5.0 itself to one running IIS 5.0. Should you plan to continue running another type of Web server along with IIS 5.0 on your network, you'll find information about interoperation methods and tools as well.

This chapter also reviews several approaches to migrating a Web application to IIS 5.0. It suggests considerations for deciding whether or not to rewrite a Common Gateway Interface (CGI) application as an Internet Server Application Programming Interface (ISAPI) extension or Active Server Pages (ASP) application in order to improve its functionality and performance.

Before migrating a Web server to IIS 5.0, you should know how to upgrade, configure, and administer the Microsoft® Windows® 2000 Server operating system and should have implemented a valid Windows 2000 Server domain and network design. For references on these subjects, see "Additional Resources" at the end of this chapter.

In This Chapter

Basic Steps to Migrate a Web Server

Migrating From Apache HTTP Server

Migrating From Netscape Enterprise Server

Upgrading or Replicating an IIS Web Server

Migrating Web Applications

Additional Resources

Basic Steps to Migrate a Web Server

Once you've completed the planning and preparatory steps described in the previous chapter, you're ready to begin migrating your Web server to IIS 5.0. This section describes the basic steps required to migrate from almost any type of Web server. Later sections provide details about migrating from Apache HTTP Server and Netscape Enterprise Server, as well as upgrading from IIS 4.0 to IIS 5.0 or replicating settings and content from one IIS 5.0 Web server to another.

The following steps are discussed in this section:

- Assessing hardware requirements and acquiring any new hardware needed for deployment.
- Preparing the destination computer by creating partitions, installing Windows 2000 Server with IIS 5.0, and structuring directories.
- Using the IIS Migration Wizard.
- Migrating Web and File Transfer Protocol (FTP) sites by replicating their content to the destination computer, correcting any problems with the files, and implementing special features.
- Replicating Web application files to the destination server, configuring the applications, and installing and configuring any necessary script interpreters. Additional steps that might be required to migrate CGI applications to IIS 5.0 are discussed in "Migrating Web Applications" later in this chapter.
- Migrating log files.
- Migrating Web server configuration settings.
- Securing the new Web server by importing user and group information, setting permissions, installing security certificates, and configuring other security parameters.

Note You can use the IIS Migration Wizard, included on the *Microsoft® Windows® 2000 Server Resource Kit* companion CD, to replicate Web site content and migrate Web server settings from Netscape Enterprise Server 3.5 or Apache HTTP Server 1.3. You can also use the wizard to upgrade a server running IIS 4.0, or to replicate a server running IIS 5.0.

Assessing Hardware Requirements

The first thing you must do is decide what type of hardware to use for the migration, and then obtain it and set it up. There are four different approaches to migration, and each one has a different implication for deployment hardware. This section describes the pros and cons of each approach and the hardware required. Subsequent sections give more information about performing the migration steps mentioned here.

1. **Migrate to a clean installation of Windows 2000 Server and IIS 5.0.** Perform a clean installation of Windows 2000 Server and IIS 5.0 on a computer other than the production Web server. Migrate settings, content, and applications from the production Web server to the new IIS 5.0 server. Test and debug the new server before deploying it and taking the old production Web server offline.

   Hardware needed: A second computer is required, in addition to the existing production Web server.

   Pros: You can opt to put new, updated hardware in place at the same time you perform the migration. You also avoid taking your production Web server offline until the new server is tested and deployed. Following deployment, if problems arise with the new server that didn't appear during testing, you can use the original server as a backup.

   Cons: You might need new hardware. However, the cost will be at least partly offset by the time saved conducting the migration as well as troubleshooting any problems that occur during the migration.

   Recommendation: This method is highly recommended because it has the least likelihood of resulting in unforeseen problems. Because of the difficulty in conducting a cross-platform migration on a single computer, this is the only approach recommended for migrating a UNIX-based Web server to IIS 5.0.

2. **Migrate a duplicate of a Windows NT–based Web server.** If you're migrating from a computer running Microsoft® Windows® NT, you can use this approach: On a second computer, install the same software as exists on the production Web server, and then use the Windows 2000 Server Setup Wizard in order to upgrade to Windows 2000 Server and IIS 5.0. Migrate Web configuration settings, content, and applications to the new Web server. Test and debug the new server before deploying it and taking the old production Web server offline.

   Hardware needed: A second computer is required, in addition to the existing production Web server.

   Pros: You can opt to put new, updated hardware in place at the same time you perform the migration. You also avoid taking your production Web server offline until the new server is tested and deployed. Following deployment, if problems arise with the new server that didn't appear during testing, you can use the original server as a backup.

   Cons: You might need new hardware. However, the cost will be at least partly offset by the time saved conducting the migration as well as troubleshooting any problems that occur during the migration.

   Recommendation: This approach is acceptable, but it requires the preliminary step of installing the software on the new server. It is only feasible for a migration from a Windows NT–based server.

3. **Migrate a mirror of a Windows NT–based Web server.** If you're migrating from a computer running Windows NT, you can use this approach: Create a mirror image of the current production Web server (operating system, software, configuration settings, and content) on a second computer. Then use the Windows 2000 Server Setup Wizard to upgrade it to Windows 2000 Server and IIS 5.0. Migrate Web configuration settings, content, and applications to the new Web server. Test and debug the new server before deploying it and taking the old production Web server offline.

   Hardware needed: For this approach to be successful, you need hardware exactly duplicating that of your existing server.

   Pros: You avoid taking your production Web server offline until the new server is deployed. Following deployment, if problems arise with the new server that didn't appear during testing, you can use the original server as a backup.

   Cons: All hardware on the second system must *exactly* duplicate the original server, so you forego having the option to upgrade your hardware at the same time you migrate or upgrade the Web server.

   Recommendation: This method is fine if you don't need to upgrade your hardware, and it's far less risky than the next approach described. It is only feasible for a migration from a Windows NT–based server.

4. **Migrate the production Web server.** Take your production Web server offline. If it's already running Windows NT Server, use the Windows 2000 Server Setup Wizard to upgrade it to Windows 2000 Server with IIS 5.0. For a new installation, install Windows 2000 Server on a primary disk partition, which may require reformatting and repartitioning of the hard disk. Migrate Web configuration settings, content, and applications to IIS 'in place' on the production Web server. Test and debug the server before deploying it.

   Hardware needed: No new hardware is required.

   Pros: There is no hardware cost.

   Cons: Migrating a production Web server is extremely risky. You must take the server offline, and it will not be available to users until you complete all migration tasks, testing, and debugging.

   Recommendation: This method is *not* recommended.

Preparing the Destination Server

Once you've acquired and set up the necessary hardware, you need to take several steps to prepare the destination server for migration, including installing Windows 2000 Server and IIS 5.0, configuring directories, and installing tools and utilities.

Note The *destination server* (sometimes called the *target* server) is running Windows 2000 Server with IIS 5.0, and the *source server* is running the Web server software you plan to migrate. While source and destination servers could exist on the same physical computer, carrying out a migration on a single computer is *not* recommended. (The issues involved are described in the previous section, "Assessing Hardware Requirements.") Therefore, in this discussion, source and destination servers are assumed to exist on separate computers.

1. **Back up the source server.** Before you begin, create a backup file of all source server configuration settings, as well as files, applications, utilities, and other software used by your Web and FTP sites.

   It's always good policy to have a backup of the server even when you are migrating to a separate computer. This step is *critical* if you plan to perform the migration on the actual production Web server (which, again, is *not* recommended). If anything goes wrong, you can then restore the original system and content.

2. **Identify items to migrate.** Decide which items to migrate from the source server. Note any applications and scripts you need to modify in order for them to run on IIS 5.0. To migrate these items, you need to transfer them to a development computer for modification, then to a test server for testing, and finally to the production Web server for deployment. For more information, see "Migrating Web Applications" later in this chapter.

3. **Estimate disk space and partition requirements.** Estimate the total disk space needed for data, files, applications, and server software on the destination computer. For each item, make a note of the disk partition on which it will reside (some considerations are discussed in the next step). Make sure the destination computer has sufficient hard disk space for your requirements.

4. **Install Windows 2000 Server and IIS 5.0.** Install Windows 2000 Server and IIS 5.0 on the destination computer. For more information, see the Windows 2000 Server product documentation. When installing to a new computer, choose the clean install option. By default, Windows 2000 installs as a file server. If you are using your server primarily as a Web server, you should install it as an *application server.*

During setup you create disk partitions and specify the file format for the primary, or *system*, partition onto which Windows 2000 Server will be installed. Be sure to allocate sufficient space in each partition for the content or system files it will hold. The following are some other issues to consider when creating disk partitions and specifying file format:

- **Security** To implement the highest level of security, format the partition onto which you install Windows 2000 Server as the NTFS file system. For more information about NTFS and security, see "Security" in this book.

  For security reasons, it is also a good idea to install the Windows 2000 Server and IIS 5.0 system software on a separate partition from the Web and FTP site content and applications. A good configuration might be as follows:

  Drive C: Allocate 1.5 to 2 GB of disk space for Windows 2000 Server and IIS 5.0 executables.

  Drive D: Allocate 1 to 2 GB of disk space for the IIS root and Web site directories.

  Drive E: Allocate sufficient disk space to contain remaining software and content for your server.

- **Access from UNIX-based Computers** Keep in mind that only Windows NT– or Windows 2000–based computers can natively access data and files stored on NTFS partitions. Therefore, when using NTFS in a mixed environment, you need to take additional steps to make sure files are available to UNIX-based computers. There are a couple of ways to do this. You can use Microsoft® Network File Sharing (NFS) client and server components that run on Windows 2000 and that support cross-platform file sharing. These and other useful tools and utilities are available as part of Windows NT Services for UNIX. For more information about these utilities, see http://www.microsoft.com/ntserver/nts/exec/overview/sfu.asp.

  Likewise the SAMBA shareware application suite allows UNIX clients to access Windows-based file systems. It also allows Windows-based clients to access files and printers on UNIX, NetWare, OS/2, or VMS servers that support the Server Message Block (SMB) protocol. For more information about SAMBA, see http://samba.anu.edu.au/samba/.

For more information about creating and configuring Windows 2000 operating system partitions and directories, see the Windows 2000 Server online product documentation.

5. **Configure Windows 2000 Server.** After installation, the next step is to configure Windows 2000 Server networking and security, and any additional services. For references on setup and administration, see the Windows 2000 Server online product documentation, as well as the "Additional Resources" section at the end of this chapter.

6. **Structure Web and FTP site directories.** When you install IIS 5.0 for the first time, the Windows 2000 Server setup program creates a Web server directory structure in Windows Explorer at c:\Inetpub\wwwroot\. This is where actual Web site content and application files are stored. IIS 5.0 also creates a Default Web Site in the IIS snap-in of Microsoft® Management Console (MMC). It points to the content stored in the root directory and provides configuration settings *(properties)* for the Web site and its associated files. IIS 5.0 gives the Default Web Site the same name as the computer running IIS 5.0, unless the computer Internet Protocol (IP) address is registered with a different name on the Internet or your corporate network.

   Figure 3.1 shows the Wwwroot directory in Windows Explorer and its associated Web site (in this case, the Default Web Site) in the IIS snap-in.

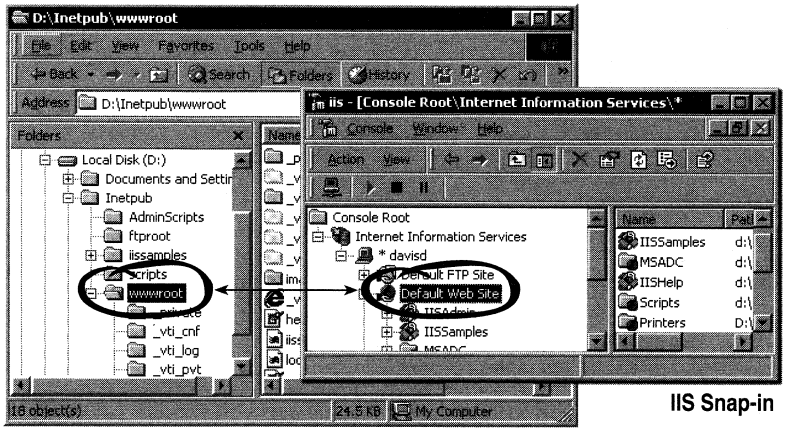


Figure 3.1 The Wwwroot Directory in Windows Explorer and the Default Web Site in the IIS Snap-in

You might be able to use the default directory structure as it is, or you might want to make some modifications to preserve paths (also called *pathnames*) within your content and application files. As much as possible, try to replicate the directory structure and names used on the source server. This minimizes broken links and the need to update URLs and pathnames within files following migration.

> **Note** The Default Web Site in the IIS snap-in points to the IIS 5.0 online product documentation, supporting code, and application samples (accessible from the IIS snap-in Help menu, or by typing http://localhost/ in the address bar of your browser). If you delete the Default Web Site, you will no longer be able to open these items. Also, be aware that the Default Web Site is assigned port 80 by default. When you create a new Web site in the IIS snap-in, it also will be assigned port 80 by default. In order to continue accessing the documentation, you must change the port number of any new Web site you create to something besides 80 so that it doesn't conflict with the Default Web Site. To avoid these problems, it is best to use the Default Web Site to manage your Web site, rather than creating a new one. For more information, see the "Adding Sites" topic in the IIS 5.0 online product documentation.

Storing Content outside the Web Server Root Directory

The Virtual Directory feature of the IIS 5.0 snap-in allows you to publish content to your Web and FTP sites that is stored outside of the Web server root directory tree (c:\Inetpub\wwwroot) in Windows Explorer. Content—HTML files, scripts, images, and other files—can be stored in any Microsoft® Windows® directory on the local computer or another computer on your network. The only restriction is that the network drive containing the directory must be in the same Windows 2000 Server Domain as the computer running IIS 5.0. For more information about virtual directories, see the "Creating Virtual Directories" topic in the IIS 5.0 online product documentation.

7. **Install tools and utilities.** Next, install tools and utilities on the destination server. These include all of the items you listed during the planning phase that must reside on the new server. If you're migrating from UNIX, you'll want to obtain and install the Windows 2000 Server counterparts of UNIX tools and utilities. Many of these are available on the Resource Kit companion CD. Others are available with Windows NT Services for UNIX. For more information, see http://www.microsoft.com/ntserver/nts/exec/overview/sfu.asp.

   You'll also want to obtain and install the appropriate script interpreters (also called *scripting engines*) to run applications written as scripts. IIS supports any scripting language for which you have installed a script interpreter that follows the Active Scripting standard. For ASP applications, IIS natively supports Microsoft® Visual Basic® Scripting Edition (VBScript) and Microsoft® JScript®, so you don't need to install their interpreters. Active Scripting interpreters for Perl 5.0 and Regina REXX are included on the Resource Kit companion CD and are also available through third-party developers. TCL/Tk, Python, and REXX script interpreters that are compatible with Microsoft® Win32® are available from third-party sources. In most cases, you should install the script interpreter in the Web server root directory (c:\Inetpub\wwwroot) and set appropriate NTFS and IIS 5.0 permissions. For more information, see "Configuring a Script Interpreter" and "Additional Resources" later in this chapter.

Using the IIS Migration Wizard

At this point, if you're migrating from Netscape Enterprise Server 3.5 or Apache HTTP Server 1.3, you can use the IIS Migration Wizard, included on the Resource Kit companion CD, to migrate Web site content and configuration settings. The wizard can also perform a cross-machine upgrade from IIS 4.0 to IIS 5.0, or replicate an IIS 5.0 Web server. Instructions for using the wizard are included on the Resource Kit companion CD.

You will still need to make necessary corrections to UNIX files to make them compatible with Windows conventions, as described later in "Converting UNIX File Names and Pathnames" and "Special Considerations for UNIX Applications." While the wizard replicates application files, you might need to take additional steps to migrate them, as described later in "Migrating Web Applications." In addition, you'll need to configure applications and components, as described in the "Configuring Applications" topic of the IIS 5.0 online product documentation.

Migrating Web and FTP Sites

The next step in migration is to populate the newly created IIS directories with Web and FTP site content by replicating these items from the source server to the destination server. Then you'll take additional steps to complete their migration by creating virtual directories, correcting file formatting problems, repairing broken hyperlinks, and implementing special features, such as content expiration, footers, and server-side includes.

Replicating Windows-Based Files

You can copy files and directories between two Windows-based computers by using FTP, a serial connection, or the rcp utility that comes with the Windows 2000 operating system, or by creating a shared directory and copying the files across your local area network (LAN).

Keep in mind that a simple file copy and paste operation does not preserve some file data, such as security configuration, hyperlinks, and share information. The following paragraphs discuss some methods of preserving file data.

Preserving Permissions and Audit Settings

To copy files while retaining current Windows permissions and audit settings, you can use the XCopy utility that comes with Windows 2000 Server, by typing the following at the command prompt:

XCopy <current>:\<dir> <new>:\<dir> /o /a /s

For more information about using this tool, see the Windows 2000 Server online product documentation.

Preserving Hyperlinks and Web Structure

Both Microsoft® FrontPage® and Microsoft® Visual InterDev® Web development systems can replicate Web sites from one Windows-based computer to another, while preserving site directory structure and hyperlinks. However, you will need to recreate virtual paths, as these tools don't import this information.

Preserving Share Information

Copying files and directories is simple enough; however, share information isn't maintained in Windows directories, but rather in the registry, under LanmanServer. To preserve share information, you must also replicate this registry information from the source server currently hosting the shared files and directories, to the destination computer, as follows:

> **Note** The action described next will destroy any existing shares on the destination computer. Be sure to back up the registry before you begin.

Using the following registry key, save the source computer registry settings for shares to a file. Use Regedt32.exe, and not Regedit.exe, to edit the registry.

```
HKEY_LOCAL_MACHINE
        \System
                \CurrentControlSet
                        \Services
                                \LanmanServer
                                        \Shares
```

Copy the resulting settings file to the destination computer and then use the following registry key to restore the share settings to the destination server from this file. The settings will take effect after you restart the computer.

```
HKEY_LOCAL_MACHINE
        \System
                \CurrentControlSet
                        \Services
                                \LanmanServer
                                        \Shares
```

> **Caution** Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

Replicating UNIX-Based Files

You can copy files and directories (directories are also called *folders* in Windows) from a UNIX-based computer onto disks or tape media, or move them across a network connection by using FTP. To transfer an entire Web directory tree in one file, you can use Tar.exe for concatenating files and directories, compress them with one of several available utilities, and then use recursive FTP to transfer them to Windows 2000 Server. SAMBA is also useful for transferring files between computers running UNIX and Windows operating systems.

You can also copy files across a network by using the Windows 2000 rcp client to access a UNIX computer that's running the rcp daemon called rshd (Remote Shell Daemon). With rcp you can specify security parameters, as well as recursively copy files and directories between source and destination computers. In this case, the name of the Windows 2000 Server computer must appear in the .rmhosts file on the UNIX computer. For more information, see the "RCP" topic in the Windows 2000 Server online product documentation.

> **Note** Microsoft® MS-DOS and Windows files use a carriage return and line feed character to mark the end of each line, while UNIX files use only a line feed character. The Windows 2000 rcp client automatically makes this conversion for you when run in ASCII transfer mode (the default).

Once the files are copied to Windows 2000 Server, you can separate any concatenated directories and files by using Tar.exe for Win32, or you can use WinZip to separate and uncompress them. WinZip provides built-in support for TAR, gzip, UNIX compress, UUencode, BinHex, and Multipurpose Internet Mail Extensions (MIME). Then you can move the files into the appropriate Web server directory, which is c:\Inetpub\wwwroot by default. For applications, it's particularly important to retain the original directory structure.

To obtain a public domain copy of Tar.exe for Win32, go to http://www.acs.oakland.edu/ and use the search term "nttar.zip."

Ws_ftp, a tool that performs recursive FTP, is available at http://www.shareware.com/.

To download an evaluation version of WinZip, see http://www.winzip.com/.

Items to Note about Windows File Systems

Here are a few items you might want to note about Windows file systems if you've been accustomed to working in a UNIX environment.

- On a Windows File Allocation Table (FAT) file system, Write access to a file is equivalent to full access. To protect data shared on a FAT file system, share it as a read-only resource. NTFS provides more security options for protecting data and is generally recommended over FAT. For more information about NTFS security, see the Windows 2000 Server online product documentation.
- Because Windows FAT file systems were designed around single-user computers, they don't support the concept of a file owner or file group. However, NTFS allows you to specify a file owner, which can be either an individual or a group.
- Windows file systems do not support file links (symbolic links), but rather use a single-name/single-file concept with no support for multiple directory entries referring to the same file.

Converting UNIX File Names and Pathnames

When you migrate files and directories from a UNIX-based Web server, you need to edit file names and pathnames (called *paths* in Windows) so that they adhere to Windows conventions. Otherwise, hyperlink references to the files might malfunction, and in some cases files could be overwritten.

Conventions are slightly different between the Windows FAT and NTFS file systems. While for security reasons you will want to use NTFS as much as possible, there are cases where you might need to use FAT for cross-platform file sharing. The issues involved with both systems are described below.

- **File Name and Extension Length** UNIX supports long file names and four-character file name extensions. When you copy a UNIX file or directory to a Windows FAT or NTFS file system, Windows truncates a long file name or extension using the Microsoft® MS-DOS® 8.3 convention. It reduces the file name to its first six letters followed by a tilde (~) and a unique number, for a total of eight characters, not including the extension. It truncates the extension to its first two letters, followed by a tilde (~). For example, the long name *ProjectOverview.esti* would change to *Projec~1.es~*.

- **Case Sensitivity** UNIX file names are case sensitive. Windows FAT file names are *not* case sensitive, and Windows NTFS file names are "case aware." NTFS preserves case, but doesn't use it in some functions, such as indexing. In UNIX you can have two different files in the same directory with names distinguished *only* by case. For example, in any given directory you could have two different files, one named *MyFile* and the other named *myfile*. If you copy these files to a FAT directory, the Windows operating system preserves the letter cases of UNIX file names, but interprets the two names as being identical as long as they use the same characters in the same order, regardless of case. In this situation, the Windows operating system reads the first name correctly, and then reads the second as a duplicate name, and appends a number to it to make it unique. Using the previous example, this would result in the file name, *MyFile(2)*. Because this is a different file name, any hyperlinks or applications that reference the file will malfunction. Therefore, before you copy UNIX files to a FAT directory make sure they have unique Windows-style file names, and also remember to correct any hyperlink or application references to the file name.

- **Illegal Characters** The following characters are supported in UNIX-style file names, but are not permitted in Windows file or directory names:

  / \ : * ? " < > |

  The Windows operating system will replace each occurrence of one of these characters with the letter "X."

- **Directory Separators** UNIX pathnames use the forward slash (/) as a directory separator, and Windows paths use the backslash (\). Windows produces an error when it encounters UNIX-style pathnames.

- **Directory Hierarchy** The UNIX file system appears to be a single directory hierarchy whereas Windows storage is divided into one or more physical or virtual disk drives with a directory hierarchy on each. To access a file on Windows, you must know what disk drive the file is on and specify the drive letter (C:, D:, E:, and so forth) as part of the pathname for the file.

You also need to edit any file names and pathnames within UNIX application files in order to run them on Windows 2000 Server. This subject is discussed later, in "Migrating Web Applications."

Completing Web Site Migration

This section describes some additional steps you might need to take when migrating Web sites in order to restore hyperlinks, set up user Web sites, and replicate (or add) special features such as server-side includes, redirects, and directory indexes.

Repairing Hyperlinks

When you migrate Web pages, you need to find and repair any hyperlinks within them that were broken due to changes you might have made in the original Web site directory structure. You can use FrontPage features for testing and repairing hyperlinks, not only for internal URLs, but for those referring to external sites on the Internet as well.

> **Note** It is *not* recommended that you install the FrontPage client on the same computer as IIS 5.0. Instead, use FrontPage on a development computer to check and repair hyperlinks. Then publish the files to the computer running IIS 5.0, using the Web publishing features of FrontPage.

Setting up User Web Sites

IIS 5.0 provides the following options for implementing user Web sites, which are common in most Internet service provider (ISP) and volume hosting environments:

- **Host Headers** Implementing the HTTP 1.1 host header standard, you can create multiple Web sites on a single IIS server that share the same IP address. For browsers that do not support the HTTP 1.1 standard, IIS displays the home page for the Default Web Site and can be configured to send a cookie that automatically redirects users to the selected site on their next visit. For this reason, it is recommended that ISPs use the Default Web Site for their Web site, rather than for a customer Web site. This Web site can then display links to customer Web sites.
- **IP Addressing** You can create multiple Web sites by assigning each one a unique IP address.
- **Unique Port Numbers** You can create multiple Web sites by assigning each one a unique port number. For sites using something other than port 80, which is the default, users must append the port number to the URL to access the site. For example, http://www.microsoft.com/IIS:82 would access a Web site named IIS that uses port 82.

For more information about implementing these methods, see the "Web and FTP Sites" and "Name Resolution" topics in the IIS 5.0 online product documentation.

Implementing Special Web Features

IIS 5.0 supports many popular Web site features, such as directory browsing and indexing, document footers, and server-side includes. The following paragraphs describe some features you might want to implement on your Web sites.

- **Directory Browsing and Indexing** You can enable directory browsing and indexing on the **Home Directory** tab of **Web Site Properties**.
- **Document Footers** You can specify a document footer on the **Enable Document Footer** tab of **Web Site Properties**. For more information about document footers, see the "Adding a Footer to Web Pages" topic in the IIS 5.0 online product documentation.
- **Dynamically Updated Content** For content that must be frequently updated, you can generate dynamic Web pages by using ASP and HTML templates. IIS 5.0 builds pages on-the-fly from content that is dynamically extracted from a database. For more information about dynamic content, see "Data Access and Transactions" in this book.
- **Personalized Content** By using Dynamic HTML (DHTML) or the Browser Capabilities Component you can detect user browser capabilities to provide personalized content based on the user environment. For more information, see the "Client Capabilities" topic in the IIS 5.0 section of the SDK documentation on MSDN.
- **Redirection (HTTP redirect)** You can specify redirection from a Web site to another URL on the **Home Directory** tab of **Web Site Properties**.
- **Server-Side Includes** IIS 5.0 supports server-side includes. A Web page that has included information must have the .stm file name extension. The virtual directory containing the .stm files must have either script or execute permissions enabled.
- **Time-Sensitive Content** To direct the user's browser to expire cached content at a specific date and time, you can enable content expiration on the **HTTP Headers** tab of **Web Site Properties**.

Completing FTP Site Migration

Here are a few steps you can take to reproduce (or add) special FTP site features.

- **Creating User Directories** To have a user automatically placed in their own FTP directory when logging on, create a virtual FTP directory with the same name as the user.
- **Limiting Access** You can lock anonymous users into the FTP directory so they cannot browse outside it, while still enabling an authenticated client (who is not using FrontPage) to upload files to the same FTP directory.

  To limit access

  1. In Windows Explorer, place the FTP directory under the Www root directory.
  2. In the IIS snap-in, point the FTP server to the FTP directory.
  3. While still in the IIS snap-in, create a second FTP server under the first one and give it the same name as the user name of the client who wants to upload files.
  4. Point the second FTP server to the FTP directory (the same one as in step 2).
  5. In Windows Explorer set the following NTFS permissions on the FTP directory: give Anonymous FTP User Full Control on the FTP directory and deny all permissions on the root directory.

  After logging in, the authenticated client is placed in the virtual FTP site that has the same name. The client has full control over directory content and can upload files. An anonymous user who logs in will be able to read the files, but will have no control over them and will be unable to browse outside the virtual FTP directory.
- **Creating Welcome Messages** On the **Messages** tab of **FTP Site Properties**, you can create a welcome message that will be displayed to users when they enter the FTP site.

Replicating and Configuring Applications

Application files are a special case. Rather than copying them directly to the new IIS 5.0 server, application files should be transferred to a development computer for any necessary editing or rewriting, and then to a test computer for testing and debugging. When you're ready to deploy the application on the new IIS 5.0 server, follow the instructions given in the "Configuring Applications" topic of the IIS 5.0 online product documentation.

Later in this chapter, "Migrating Web Applications" describes your options for migrating a CGI application to IIS 5.0. You can leave it as a CGI and make relatively minor changes so it runs on IIS 5.0. Or you can rewrite it as an ISAPI extension or an ASP application. The pros and cons of each approach are discussed in terms of development costs and server resources.

Migrating Log Files

IIS 5.0 supports the new W3C Extended Logging Format, and you can migrate any compliant log file to IIS 5.0. IIS 5.0 logs are stored in ASCII (text) files. If you want to preserve logging information from the source server, you can copy ASCII data from your log files to the IIS 5.0 log files.

The primary difference between UNIX and Windows 2000 Server logging is that UNIX log files are generally stored and viewed in plaintext, but Windows 2000 Server provides a graphical user interface (GUI), called Event Viewer, for logging administration and viewing.

IIS 5.0 provides several features you can use to customize logged information, and you can either log to a text file or to an Open Database Connectivity (ODBC) Data Source Name (DSN) for dynamic evaluation. For more information about logging, see the "Logging Site Activity" topic in the IIS 5.0 online product documentation and "Administering an ISP Installation" in this book.

Migrating Configuration Settings

Web servers—whether they are IIS 5.0, Netscape, Apache, or some other type—perform many of the same basic functions, and therefore offer many similar configuration options. Web servers differ primarily in secondary features that address a special requirement, such as security. However, the way in which you configure a server and what settings are named can vary a great deal from one server to the next.

IIS 5.0 uses property sheets to provide a GUI for server configuration. You open property sheets from the IIS 5.0 MMC snap-in by right-clicking the item you want to configure, and then clicking **Properties**. For information about IIS 5.0 configuration options, see the "Server Administration" topic in the IIS 5.0 online product documentation.

You can set IIS 5.0 properties in the metabase for greater granularity at the computer, Web site, virtual directory, directory, and file level. For more information about the metabase, see the "Introduction to the IIS Metabase" topic in the IIS 5.0 online product documentation. For more information about automating IIS 5.0 administration tasks and administering from the command line, see "Administering an ISP Installation" in this book.

Here are some things to keep in mind when configuring IIS 5.0:

1. **When in doubt, right-click.** When you are running Windows, right-clicking a window or an object reveals a menu of useful operations. In the case of IIS 5.0, if you want to configure an object that appears in the IIS snap-in, such as the Administration Web Site, Default Web Site, or other Web site you have created, right-click the object, and then click **Properties**. Then at the top of the dialog box, click the tab that corresponds to the configuration options you want to set.

2. **If you get stuck, click the Help button.** Click the Help button that appears in most dialog boxes to open a topic specific to that dialog box. You can also open the in-depth IIS 5.0 online product documentation as follows: In the left pane of the IIS snap-in window, click the Default Web Site and make sure it's running. At the top of the IIS snap-in window, click **Help** or click the **Help** toolbar button, and then click **Help on Internet Information Services**. You can also open IIS Help from Windows 2000 Server Help, from within the "Internet Information Services" topic.

3. **Remember that the IIS snap-in isn't Windows Explorer.** Even though the IIS snap-in and Windows Explorer look very similar, you use them for different purposes. In the IIS snap-in, you configure Web site, FTP site, and Web administration properties. In the IIS snap-in you can also create *virtual* directories for a Web or FTP site, but only in Windows Explorer can you actually manage files and directories, performing operations such as the following:

   - Copying files and directories (or *folders*) to your Web and FTP site directories
   - Creating, moving, and deleting files and directories
   - Setting NTFS permissions for files and directories
   - Defining network file sharing for files and directories

4. **Configure IIS 5.0 to recognize both Windows and UNIX-style home page names.** To make migrating UNIX Web sites easier, specify the following five home page names in IIS Web Site Properties:

   - Index.htm
   - Index.html
   - Default.htm
   - Default.html
   - Default.asp

   This will cover nearly all cases for both UNIX and Windows Web sites. When you transfer a Web site from a UNIX-based computer, you won't need to change the home page file name for IIS 5.0 to serve it.

Later sections in this chapter—"Migrating from Apache HTTP Server," "Migrating from Netscape Enterprise Server," and "Upgrading or Replicating an IIS Web Server"—describe using the IIS Migration Wizard to replicate configuration settings from a server running Netscape Enterprise Server, Apache HTTP Server, IIS 4.0, and IIS 5.0 to a different server running IIS 5.0.

Securing the Server

Once you've migrated Web content and applications to IIS 5.0, you need to configure security. This section discusses the basics of securing your Web server: migrating user and group accounts, migrating certificates, setting NTFS file and directory permissions, and setting IIS 5.0 permissions. For more in-depth information, see "Security" in this book, as well as the security topics in the IIS 5.0 and Windows 2000 Server online product documentation. For information about using FrontPage security with IIS 5.0, see "Administering an ISP Installation" in this book. For additional details on security, see http://msdn.microsoft.com/workshop/security/default.asp and http://www.microsoft.com/security/default.asp.

Migrating Users and Groups

An important component of migration is transferring the identities and passwords of system users and groups to the new server. To make this easier, you can use Addusers.exe, a utility for adding user accounts to Windows 2000 Server, included on the Resource Kit companion CD.

If you're migrating from Netscape Enterprise Server 3.5, consider using the IIS Migration Wizard included on the Resource Kit companion CD to automate the migration of your local user database.

You also need to configure security on the user and group accounts, and you might want to create new group identities to enhance security. For information about doing this, see the Windows 2000 Server online product documentation.

Setting NTFS Permissions

To protect your Web server from unwanted intrusions, you can set access permissions by user and group account for each file or directory stored in the Windows NTFS file system. Permissions set on a directory apply to each file within the directory. However, if a file within the directory has more restrictive settings, the more restrictive settings apply. For information about setting NTFS permissions, see the "Setting NTFS Permissions for a Directory or File" topic in the IIS 5.0 online product documentation. Note that you cannot set these access permissions for files and directories stored in a FAT file system.

The following are a few guidelines for configuring NTFS permissions for user and group accounts used by IIS 5.0:

- **Anonymous User** During installation, the IIS 5.0 Web and FTP services are assigned a default user account, called IUSR_*computername,* for anonymous users. Do not make this account a member of any privileged group.
- **IIS Admin Service** Do not give the IIS Admin Service the right to log on as the LocalSystem account.
- **Test** It's a good idea to create a Test group account to which you can give enlarged access permissions, such as Write or Execute, needed by application developers and testers.
- **Full Control** Only two accounts should be always given NTFS Full Control permissions: LocalSystem and Administrator. On selected files, you might want to also give full control to Owner/Creator.

Setting IIS 5.0 Permissions

You set additional access permissions for Web users in the IIS snap-in. The easiest way to set up basic IIS 5.0 security is to use the Permissions Wizard. To start the wizard, in the IIS snap-in select the Web site or directory for which you want to set permissions, click **Action** on the toolbar, point to **All Tasks**, and then click **Permissions Wizard**.

You can also configure security on Web Site property sheets. The following are some rules-of-thumb for setting IIS 5.0 permissions based on the type of access you want to provide. You might need to implement security differently than described here, depending on the requirements of your particular system. For more information about setting access permissions, see the "Access Control" topic in the IIS 5.0 online product documentation and "Security" in this book.

> **Note** If NTFS file and directory access permissions do not match the access permissions set in the IIS snap-in, the more restrictive settings take effect.

- **Web and FTP Site Anonymous Access** To protect the information on your computer, restrict access by anonymous users. To do this, in the IIS snap-in deny all access to the anonymous user account, which is IUSR_*computername* by default. Next, select the specific files and directories under the root to which you want to allow anonymous access and enable the appropriate permissions for the anonymous user account. To override the access restrictions you set at the root level for these files and directories, clear the **Allow inheritable permissions from parent to propagate to this object** check box.
- **Web and FTP Site Authenticated Access** To require users to provide a valid user name and password in order to access a Web or FTP site, in the IIS snap-in disable anonymous access on the **Directory Security** tab of **Web Site Properties** or **FTP Site Properties**. For Web sites, you can then select the type of authentication: Basic, Digest, or integrated Windows authentication. For more information, see the "Authentication" topic in the IIS 5.0 online product documentation and "Security" in this book.

  By default IIS 5.0 attempts to authenticate Web and FTP users from the local user database. For a Web site, you can change authentication to the domain user database from within the IIS snap-in. For an FTP site, you must modify the **DefaultLogonDomain** metabase property for the FTP service. To do this, you can use the IIS Administration Script Utility (Adsutil.exe), installed with IIS 5.0, as follows:

  At the command prompt, type:

  adsutil.exe set msftpsvc/DefaultLogonDomain "*Name of Your Domain*"

  > **Note** To set up an FTP site where users can upload files, but not see files already uploaded to the site by other users, use virtual directories. Enable Write, but not Read, permission for the user accounts. Give Read permission to the Administrator account only.

Setting Permissions Based on Content

Table 3.1 provides guidelines for setting NTFS and IIS 5.0 security on a directory, based on its type of content.

Table 3.1 Basic Web Security Settings

Content	Directory Name/Type	NTFS Account	NTFS Directory Permissions	IIS 5.0 Virtual Directory Permissions
Static (.htm, .gif, .jpg, and so on.)	Content	Authenticated Users	Read	Allow Anonymous Access. Allow Read permissions. Directory Browsing okay.
ASP pages	ASP pages	Authenticated Users	Execute	Allow Anonymous Access. Allow Read permissions. For Execute Permissions, choose Scripts only. Directory Browsing okay.
ASP-page includes	Includes	Authenticated Users	Execute	Allow Read permissions.
Server-side includes	Content	Authenticated Users	Execute	Disable Anonymous Access. For Execute Permissions, choose Script or Execute permissions.
CGI scripts	Scripts	Authenticated Users	Execute	Disable Anonymous Access. For Execute Permissions, choose Scripts only. Disable Read, Write, and Directory browsing.
ISAPI server extensions	ISAPI Extensions	Authenticated Users	Execute	Disable Anonymous Access. For Execute Permissions, choose Execute. Disable Read, Write, and Directory browsing.
ISAPI filters	ISAPI Filters	Authenticated Users	Execute	Disable Anonymous Access. For Execute Permissions, select Execute. Disable Read, Write, and Directory browsing.
Executable CGI applications	CGI bin	Authenticated Users	Execute	Disable Anonymous Access. For Execute Permissions, choose Execute. Disable Read, Write, and Directory browsing.

Continued

Table 3.1 Basic Web Security Settings *(continued)*

Content	Directory Name/Type	NTFS Account	NTFS Directory Permissions	IIS 5.0 Virtual Directory Permissions
Databases	Databases	For remote databases, share out the directory and enable the Guest account for the IIS 5.0 Web service that accesses the share.	Security depends on the database. * See note that follows.	Security depends on the database.
Microsoft® Component Object Model (COM) and Microsoft® Distributed Component Object Model (DCOM) components	Components		** See note that follows.	Disable Anonymous Access. Enable Execute permissions only. Disable Read, Write, and Directory browsing.
Downloadable executable files	Downloads	Authenticated Users	Read	Enable Read permissions only. Disable Execute permissions or the file will execute rather than download.

Note *Whenever possible, keep Microsoft® Access databases on the same computer as IIS 5.0. There isn't a secure way for an IIS 5.0 application to connect to an Access database located on a networked drive.

**In general, you should place COM and DCOM components in a directory with Execute permissions only. Place COM and DCOM components that need to write to data files in the same directory with the data files and enable both Execute and Write permissions. Be aware that setting Write permissions on a components directory creates the potential for intruders to upload and run an executable file on your server.

To help prevent unauthorized access to a directory

1. In the IIS snap-in, disable Directory Browsing for that directory and its parent directory.
2. Set up auditing on the directory so you can monitor any suspicious activity. For more information about auditing, see "Security" in this book.

Migrating Security Certificates

The easiest approach to migrating security certificates is using the Web Server Certificate Wizard. You can start the wizard in the IIS 5.0 snap-in from the **Directory Security** tab of **Web Site Properties**. For information about using the wizard, see the "Using the New Security Task Wizards" topic in the IIS 5.0 online product documentation.

Another approach to migrating a certificate is saving it as a .cer file and copying it to the new Windows 2000 Server. You can install the certificate by double-clicking the .cer file and then use the Web Server Certificate Wizard to bind the certificate to the appropriate Web site. Remember to create a backup copy of server and client certificates and keys in case they become corrupted during the transfer.

Integrating UNIX and Windows 2000 Server Security

Windows 2000 Server and UNIX handle their respective user accounts very differently, complicating security implementation in a mixed environment. If you plan a multistep approach to migrating from UNIX to the more secure Windows 2000 Server environment, you might find it necessary to use a mixed authentication scheme during the early stages.

Windows 2000 Server implements the Kerberos v5 authentication protocol, a mature Internet security standard, as the default protocol for network authentication. This provides a foundation for authentication interoperability with other platforms, such as UNIX. For more information about the Kerberos v5 protocol, see "Security" in this book.

Migrating from Apache HTTP Server

So far this chapter has described migration in general terms. Each step is applicable to a migration from almost any type of Web server. This section provides details about migrating from Apache HTTP Server. It briefly compares some Apache and IIS 5.0 features and then provides tables that match Apache directives to their corresponding IIS 5.0 configuration properties. The tables indicate whether or not the IIS Migration Wizard, included on the Resource Kit companion CD, migrates a particular directive. They also describe how to configure each setting manually on IIS 5.0.

Comparing Apache and IIS 5.0

The following are some Apache features that have different names and implementations on IIS 5.0.

Administration Interface

IIS 5.0 properties, which correspond with Apache directives, are contained in the metabase. The most significant difference you'll find between administering Apache and IIS 5.0 is the fact that IIS 5.0 provides graphical tools, the IIS snap-in and the Internet Services Manager (HTML), for configuring the metabase, whereas Apache provides plaintext configuration files, usually httpd.conf, srm.conf, and access.conf, in which you configure directives.

Apache administrators who prefer to work from the command line and to script routine tasks will be happy to know that IIS 5.0 also provides a set of tools for programmatic administration. These tools are described in the "Administering IIS Programmatically" topic in the IIS 5.0 online product documentation. For more information about using the tools, see "Administering an ISP Installation" in this book.

Apache administrators will also find that Windows 2000 Server commands are similar to familiar UNIX commands. Additional utilities for command-line administration are included with the Windows NT Services for UNIX, a suite of interoperability tools and utilities provided by Microsoft for Windows NT Server and UNIX. At the time of this writing, a version of these tools is under development for Windows 2000.

Security

If you're accustomed to Apache security, be aware that Allow and Deny access permissions are processed in a different order by Windows 2000 Server, producing results you might not expect. Windows 2000 Server always honors a Deny. If you deny permission to a group, and then try to allow permission to an individual member of the group, the member still will not be able to access the resource.

Also note that, unlike in Apache, CGI-bin is contained within the IIS 5.0 Web space. This is because NTFS and IIS 5.0 security sufficiently protects it from attack.

User Directory

You set up individual user Web sites differently on IIS 5.0 than on Apache HTTP Server. With Apache, you add a user to the machine, and then create a <*~username*> directory for the user's Web pages. The server then responds to the following request by displaying the user's Web pages:

http://domain.com/<*~username*>

IIS 5.0 does not automatically create virtual directories for users. Instead, to add a user Web site you create a <*username*> directory for their files in Windows Explorer, create a virtual directory named <*~username*> in the IIS 5.0 snap-in, and then point it to the <*username*> directory.

Virtual Host

The IIS 5.0 counterpart to the Apache Virtual Host is a *virtual server*. As with Apache virtual hosts, each virtual server in IIS 5.0 has its own domain name and IP address. Apache also includes "name-based" virtual hosts in this category. IIS 5.0 supports this feature in the same manner as Apache, through host header names, but doesn't have a specific term for it.

Alias/Directory Alias

In IIS 5.0 an alias or directory alias is called a *virtual directory*. In Apache, you use the RedirectMatch/AliasMatch command to map a directory alias to a real directory located on the hard disk, as shown in the following example:

```
Alias /folder "c:/apache/htdocs/newfolder/"
```

To do the same thing in IIS 5.0, in the IIS snap-in, create a virtual directory and point it to the real directory in Windows Explorer. Using the previous example, the virtual directory would be named "folder," and you would point it to c:\apache\htdocs\newfolder\. Note the change from using forward slashes (/) in the UNIX pathname to backslashes (\) in the Windows path.

Custom Error Messages

In Apache, you provide custom error messages by editing the Error Document and referring to it by using the command:

```
ErrorDocument 404 http://domain.com/404.html
```

To customize error messages in IIS 5.0, in the IIS snap-in open **Properties** for the Web site. On the **Custom Errors** tab, you'll see the location of the error message files. From here, you can map custom error messages to a file or to a URL on the local server.

Redirects

In Apache you use the following .htaccess command to redirect a user to another file:

```
Redirect /oldfile.html http://domain.com/path/to/new/file
```

There are two ways to implement a redirection in IIS 5.0:

1. In the IIS snap-in open **Properties** for the Web site. On the **Home Directory** tab, select **A redirection to a URL**, and choose the appropriate options.
2. Or, put a Default.asp file, containing the following code, in the same directory as the old file:

   ```
   Response.Redirect /oldfile.html http://domain.com/path/to/new/file
   ```

Migrating Apache Directives

This section lists the core Apache HTTP Server 1.3 directives and indicates whether, and how, the IIS Migration Wizard migrates them to IIS metabase properties. This section also explains how to configure each property in the IIS snap-in. There is not a one-to-one correspondence between Apache and the IIS 5.0 configuration options: not all IIS 5.0 settings exist on Apache and vice versa. This section does not cover IIS 5.0 properties that have no counterpart in Apache. For in-depth information about IIS 5.0 configuration parameters and metabase properties, see the "Administrator's Reference" topic in the IIS 5.0 online product documentation.

Server Directives

Table 3.2 Apache httpd.conf Directives and Corresponding IIS 5.0 Properties

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
AccessConfig	No	Not applicable	The wizard uses these parameters for mapping to the access configuration information. However, in IIS 5.0 there is no separate access configuration file.
BindAddress	Yes	ServerBindings	For multihoming, IIS 5.0 allows you to specify Virtual Hosts, or *virtual servers*. To configure a virtual server, right-click a Web site, choose **Properties**, and then select the **Web Site** tab. Click the **Advanced** button on the **Web Site** tab, and add the correct IP address and Transmission Control Protocol (TCP) port.
CacheNegotiatedDocs	No	Not applicable	You can specify an expiration date for content in a browser or proxy cache. To configure this setting, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. Select the **Enable Content Expiration** check box and enter expiration dates.
ErrorLog	No	Not applicable	All errors for the Inetinfo process are logged to the Windows® Event Log and do not need to be specified in the Web server configuration. The wizard uses IIS 5.0 default log settings.

Continued

Table 3.2 Apache httpd.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
ExpiresActive	Yes	HttpExpires	In IIS 5.0 content expiration is disabled by default. To enable content expiration, right-click a Web site, choose **Properties**, select the **HTTP Headers** tab, and then check the **Enable Content Expiration** check box.
ExpiresDefault	Yes	HttpExpires	In IIS 5.0 content expiration is disabled by default. To enable content expiration, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. Select the **Enable Content Expiration** check box, and then set the parameters.
Header	Yes	HttpCustomHeaders	To create a custom header, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. In the **Custom HTTP Headers** box, click **Add**, and then type a name and a value for the header.
HostnameLookups	Yes	EnableReverseDNS	IIS 5.0 log files capture the IP addresses of Web site visitors. To look up the host name of a given IP address, enable the metabase property EnableReverseDns. To set IP address restrictions, right-click a Web site, click **Properties**, click the **Directory Security** tab, and then click the **Edit** button in the **IP Address and Domain Name Restrictions** box.
IdentityCheck	Yes	LogExtFileUserName	To log the identity of each Web site visitor, right-click a Web site, click **Properties**, and then click the **Web Site** tab. Select the **Enable Logging** check box, and then click **Properties**. Click the **Extended Properties** tab, and then select the **User Name** check box.
<IfDefine>	Yes	Not applicable	The wizard uses this information when parsing the Apache files.
Include	Yes	Not applicable	This directive is not needed in IIS 5.0.

Continued

Table 3.2 Apache httpd.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
KeepAlive	Yes	AllowKeepAlive, MaxConnections	In IIS 5.0, HTTP Keep-Alives are enabled by default. To disable Keep-Alives, right-click a Web site, and choose **Properties**. On the **Web Site** property sheet, clear the **HTTP Keep-Alives Enabled** check box. You set the maximum number of connections and the connection time-out in this location as well.
KeepAliveTimeout	Yes	Connection Timeout	To set the Keep-Alive time-out, right-click a Web site and then choose **Properties**. In the **Connections** box on the **Web Site** property sheet, select the **Limited To** radio button and, in the **Connection Timeout** box, specify the number of seconds you want before an idle connection times out.
Listen	Yes	ServerBindings	IIS 5.0 allows you to specify a port for name-based virtual hosts. To configure this setting, right-click a Web site and then choose **Properties**. On the **Web Site** property sheet, click the **Advanced** button, and then enter the TCP port number.
ListenBacklog	Yes	ServerListenBacklog	This is a service-level property, and it cannot be configured from MMC.
MaxClients	Yes	MaxConnections	To configure this property, right-click a Web site, choose **Properties**, and then select the **Web Site** tab. Select the **Unlimited** or the **Limited To** radio button. For limited connections, in the **Connection Timeout** box specify the number of seconds before a connection should time out.
MaxKeepAliveRequests	No	Not applicable	There is no equivalent in IIS 5.0.
MaxRequestsPerChild	No	Not applicable	IIS 5.0 uses a limited number of processes, and there is no need to set the maximum number of requests per child process as there is with Apache.

Continued

Table 3.2 Apache httpd.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
Min/MaxSpareServers	No	Not applicable	IIS 5.0 uses a limited number of processes, and there is no need to account for this number.
NameVirtualHost	Yes	ServerBindings	An Apache virtual host corresponds to an IIS 5.0 Web site. In IIS 5.0, you can create virtual hosts either by using multiple IP addresses or by using a single IP address and HTTP 1.1 Host Header Names. To create a virtual host with a unique IP address, right-click a Web site, choose **Properties**, and then select the **Web Site** tab. Click the **Advanced** button and specify the IP address. To specify a host header name for a name-based virtual host, right-click a Web site, and then choose **Properties**. On the **Web Site** property sheet, click the **Advanced** button and enter the host header name for the IP address you want to use.
PidFile	No	Not applicable	IIS 5.0 does not offer the option to log its process ID to a file.
Port	Yes	ServerBindings	To set the port to which your Web server should listen, right-click a Web site, choose **Properties**, and then select the **Web Site** tab. Enter a port number in the **TCP Port** box.
Proxy Cache Parameters	No	Not applicable	IIS 5.0 cannot function as a proxy server on its own. Microsoft® Proxy Server is recommended as a proxy server for use with Windows 2000 Server.
ProxyRequests	No	Not applicable	See the previous note for Proxy Cache Parameters.
RlimitCPU	Yes	CpuLimit	IIS 5.0 has a number of properties that specify CPU throttling parameters. To specify performance parameters in the IIS snap-in, right-click a Web site, choose **Properties**, click the **Performance** tab, and then choose the settings you want.

Continued

Table 3.2 Apache httpd.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
ScoreBoardFile	No	Not applicable	IIS 5.0 does not have a Scoreboard file.
ServerAdmin	No	Not applicable	IIS 5.0 does not have a configuration setting for displaying the administrator's name and contact information. You can add this information to your pages by using ASP.
ServerAlias	Yes	ServerBindings	IIS 5.0 allows you to specify a host header name for a name-based virtual host. To configure this setting, right-click a Web site, and then choose **Properties**. On the **Web Site** property sheet, click the **Advanced** button and enter the host header name for the IP address you want to use.
ServerName	No	Not applicable	The host name for your Web server is stored in your Domain Name System (DNS) server and is not required in IIS 5.0 configuration properties. However, you must specify an IP address and HTTP port in order for IIS 5.0 to serve content.
ServerPath	Yes	Path	This directive is migrated to the **Path** property of the IIS **Virtual Directory** object. This property defines the path from a virtual directory to its corresponding physical directory. You configure this property in the IIS snap-in when you create a new virtual directory, by specifying from which directory the content is to be served.
ServerRoot	Yes	Not applicable	The wizard uses this information for parsing Apache files, but IIS 5.0 does not have the same concept of server root and does not have a corresponding property.
ServerType	No	Not applicable	IIS 5.0 always runs in stand-alone mode. Once IIS 5.0 is started, its process remains in memory and listens to the specified HTTP port. You can't configure IIS 5.0 to dynamically load as with an inetd server on Apache.

Continued

Table 3.2 Apache httpd.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
StartServers	No	Not applicable	See the previous note for Min/MaxSpareServers.
Timeout	Yes	ConnectionTimeout	You can specify the maximum amount of idle time to elapse before your server drops a connection. To configure this setting, right-click a Web site, choose **Properties**, and then select the **Web Site** tab. Enter the maximum timeout value in the **Connection Timeout** box.
TransferLog	No	Not applicable	IIS 5.0 does not use a transfer log.
User/Group	No	Not applicable	When IIS 5.0 is installed, by default it creates the IWAM user account under which the server runs. You must be logged on as an administrator or operator in order to start and stop the IIS 5.0 service, but the process does not retain your permissions.
<VirtualHost>	Yes	Not applicable	For each Apache virtual host, the wizard creates a new IIS Web site. It migrates the directives contained between the <VirtualHost> tags, including server bindings, and applies them to the Web site. You might need to correct the IP address of the new Web site following migration.

Resource Configuration

Table 3.3 Apache srm.conf Directives and Corresponding IIS 5.0 Properties

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
AccessFileName	Yes	Not applicable	The wizard uses these parameters for mapping access configuration information. However, in IIS 5.0 there is no separate access configuration file. IIS 5.0 security is integrated with Windows 2000 security. To limit access to a site or directory by user, you must configure a new user account in the Windows 2000 Server User Manager. You can also classify individuals or groups as "Web site Operators" with limited authority to administer a Web site. They do not have to be Windows 2000 Administrators. To define Web site Operators, in the IIS snap-in right-click a Web site, click **Properties**, and then click the **Operators** tab.
AddDescription	No	Not applicable	There is no corresponding property in IIS 5.0.
AddEncoding	Yes	MimeMap	To map a file extension to a MIME type, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. In the **Mime Map** box, click **File Types**, and then click **New Type**. Or, to edit an existing MIME type, select a file type in the list, and then click **Edit**. Type the file extension and associated MIME type in the appropriate boxes.
AddIcon	No	Not applicable	IIS 5.0 uses the standard Windows 2000 icons when displaying a directory. You cannot specify a substitute icon.
AddLanguage	No	Not applicable	There is no corresponding property in IIS 5.0.
AddType	Yes	MimeMap	To add MIME types, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. In the **Mime Map** box, click **File Types**, and then click **New Type**. Type the file extension and the associated MIME type in the appropriate boxes.

Continued

Table 3.3 Apache srm.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
Alias	Yes	Not applicable	The wizard creates a **Virtual Directory** object for each Apache alias and applies the appropriate parameters by configuring corresponding metabase properties. To create a virtual directory, right-click the FTP or Web site, click **New**, and select **Virtual Directory**. Use the New Virtual Directory Wizard to complete this task.
AliasMatch	Yes	Not applicable	There is no direct equivalent in IIS 5.0 because there is no concept of regular expressions. The wizard creates a corresponding **Virtual Directory** object. See the previous note for Alias.
DefaultIcon	No	Not applicable	Windows 2000 Server offers a standard default icon for file types that do not have a preset icon in the file system.
DefaultType	Yes	MimeMap	IIS 5.0 contains a comprehensive list of MIME types. You can add new MIME types to the list should you need to serve a new MIME type. To view default MIME types, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. Click the **File Types** button in the **MIME Map** section of the tab.
DirectoryIndex	Yes	EnableDirBrowsing	You can configure IIS 5.0 to allow directory browsing. To configure this setting, right-click a Web site, choose **Properties**, select the **Home Directory** tab, and then select the **Directory Browsing** check box. IIS 5.0 does not allow you to specify a prewritten HTML document as a directory index.
DocumentRoot	Yes	Path	The wizard migrates this directive to the **Path** property of the IIS **Root** object. This property defines the path from a Web site home directory to its corresponding physical directory. To configure this property, right-click a Web site, choose **Properties**, and select the **Home Directory** tab. Then specify the location of the home directory (document root).

Continued

Table 3.3 Apache srm.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
ErrorDocument	Yes	HttpErrors	To enable custom error messages, right-click a Web site, choose **Properties**, and then select the **Custom Errors** tab. In cases where the custom error page is a standard HTML page, you need only copy the file to the IIS 5.0 system in order to complete the migration. In the case of CGI custom errors, you need to test the CGI scripts after moving them to IIS 5.0.
FancyIndexing	No	Not applicable	IIS 5.0 offers default indexing only.
HeaderName	No	Not applicable	There is no corresponding property in IIS 5.0.
IndexIgnore	No	Not applicable	There is no corresponding property in IIS 5.0.
LanguagePriority	No	Not applicable	There is no corresponding property in IIS 5.0.
MetaDir	No	Not applicable	You do not need to specify a Meta Directory to serve HTTP header information. To specify custom HTTP headers, right-click a Web site, choose **Properties**, select the **HTTP Headers** tab, and then click **Add**. Specify a header name and value in the appropriate boxes.
MetaSuffux	No	Not applicable	See the previous note for MetaDir.
ReadmeName	No	Not applicable	IIS 5.0 does not specify a default name for ReadMe files.
Redirect	Yes	HttpRedirect	To redirect a request to another resource, right-click a Web site, choose **Properties**, select the **Home Directory** tab, and then select **A redirection to a URL**. Type the URL in the **Redirect to** box.
RedirectTemp	Yes	HttpRedirect	In IIS 5.0, redirections are temporary by default.
RedirectPermanent	Yes	HttpRedirect	To make a redirection permanent, follow the steps previously given for Redirect. In addition, select the **A permanent redirection for this resource** check box after typing the URL.

Continued

Table 3.3 Apache srm.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
ResourceConfig	Yes	Not applicable	The wizard uses this information, but it does not directly translate to an IIS 5.0 property.
ScriptAlias	Yes	Not applicable	The wizard creates a **Virtual Directory** object using the Apache path information and sets the IIS **AccessExecute** property to TRUE. Any virtual directory can execute scripts when the "Allow Scripts" permission is enabled in the IIS snap-in. To configure this property, right-click a Web site, choose **Properties**, select the **Home Directory** tab, and then select the **Scripts only** or the **Scripts and Executables** option in the **Execute Permissions** box.
TypesConfig	Yes	MimeMap	The wizard adds specified MIME types to the IIS 5.0 MIME map. To view or configure MIME types, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. Click the **File Types** button in the **MIME Map** section of the tab.
UserDir	No	Not applicable	IIS 5.0 does not offer a default directory for ISP user httpd directories. You must create a virtual directory for each user in the IIS snap-in, and then point it to the user directory in Windows Explorer.

Access Configuration

Table 3.4 Apache access.conf Directives and Corresponding IIS 5.0 Properties

Apache Directive	Wizard Migrates (Y/N)	IIS 5.0 Metabase Property	IIS Snap-in Configuration
AllowOverride	Yes	Not applicable	The wizard uses this information to parse the access configuration files. However, IIS 5.0 utilizes Windows 2000 security in order to restrict access to a site, so htaccess files are not necessary to control access. Note that in IIS 5.0 you can classify individuals or groups as "Web site Operators" with limited authority to administer a Web site. They do not have to be Windows 2000 Administrators. To define Web site Operators, in the IIS snap-in right-click a Web site, click **Properties**, and then click the **Operators** tab.
AuthDBGroupFile	No	Not applicable	The wizard does not directly migrate this information, but uses it to create the Users file for use with Addusers.exe. Following migration, you must reconfigure all security on IIS 5.0.
AuthDBUserFile	No	Not applicable	See previous note for AuthDBGroupFile.
AuthDBMGroupFile	No	Not applicable	See previous note for AuthDBGroupFile.
AuthDBMUserFile	No	Not applicable	See previous note for AuthDBGroupFile.
AuthName	No	Realm	See previous note for AuthDBGroupFile.
AuthType	No	AuthBasic	In Apache, AuthType is usually set to Basic. The corresponding IIS 5.0 metabase property is **AuthBasic**. To configure authentication, right-click the virtual directory for which you want to set authentication, click **Properties**, and then click **Directory Security**. In the **Enable anonymous access and edit the authentication methods for this resource** box, click **Edit**, and then choose an authentication method.
<Directory>	Yes	Not applicable	The wizard migrates defined directives enclosed in this tag to the corresponding IIS 5.0 virtual directory.

Continued

Table 3.4 Apache access.conf Directives and Corresponding IIS 5.0 Properties *(continued)*

Apache Directive	Wizard Migrates (Y/N)	IIS 5.0 Metabase Property	IIS Snap-in Configuration
<DirectoryMatch>	Yes	Not applicable	The wizard migrates defined directives enclosed in this tag to the corresponding IIS 5.0 virtual directory.
<Files>	Yes	Not applicable	The wizard migrates defined directives enclosed in this tag to the corresponding IIS 5.0 virtual directory.
<FilesMatch>	Yes	Not applicable	The wizard migrates defined directives enclosed in this tag to the corresponding IIS 5.0 virtual directory.
<Limit>	No	Not applicable	There is no equivalent in IIS 5.0.
<LimitExcept>	No	Not applicable	There is no equivalent in IIS 5.0.
<Location>	Yes	Not applicable	The wizard migrates defined directives enclosed in this tag to the corresponding IIS 5.0 virtual directory.
<LocationMatch>	Yes	Not applicable	The wizard migrates defined directives enclosed in this tag to the corresponding IIS 5.0 virtual directory.
Options, ExecCGI	Yes	AccessExecute	You can set most of the IIS 5.0 equivalents of the *Options* parameter by enabling execution or script permissions for any virtual directory. To do this, in the IIS snap-in right-click the directory for which you want to set permissions, and then click **Properties**. Click the **Home Directory** tab, and then set permissions in the **Execute Permissions** box.
Options Indexes	Yes	EnableDirBrowsing	To enable users to view directory contents, in the IIS snap-in right-click the directory for which you want to enable browsing, and then click **Properties**. Click the **Home Directory** tab, and then select the **Directory browsing** check box.
Server status reports	No	Not applicable	IIS 5.0 does not provide server status reports.

Migrating Custom Modules

In Apache, custom modules extend the capabilities of the Web server. With IIS 5.0, there are a number of options for extending server capabilities. There is no direct way to migrate a custom module, so it must be recreated in IIS 5.0 using one of the following approaches:

- **ASP with COM components or ISAPI DLLs** can be written to duplicate most of the desired functionality, such as database or file system access. There are a number of built-in ASP objects to speed this task, such as **FileSystemObject**, which provides the methods, properties, and collections you use to access the file system.
- **ISAPI filters** allow you to implement custom authentication and logging, as well as URL rewriting, or "munging." For more information about using ISAPI filters to extend IIS 5.0 security, see "Security" in this book.

For more information about using these technologies, see the IIS 5.0 section of the SDK documentation on MSDN.

Migrating from Netscape Enterprise Server

This section provides additional details about migrating from Netscape Enterprise Server (NES). It briefly compares NES with IIS 5.0 and provides tables that match NES configuration settings to their corresponding IIS 5.0 metabase properties. The tables indicate whether or not the IIS Migration Wizard, included on the Resource Kit companion CD, migrates a particular setting. They also describe how to configure each setting manually.

Comparing NES and IIS 5.0

Terminology

Here are some essential terms used in NES with explanations of their equivalents in IIS 5.0.

- **Hardware Virtual Server** In NES, a "hardware virtual server" is a site with a separate IP address; the term implies a number of Web sites on a single computer, each with a separate IP address. The counterpart in IIS 5.0 is a virtual server. As with NES hardware virtual servers, each virtual server in IIS 5.0 has its own domain name and IP address.

- **Software Virtual Server** In NES, a "software virtual server" is a site that may share an IP address with one or more other sites. In IIS 5.0, you can assign any number of sites to a single IP address and distinguish them by using host headers, but no special term is employed to describe them.
- **Multiple Instances of the Server** In NES, you host multiple Web sites on one computer by using multiple instances of the server if:
  - The operating system does not have strong thread support.
  - The operating system does not allow a single process to schedule threads on more than one processor.
  - Multiple instances of the server provide full process isolation, protecting a Web site from failure should another site on the same system crash.

    It isn't appropriate to run multiple instances of IIS 5.0, because IIS 5.0 running on Windows 2000 Server offers thread support across multiple processors, full configuration of each site hosted by the server, and process isolation for applications.
- **Server Manager** Server Manager is the NES administration tool equivalent to the IIS snap-in for MMC.
- **Directory Aliases** In NES, you can map an alias such as this:

  /admin-offices/student

  to a real directory such as this:

  /admin-offices/studaffairs/cpc

  In IIS 5.0 a virtual directory corresponds to an alias. You can use the New Virtual Directory Wizard to create a virtual directory. To start the wizard, in the IIS snap-in select the Web site for which you want to define a virtual directory, click the **Action** button, point to **New**, and then select **Virtual Directory**. For more information, see the "Creating Virtual Directories" topic in the IIS 5.0 online product documentation.

Migrating NES Configuration Settings

The IIS Migration Wizard, included on the Resource Kit companion CD, migrates NES Web server settings, the virtual directory structure, and user accounts.

The tables in this section list NES 3.5 settings and indicate whether or not the IIS Migration Wizard migrates the setting and where the setting is found in IIS 5.0. Each heading within this section corresponds to a tab within the NES Server Manager.

Note that IIS 5.0 administrative settings, or properties, can be set on the server, site, directory, or even file level. Most NES settings apply to the site level only.

Server Preferences

Table 3.5 NES 3.5 Server Preferences and Corresponding IIS 5.0 Properties

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
Bind To Address	Yes	ServerBindings	Right-click a Web site, click **Properties**, and then click the **Web Site** tab. The setting appears in the **IP Address** box.
Convert 2.0 ACL file	No	Not applicable	There are no settings to migrate.
Dynamic Configuration Files	No	Not applicable	In IIS 5.0 you can classify individuals or groups as "Web site Operators" with limited authority to administer a Web site. They do not have to be Windows 2000 Administrators. To define Web site Operators, in the IIS snap-in right-click a Web site, click **Properties**, and then click the **Operators** tab.
Enable DNS	No	Not applicable	In IIS 5.0 you can restrict access by domain name. However, this feature can have a significant negative effect on server performance.
Encryption	No	Not applicable	You can install a server certificate and enable encryption by using the Security Task Wizards.
Error Responses	Yes	HttpErrors	When migrating a custom error page that is a standard HTML page, you need only copy the file to the IIS 5.0 system to complete the migration. In the case of CGI custom errors, you need to test the CGI scripts after moving them to IIS 5.0. To enable custom error messages in the IIS 5.0 snap-in, right-click a Web site, choose **Properties**, and then select the **Custom Errors** tab.
HTTP Persistent Connection Timeout	Yes	ConnectionTimeout	Right-click a Web site, click **Properties**, and then click the **Web Site** tab. The setting appears in the **Connection Timeout** box.

Continued

Table 3.5 NES 3.5 Server Preferences and Corresponding IIS 5.0 Properties *(continued)*

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
Maximum Simultaneous Requests	Yes	MaxConnections	Right-click a Web site, click **Properties**, and then click the **Web Site** tab. The setting appears in the **Limited To** box.
MIME Types	Yes	MimeMap	Right-click a Web site, click **Properties**, click the **HTTP Headers** tab, and then select the **File Types** button.
MTA Host and NNTP Host	No	Not applicable	Windows 2000 Server includes Simple Mail Transfer Protocol (SMTP) and Network News Transfer Protocol (NNTP) services. For more information, see the Windows 2000 Server online product documentation.
On/Off	No	Not applicable	Select a Web site you want to stop or start, and then click the **Stop** or **Start** toolbar button.
Restore Configuration	No	Not applicable	IIS 5.0 supports configuration backup. To back up the IIS 5.0 metabase, in the IIS snap-in right-click the computer name, choose **Backup/Restore Settings**, click **Backup**, type a name for the backup file, and then click **OK**.
Restrict Access	No	Not applicable	IP address restrictions are not migrated because they are defined differently in IIS 5.0. To set IP address restrictions, right-click a Web site, click **Properties**, click the **Directory Security** tab, and then click the **Edit** button in the **IP Address and Domain Name Restrictions** box.
Server Name	Yes	ServerBindings	The setting is migrated to a host header name. For information about this feature, see the "Naming Web Sites" topic in the IIS 5.0 online product documentation.
Server Port	Yes	ServerBindings	Right-click a Web site, click **Properties**, and then click the **Web Site** tab. The setting appears in the **TCP Port** box.

Applications

NES CGI applications can be ported directly to run on IIS 5.0, or they can be converted to ISAPI or ASP. For more information, see "Migrating Web Applications" later in this chapter.

Table 3.6 NES 3.5 Application Settings and Corresponding IIS Properties

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS Metabase Property	IIS Snap-in Configuration
CGI Directory	Yes	Not applicable	The wizard creates a CGI virtual directory and enables the CGI Execute permission.
CGI File Type	No	Not applicable	There is no corresponding property in IIS 5.0.
Java	No	Not applicable	The Java virtual machine is already enabled on IIS 5.0, so there is no need to migrate this setting.
Query Handler	No	Not applicable	There is no corresponding property in IIS 5.0.
Server Side JavaScript	No	Not applicable	IIS 5.0 includes server-side support for JScript and VBScript. There is no need to migrate a switch setting for these languages.
ShellCGI Directory	Yes	Not applicable	The wizard creates a ShellCGI virtual directory and enables the CGI Execute permission.
URL Prefix	Yes	Not applicable	The wizard creates a corresponding virtual directory.
WAI Management	No	Not applicable	This setting has to do with the Internet Inter-ORB Protocol (IIOP); IIS 5.0 uses the COM and DCOM object models.
WinCGI Directory	No	Not applicable	WIN CGI is not supported in IIS 5.0.

Server Status

Table 3.7 NES 3.5 Server Logging and Corresponding IIS 5.0 Properties

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS 5.0 Metabase Property	IIS Snap-in Configuration
Archive Log	No	Not applicable	There are no settings to migrate. When you set your logging preferences in IIS 5.0, you can use Microsoft® Windows® 2000 Backup or other third-party backup tools to archive the log files and remove them from the server as appropriate.

Continued

Table 3.7 NES 3.5 Server Logging and Corresponding IIS 5.0 Properties *(continued)*

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS 5.0 Metabase Property	IIS Snap-in Configuration
Generate Report	No	Not applicable	There are no settings to migrate. You can customize and extend IIS 5.0 logging in the IIS snap-in. You can set viewing options and filters in the Windows 2000 Server Event Viewer.
Log Preferences	Yes	Not applicable	Basic log file settings are migrated, but due to differences in logging methods you should review the settings created in IIS 5.0 to make sure they are optimal for your new environment.
Log Client Accesses	Yes	LogType	See previous note for Log Preferences.
Record Domain Names/IP Addresses	Yes	LogExtFileClientIp, LogExtFileComputerName	See previous note for Log Preferences.
Format	No	Not applicable	There is no corresponding property in IIS 5.0.
Monitor Current Activity	No	Not applicable	There are no settings to migrate. To monitor server activity on IIS 5.0, use the Windows 2000 Server System Monitor to evaluate performance and resource consumption.
Simple Management Network Protocol (SMNP) Sub-Agent Configuration	No	Not applicable	There is no corresponding property in IIS 5.0.
Rotate Log	Yes	LogFilePeriod	See previous note for Log Preferences.
View Access Log	No	Not applicable	There are no settings to migrate. You can view access logs from the Windows 2000 Server Event Viewer.
View Error Log	No	Not applicable	There are no settings to migrate. You can view error logs from the Event Viewer.

Configuration Styles

The settings under this heading are not migrated to IIS 5.0. IIS 5.0 includes support for property inheritance, which achieves much the same result as configuration styles. In the IIS snap-in, you can right-click the server and set global properties for the WWW Service. Every new Web site created on the server inherits these properties. Similarly, when you set properties for a Web site, directories created for the site inherit site properties.

Content Management

Table 3.8 NES 3.5 Content Management and Corresponding IIS 5.0 Properties

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS 5.0 Metabase Property	IIS Snap-in Configuration
Document Footer (Footer Text)	Yes	DocFooter	You can specify a document footer for the entire IIS 5.0 server, for a single Web site, or for a directory. To configure this setting, right-click the server, a Web site, or a directory; click **Properties,** and then click the **Documents** tab.
Additional Document Directories (URL Prefix, Map to Directory)	Yes	Not applicable	The wizard creates a corresponding virtual directory. You can use the New Virtual Directory Wizard to create a virtual directory. To start the wizard, select the Web site for which you want to define a virtual directory, click the **Action** button, point to **New**, and then select **Virtual Directory**.
Cache Control Directives	No	Not applicable	By default, HTML pages in IIS 5.0 are cached by proxy servers. The default value for ASP pages is "private," meaning they cannot be cached. You can use the **Response** object to control whether a proxy server caches the page.
Default MIME Type	No	Not applicable	IIS 5.0 contains a comprehensive list of MIME types. Should you need to serve new MIME types, you can add them to the list. To view or edit MIME types, right-click a Web site, choose **Properties**, and then select the **HTTP Headers** tab. Click the **File Types** button in the **MIME Map** section of the tab.

Continued

Table 3.8 NES 3.5 Content Management and Corresponding IIS 5.0 Properties *(continued)*

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS 5.0 Metabase Property	IIS Snap-in Configuration
Directory Indexing	Yes	EnableDirBrowsing	If you have selected "Simple" or "Fancy" directory indexing, the IIS Migration Wizard sets the **Directory Browsing Allowed** setting. To configure this setting, right-click a Web site, click **Properties**, and then click the **Home Directory** tab.
Hardware Virtual Servers (IP Address, Document Root)	Yes	ServerBindings	This is migrated as a component of **ServerBindings**. To configure a virtual server, right-click a Web site, choose **Properties**, and then select the **Web Site** tab. Click the **Advanced** button, and add the IP address and TCP port.
Home Page/Index File	Yes	DefaultDoc	A home page is migrated to IIS 5.0 using the default document name, and is listed ahead of any documents using index file names (according to the Index Filenames setting in this table). If no home page is listed, any specified index file names are used for the default document.
Index Filenames	Yes	DefaultDoc	See the previous note for the Home Page/Index File.
International Characters	No	Not applicable	With ASP pages, you specify the character set by using the **Response.Charset** property.
Parse Accept Language Header	No	Not applicable	Microsoft® Indexing Service can interpret this header, in order to determine the language in which a query is being written.
Parse HTML	No	Not applicable	By default IIS 5.0 processes files with .stm, .shtm, or .shtml file name extensions. For information about enabling and using server-side includes, see "About Server-Side Includes" in the IIS 5.0 online product documentation.

Continued

Table 3.8 NES 3.5 Content Management and Corresponding IIS 5.0 Properties *(continued)*

NES 3.5 Configuration Setting	Migrated by Wizard? (Y/N)	IIS 5.0 Metabase Property	IIS Snap-in Configuration
Primary Document Directory	Yes	Not applicable	The wizard creates a corresponding virtual directory. You can use the New Virtual Directory Wizard to create a virtual directory. To start the wizard, select the Web site for which you want to define a virtual directory, click the **Action** button, point to **New**, and then select **Virtual Directory**.
Software Virtual Servers (URL Host, Home Page)	Yes	ServerBindings	In IIS 5.0, you can assign any number of sites to a single IP address and distinguish them by using host headers, but no special term is employed to describe them. To configure a virtual server by using host headers, right-click a Web site, and then choose **Properties**. On the **Web Site** property sheet, click the **Advanced** button and enter the host header name for the IP address you want to use.
URL Prefix, Forward Requests To	Yes	HttpRedirect	This is called redirection in IIS 5.0. To redirect a request to another resource, right-click a Web site, choose **Properties**, select the **Home Directory** tab, and then select **A redirection to a URL**. Type the URL in the **Redirect to** box.

Web Publishing

The wizard does not migrate Web publishing settings. Web publishing can be supported with client-side development and management tools such as FrontPage (a member of the Microsoft® Office family) and Visual InterDev.

Agents and Search

The wizard does not migrate Agents and Search settings. Indexing Service, included with Windows 2000 Server, provides index and search capabilities. For more information, see the Windows 2000 Server online product documentation.

Auto Catalog

The wizard does not migrate Auto Catalog settings. As with Agents and Search settings, IIS 5.0 uses Indexing Service to build a searchable catalog of information about the content of the Web site.

Upgrading or Replicating an IIS Web Server

In addition to migrating to IIS 5.0 from another Web server, you might want to upgrade from an older version of IIS, or you might want to replicate an IIS 5.0 Web server to another computer. This section provides some tips for upgrading and replicating an IIS Web server.

First, here are a few items to note about upgrading IIS:

- You can no longer use IS2WCGI to run WinCGI applications on IIS, starting with IIS 4.0.
- For important information about what's changed between IIS 4.0 and IIS 5.0, see the "What's Changed" topic in the IIS 5.0 online product documentation.
- For information about changes to ASP in IIS 5.0, see the "Important Changes in ASP" topic in the IIS 5.0 online product documentation.

Choosing an Approach

There are several ways you can approach upgrading an IIS Web server, as summarized in the following paragraphs.

1. **Upgrade to a clean installation.** Perform a clean installation of Windows 2000 Server and IIS 5.0 on new hardware, and then migrate settings, content, and applications to the new server by using the IIS Migration Wizard included on the Resource Kit companion CD. Thoroughly test and debug the migrated server prior to deploying it and taking the original production Web server offline.

   Pros By doing this, you avoid taking your production Web server offline for a potentially extended time period while you upgrade and test it. Following deployment, if problems arise with the new server that didn't appear during testing, you have the original server available as a backup.

   Cons The IIS Migration Wizard will not migrate large numbers of Web sites.

   Recommendation This is the recommended method if you want to upgrade your hardware at the same time you upgrade IIS.

2. **Upgrade a duplicate of the IIS server.** On a second computer, install the same version of Windows NT Server and IIS as exists on the current production Web server and replicate content and applications. Upgrade the new server to Windows 2000 Server with IIS 5.0 by using the Windows 2000 Server Setup Wizard. Install IIS 5.0 during setup. Thoroughly test and debug the migrated server prior to deploying it and taking the original production Web server offline.

   Pros: By doing this, you avoid taking your production Web server offline for a potentially extended time period. Following deployment, if problems arise with the new server that didn't appear during testing, you have the original server available as a backup.

   Cons: You could potentially run into problems with the upgrade if the new server hardware is quite different from the old one. You'll probably need to adjust performance settings that are affected by hardware.

   Recommendation: This is a good approach if you want to upgrade your existing hardware.

3. **Upgrade a mirror of the production IIS server.** Mirror the current production IIS server (operating system, software, configuration settings, and content) to a second computer. Then upgrade the mirror to Windows 2000 Server with IIS 5.0 by using the Setup Wizard. Thoroughly test the upgraded server prior to deploying it and taking the original production Web server offline. (You might be able to use Iissync.exe to replicate configuration settings. For more information about using this tool, see the topic "Replication and Clustering in IIS" in the IIS 5.0 online product documentation.)

   Pros: By doing this, you avoid taking your production Web server offline for a potentially extended time period. Following deployment, if problems arise with the new server that didn't appear during testing, you have the original server available as a backup.

   Cons: All hardware on the second system must *exactly* duplicate the original IIS server, so you must forgo the option to upgrade your hardware when you upgrade IIS.

   Recommendation: This is an acceptable approach if you don't want to upgrade your hardware.

4. **Upgrade the production IIS server.** Take your existing IIS production Web server offline, and then upgrade it to Windows 2000 Server with IIS 5.0 by using the Setup Wizard. Thoroughly test the upgraded server prior to re-deploying it.

   Pros: No hardware cost.

   Cons: Upgrading a production Web server is extremely risky. You must take the server offline, and it will not be available to users until you complete all upgrade tasks, testing, and debugging.

   Recommendation: This method is *not* recommended except when you have implemented a Web server cluster and only need to take one of the clustered production Web servers offline at a time to implement the upgrade. The remaining servers in the cluster stay online and are fully functional.

Recommendations for Upgrading or Replicating

It is recommended that you do the following when upgrading or replicating an IIS Web server:

1. Back up the metabase. To do this, from the IIS snap-in, right-click the computer name, choose **Backup/Restore Settings**, click **Backup**, type a name for the backup file, and then click **OK**. Or you can use the Iissync utility to replicate the metabase.
2. Use the Iissync utility to replicate configuration settings to a different computer, as described in the "Replication and Clustering in IIS" topic of the IIS 5.0 online product documentation.
3. Back up all Web and FTP site content as well as security certificates to a different computer. You can use XCopy, included in the Windows 2000 operating system, to copy the Web directory structure recursively and preserve security settings.
4. Write down the Web site names and IP addresses.
5. Use one of the approaches listed in the previous section to upgrade or replicate the server to the upgraded or new IIS server.
6. Add all the IP addresses to the Network Information Center (NIC) on the upgraded or new server and configure the network accordingly.
7. Install Microsoft® Internet Explorer on the upgraded or new server.
8. In the IIS snap-in, assign IP addresses to the virtual directories.
9. Make sure that the IUSR account agrees with the IIS snap-in and User Manager for domains.
10. Using FrontPage Server Administrator, apply new FrontPage Extensions to the Web sites.

Migrating Web Applications

An important decision to make when migrating to IIS 5.0 is how to handle existing Web applications. IIS 5.0 supports applications based on CGI, ISAPI, and ASP. In some cases, it is best to simply port an application to IIS 5.0, making only a few necessary changes. This is usually true for existing ISAPI extensions and ASP applications, which make efficient use of server resources, and for CGI applications that are not heavily used. However, you might decide to rewrite CGI applications as either ISAPI extensions or as ASP applications in order to improve their scalability and performance as well as to simplify the development process. This will save both time and money.

This section discusses the following:

- Issues to consider when deciding whether to port a CGI application directly to IIS 5.0 or to rewrite it as an ASP application or ISAPI extension.
- The basic work involved in each approach to migration and references to more in-depth information.
- Changes that must be made to UNIX-based applications before they will run on IIS 5.0.

This material assumes you're familiar with Web-based applications and CGI. You can find additional references on migrating and porting applications in the "Additional Resources" section at the end of this chapter, and more information about developing with ASP in "Developing Web Applications" in this book. For information about installing and configuring applications, see the "Configuring Applications" topic of the IIS 5.0 online product documentation.

IIS 5.0 Application Technologies

The following is a brief description of the application technologies supported by IIS 5.0.

- **CGI** IIS 5.0 fully supports both scripts and executables written to the CGI specification. IIS 5.0 supports scripts written in a variety of languages when the appropriate Win32 script interpreter is installed and configured, for example, Perl, Python, TCL, REXX, and JScript. It also supports scripts written in VBScript that have been modified to support standard input and output variables. As a result, many CGI applications can be ported to IIS 5.0 with minimal changes.

- **ISAPI Extensions** ISAPI extensions consist of multithreaded DLLs loaded into IIS 5.0. ISAPI extensions have a strong performance advantage over CGI applications for several reasons. First, ISAPI extensions can be configured so all of them run in a single process, or so they run in the same memory space as the Web service. Second, instead of loading an executable for each request, ISAPI uses thread-safe DLLs that are loaded only once. Finally, ISAPI uses Win32-based APIs to communicate with the Web service, which is much faster than CGI methods. ISAPI extensions developed for other Web servers are generally easy to port to IIS 5.0. In addition, it is sometimes advantageous to rewrite existing CGI applications as ISAPI extensions to improve their performance, as discussed later in this section.

- **ISAPI Filters** ISAPI filters extend the capabilities of IIS 5.0. You can write an ISAPI DLL to intercept specific server events and perform appropriate actions. This functionality is especially useful in implementing server security. For more information, see http://www.bnt.com/inetsdk/httpfilt.htm#_in_Overview.

- **ASP** ASP is an open, server-application environment in which you can combine HTML, server-side scripts, and reusable COM components to create dynamic and powerful Web-based business solutions. IIS 5.0 natively supports scripts in ASP pages written in both VBScript and JScript, but you can use any scripting language when the appropriate script interpreter conforming to the Active Scripting standard is installed. ASP provides a number of intrinsic objects that make application development quick and easy, including **Application**, **Session**, **Request**, **Response**, and **Server** objects. ASP also supports COM components, which allow you to reuse business logic in other applications. ASP is supported by a number of Web servers, and existing ASP applications can be easily ported to IIS 5.0. In addition, it is often a good idea to rewrite CGI applications as ASP, as discussed later in this section, particularly those with functionality that you can quickly and easily reproduce by using ASP built-in objects.

  Note The terms "CGI script" and "script in an ASP page" can be confusing. Generally, when people use the term "CGI script," they are referring to a Perl script and then become mystified when this "CGI script" causes an ISAPI error. To ease the confusion, keep in mind that a script is simply a file containing instructions that are processed by some other component. A script in an ASP page, for example, might be a VBScript or a JScript file that is handed off to Asp.dll for processing. As with scripts in an ASP page, Perl scripts are handed off to a Perl interpreter for processing. However, some Perl interpreters are CGI-based executables and some are ISAPI-based DLLs—and the latter can create an ISAPI error.

Deciding to Port or Rewrite CGI Applications

An important decision you must make is whether to simply port a CGI application, making minimal changes needed for it to run on IIS 5.0, or to rewrite it as an ISAPI extension or an ASP application. Two basic considerations are as follows:

1. Expected performance gains compared to the work involved in rewriting the application.
2. Relative ease of developing and maintaining the application over time.

Performance vs. Development Work

Regarding the first consideration, ISAPI and the ASP scripting environment yield applications that are faster and more scalable than CGI applications. However, the work involved to migrate a particular CGI application to ISAPI or ASP can vary a great deal, depending on the structure of the application. In the case of a simple, clearly structured CGI application, you may be able to simply preserve the business logic and implement IIS 5.0 built-in input and output techniques. For a discussion of this approach, see "Migrating from CGI to ASP" later in this section. However, if the application is complex and the business logic is not easily separated from the rest of the application, rewriting it can be time-consuming. At this point, you need to carefully weigh the expected performance gains against the required effort.

Performance differences between CGI, ISAPI, and ASP are due to differences in server process overhead. CGI applications do not use server resources efficiently because a new, separate process is created for every client request, even if a single client submits more than one request. At any given time, the server must support a separate process for each ongoing request.

Unlike CGI applications, ISAPI extensions and ASP applications can be configured to either run in the same process as the Web service, or run in a pooled process. The server does not create a new process for each request, which greatly reduces the drain on resources. As a result, some benchmarks show that CGI applications run anywhere from two to three times slower than ASP applications and from three to five times slower than ISAPI extensions.

Besides avoiding server process overhead, ASP provides other performance-enhancing features, such as ODBC connection pooling and Microsoft® OLE DB session pooling, as described in "Data Access and Transactions" in this book.

Besides using more resources, CGI applications scale poorly on IIS 5.0. Adding additional processing power or RAM typically does not allow the application to support correspondingly more concurrent users. Therefore, if a CGI application is heavily used, you have compelling reasons to move to a solution that uses ISAPI or ASP, even when the application is complex and will be time-intensive to convert.

In deciding between ISAPI and ASP, bear in mind that ISAPI extensions offer some performance advantages over ASP applications, but ASP applications can often be developed more quickly.

Effort to Develop and Maintain

Regarding the second consideration, the ASP development environment has several advantages over that of CGI. First, the Microsoft® Script Debugger, IIS 5.0 application management features, and built-in database connectivity enhance ASP development productivity. Also, the ASP environment provides much of the functionality for managing forms, output, and state, so with ASP you might be able to remove a significant amount of existing CGI code, giving you a smaller code base to maintain.

Note When migrating an application, it's important to consider application architecture. To build a successful and scalable distributed application, where application logic can reside on a Web client, a Web server, and on other back-end servers (such as databases), you must separate the application's functionality into logical groups of services, or *tiers*.

For more information about three-tier architecture, about designing and building Web applications, and about reusable application components, see "Developing Web Applications" in this book and the "Windows Script Components" topic in the IIS 5.0 online product documentation. For references to other resources, see the "Additional Resources" section at the end of this chapter.

Porting CGI Applications

IIS 5.0 supports the CGI 1.1 standard. Porting a CGI application to IIS 5.0 from a Windows-based Web server is fairly simple, and you do not need to make many changes for the application to run. The most frequent problems encountered by individuals porting a Windows-based CGI application to IIS 5.0 are installing the application in the correct directory and configuring security. You also need to make sure you have the necessary script interpreter installed, if the application consists of a script. These issues are covered in the "Configuring CGI Applications" topic in the IIS 5.0 online product documentation.

Porting a CGI application to IIS 5.0 from a UNIX-based Web server requires some work to bring it into conformance with Windows file and application conventions. For more information, see "Special Considerations for UNIX Applications" later in this chapter.

Another type of CGI application, WinCGI, isn't as easy to port. WinCGI was developed as a simple way to get Visual Basic applications to work with the O'Reilly Web server. Visual Basic doesn't normally use the standard input (stdin) and output (stdout) streams or environment variables as do CGI- and character-based applications. Therefore, WinCGI takes the query string arguments, server variables, and stdin data and places them in an .ini file. The application then reads the .ini file for input and writes to it for output. WinCGI isn't supported on IIS 4.0 or IIS 5.0, and you cannot directly port applications based on WinCGI. You have three options for dealing with this type of application:

1. Rewrite it as a CGI that uses standard input and environment variables to transfer data.
2. Rewrite it as a script in an ASP page.
3. Write an ISAPI DLL to host WinCGI.

ASP is by far the best solution for anyone who wants to write IIS 5.0 applications in Visual Basic. ASP is very fast. In addition, it's designed to work with VBScript, and contains built-in objects that make application development easy—you can write a script in Notepad in seconds without having to build an .exe or a .dll file. Finally, ASP is robust with COM, is fully supported, and handles non-thread-safe components (although it's faster with thread-safe components).

Configuring a Script Interpreter

IIS 5.0 provides built-in JScript and VBScript interpreters for ASP, so you don't have to take any additional steps to run ASP applications written in these scripting languages. However, to run ASP applications written in Perl or to run CGI scripts, you must install and configure a Win32-compatible version of the appropriate script interpreter (also called a *scripting engine*).

To configure a script interpreter

1. **Obtain the script interpreter.** Perl 5.0 and Regina REXX script interpreters are included on the Resource Kit companion CD. You can also obtain Win32-compatible script interpreters on the Internet. The following Web sites provide Win32 implementations of Perl, TCL, Python, and REXX:

   Perl http://www.activestate.com/
   TCL http://www.scriptics.com/
   Python http://www.python.org/windows/
   REXX http://www.software.ibm.com/ad/obj-rexx/

   Note The ActiveState Tool Corporation provides three Perl interpreters of interest to IIS 5.0 developers: Perl for Win32 (Perl.exe), Perl for ISAPI (PerlIS.dll), and PerlScript, an Active Scripting interpreter for PerlScript code in ASP pages.

2. **Install the script interpreter.** Extract the script interpreter archive into a new directory on your hard drive, for example, c:\perl. Be sure directory names are expanded from the archive. Follow the instructions provided by the vendor to build the script interpreter from the source code package, and install the files. If the script interpreter package contains files with long file names, when downloading from the Internet be sure to use a zip file handler that opens them correctly, such as WinZip or Info-Zip Unzip.

   Note Before running Install.bat, you might want to change directory to the DOS version of your directory if you gave it a long file name. This will ensure that readable paths are listed in the registry.

3. **Set permissions.** In Windows Explorer, set NTFS permissions on the directory containing the script interpreter by giving Execute permissions to the Everyone account. Next, set permissions on the directory containing the actual scripts. For testing purposes, you can give Read permissions to the Everyone account. Otherwise, you should set access permissions to Script, and disable Read permissions. In the IIS snap-in, set Script only permissions on the scripts virtual directory. For more information, see Table 3.1, "Basic Web Security Settings."

4. **Set application mappings.** In the IIS snap-in, map the extension for the script file(s) to the executable for the script interpreter. For example, you might map the extension .py to Python.exe, the executable for the Python script interpreter. For more information, see the "Setting Application Mappings" topic in the IIS 5.0 online product documentation.

   Note For the ActiveState Perl script interpreter, the extension .pl is associated with PerlIS.dll by default. If you want to change the association of .pl to perl.exe, you need to change the application mapping. In the mapping, you must add two percent (`%`) characters to the end of the pathname for perl.exe, as shown in the following example:

   ```
   c:\perl\bin\perl.exe %s %s
   ```

5. **Set permissions on the script interpreter executable directory.** In the IIS snap-in, edit the application configuration for the home directory to recognize the script interpreter executable.

Special Considerations for UNIX Applications

There are special considerations when you're migrating Web applications from a UNIX environment. Not only do you need to be concerned with how the application functions on the Web server, but on the operating system as well. There are many differences between UNIX and Windows 2000 operating systems, and the degree to which your application is based on UNIX-specific technology affects how easy or difficult it will be to migrate.

The effort required to port a UNIX application to IIS 5.0 will be affected not only by the portability of the application, but also by the products and tools used to build it. This means that the database, round-robin DNS, developer tools, dynamic content pages, CGI applications, and any custom binaries or third party tools must work well with Windows 2000 Server.

The "ideal" UNIX-based application to port to IIS 5.0 would have the following characteristics:

- It would use the Perl language for CGI applications (Perl script interpreters are available for IIS 5.0).
- It would use FrontPage Extensions for Web publishing.
- It would access a Structured Query Language (SQL) Server or Oracle back end.
- Third-party tools it used would also run on Windows 2000 Server.
- Development tools used for it would work with Windows 2000 Server.
- It would not use any custom Apache modules.

This type of application would be a good candidate to migrate "by hand," converting it to a CGI, ISAPI, or ASP-based application that can run natively on IIS 5.0.

Most applications won't be so ideal. You might decide it isn't cost-effective to migrate some or all of your CGI applications, for example, if they're difficult to migrate, or if you have a large number of them. In this case, you might consider using a third-party tool such as PerlEX, a plug-in you can use with IIS 5.0 that dramatically improves the performance of Perl CGI applications. This way, you can run existing Perl CGI applications without creating a drain on IIS 5.0 resources. PerlEX also supports embedding Perl scripts within HTML. For more information, see http://www.activestate.com/plex/.

As an interim solution, you might also consider using a third-party tool that provides a UNIX-like environment for running applications on IIS 5.0. This way, you can gradually migrate applications to run natively on IIS 5.0, avoiding the need to rewrite a large amount of code in a short time frame. One such tool is Interix, by Softway Systems, Inc., which provides a Portable Operating System Interface (POSIX) environment. In addition, NutCracker, by DataFocus, has a run time that provides UNIX calls on Windows NT and Windows 2000. It uses the Win32 environment, which is more efficient than POSIX, and also provides access to COM.

For more information about available tools for migrating UNIX applications, see "Additional Resources" at the end of this chapter.

Migrating a UNIX Perl Script

Here are some issues you might encounter when migrating a Perl script from a UNIX-based Web server to IIS 5.0.

Binary Mode

If you plan to use the Perl for Win32 script interpreter to run UNIX Perl scripts, you might need to make modifications to ensure the scripts run correctly. When reading a text file from a disk, Windows translates `\r\n` into `\n`, and reads `^Z` as an end-of-file marker, but it doesn't use any such translation for binary files. For some Perl interpreters, the C run-time library opens files in text mode by default. For binary data, you should specify binary mode for the file handle by using the binmode function. For more information about doing this, see the perlfunc documentation.

Pathname Delimiters

When migrating a Perl script from UNIX, remember that Windows uses the backslash (`\`) character as a delimiter for paths in a file name, for example, c:\path\somefile.txt. Because Perl uses the backslash as an escape code, to symbolize a special character such as the line feed character (`\n`) or the tab character (`\t`), an error results if you attempt to open a file in this manner:

```
Open( MYFILE, "C:\path\somefile.txt" )
```

To resolve this problem, you can replace each backslash with a double backslash (`\\`), in order to indicate an actual backslash rather than the introduction of a special character. Or you can use noninterpolating single quote strings in order to indicate there are no special characters in the string, as follows:

```
Open( MYFILE, 'C:\path\somefile.txt' );
```

Converting UNIX Application Files

There are many differences between UNIX and Windows file descriptors and conventions. Those pertinent to migrating a Web application to IIS 5.0 are listed in the following table. To avoid application errors, you must make the necessary changes so your UNIX files adhere to Windows conventions.

Table 3.9 Necessary Changes to UNIX Application Files

Feature	UNIX Implementation	Change to This Windows Implementation
Directory separator	Forward slash (/).	Backslash (\). Note that Perl uses the backslash as an escape code, so in Perl scripts you'll want to replace the forward slashes (/) in paths (pathnames) with double backslashes (\\). Or you can use non-interpolating single quote strings to indicate there are no special characters in the string.
File names and pathnames	See the earlier discussion in "Converting UNIX File Names and Pathnames."	
Directory hierarchy	The UNIX file system appears to be a single directory hierarchy.	Windows storage is divided into a number of physical or virtual disk drives with a directory hierarchy on each. To access a file on Windows, you must know what disk drive the file is on and specify the drive letter (C, D, E, and so forth) as part of the file path (pathname).
Text file line termination	UNIX uses only a line feed character to flag an end of a line in text files.	Text file lines should be terminated by a line feed and a carriage return. Some applications require both line feed and carriage return terminations to work correctly.
File descriptors/handles	UNIX file descriptors 0, 1, and 2 represent `stdin`, `stdout`, and `stderr`, respectively.	Win32 uses *handles* rather than descriptors, and you can't assume that they're fixed as you can in UNIX. Use the Win32 API call, `GetStdHandle()`, to ascertain which handles are being used.

Converting CGI to ISAPI

ISAPI extensions are Web applications implemented as DLL files. Converting a CGI application into an ISAPI extension will improve its performance.

ISAPI extensions run faster than any other type of application on IIS 5.0. ISAPI extensions run in the same memory space as the Web server, or in a pooled application process, unless a separate process is called. IIS 5.0 typically loads an ISAPI DLL the first time a request that calls the DLL is received. The DLL then remains in memory, ready to service other requests until the Web server is stopped or until the DLL is manually or programmatically unloaded.

Rather than using environment variables and standard input/output (I/O), as CGI does, ISAPI extension DLLs communicate through a data structure called the Extension Control Block (ECB). A small set of functions allow the DLL to communicate with the Web server. Because function pointers are used to communicate with the server, applications written in languages that support function pointers, such as C, C++, and Perl, are good candidates for conversion to ISAPI. CGI applications written in Visual Basic should be rewritten as ASP applications.

Perl scripts run most efficiently under ISAPI. PerlIS.dll, also called Perl for ISAPI, is an ISAPI extension that runs Perl scripts on the Windows operating system. Using the Perl for Win32 Perl.exe rather than PerlIS.dll to run your Perl scripts requires that the server create a new process for each request. For information about obtaining and installing PerlIS.dll, see "Configuring a Script Interpreter" earlier in this chapter.

Creating ISAPI DLLs requires a thorough understanding of the Win32 programming environment, including thread management. For some useful information sources on Win32 programming, see the "Additional Resources" section at the end of this chapter.

Note An ISAPI DLL can be deployed by the developer or system administrator to run either within the IIS 5.0 process, in a pooled process with other DLLs, or in its own process. Developers frequently configure their applications to run out of process until they are fully tested because, should they malfunction, they won't affect other applications running on the computer. Once applications are fully tested, they are then configured to run in a pooled process for faster execution. Or if an application is prone to violation errors and failures, it can continue to run out of process on the production Web server.

Migrating from CGI to ASP

Whenever possible, migrate CGI applications to ASP, to take advantage of improved performance and the simplified development environment. Converting to the ASP scripting environment may not require as much effort as you think, because when you examine the application code you might find that much of it is unnecessary in the ASP environment. All you might need to do is port the business logic, greatly reducing the work involved. Removing CGI code that isn't required within the ASP environment—form processing, environment variable processing, e-mail manipulation, database handling, state management, and most importantly, output processing—can result in much smaller applications to migrate.

ASP Scripting Support

ASP enables server-side scripting for IIS 5.0 with native support for VBScript and JScript development software. In addition to these native languages, the ASP environment supports any scripting language that conforms to the Active Scripting requirements when its script interpreter is installed and configured. You can obtain ASP implementations of the Perl and Python script interpreters. (Perl 5 is included on the Resource Kit companion CD.)

You can select the scripting language to use within an ASP page. If you're migrating a CGI application written in C or Java, it may be easiest to use JScript because of its syntactical similarity. If the CGI application is written in Perl, you might find it convenient to use PerlScript.

In any case, consider using VBScript for building ASP pages. Millions of developers use the Visual Basic language. Organizations that move to VBScript can take advantage of this synergy and might find they have many more resources at their disposal.

Analyzing the Application

The first step in porting a CGI application to ASP is to analyze its logic. Most Web applications consist of five basic types of logic:

- Input processing (reading forms, environment variables, and decoding HTML)
- Business logic
- Database (or gateway) logic for connecting to external services
- Logic for maintaining state between forms
- HTML output logic for returning results to browser clients

The following is a general discussion on how to handle each of these types of logic within the ASP environment.

Input Processing

A great deal of CGI processing involves capturing the contents of HTML forms as presented to the standard input stream, then reformatting them, decoding the HTML, reading environment variables, and so on. Because this is one of the most tedious aspects of developing CGI applications, many developers use third-party libraries and tools, such as the Perl utilities Cgi.pm or Cgi-lib.pl. In most cases, these utilities are unnecessary in ASP applications because the ASP environment takes care of most of these housekeeping tasks for you, as described in the following paragraphs.

Accessing Form Input and Decoding HTML

One of the major differences between CGI applications and ASP applications is the method for handling form input. Accessing form data from an ASP page is much simpler than it is from a CGI application. When migrating to ASP, you can often remove most of the third-party form decoding and processing utilities, as well as any custom generic form-processing modules.

A CGI application written in Perl receives form input from a POST request on the standard input stream using code such as:

```
$form_size = $ENV('CONTENT_LENGTH');
read( STDIN, $form_data, $form_size );
```

This form data is encoded and must be translated. Using Perl, the translation code might look like:

```
$value =~ s/%([0-9a-fA-F]{2})/pack("c",hex($1))/ge;
```

Once the form data is decoded, the developer can search for a form variable. Here is a typical Perl routine to parse the form data and place it in a list:

```
sub get_form
{
   local (*FORM) = @_;
   local ( $env_string, @collection, $key_value,
            $key, $value );
   read (STDIN, $env_string, $ENV{'CONTENT_LENGTH'});
   @collection = split( /&/, $env_string );
   foreach $key_value (@collection)
   {
      ($key, $value) = split( /=/, $key_value);
      $value =~ tr/+/ /;
      $value =~ s/%([0-9a-fA-F]{2})/pack("c",hex($1))/ge;
            if (defined($FORM{$key}))
{
         $FORM{$key} = join("\0",, $FORM{$key}, $value);
      } else
{
         $FORM{$key} = $value;
      }
   }
}
```

Once the routine is in place, it can be called to parse the form. The following code calls the `get_form` routine and references a variable called `home_address`:

```
&get_form(*my_form);
print $my_form('home_address');
```

Fortunately, the ASP environment takes care of these form processing and decoding chores for you. In ASP, the content of an HTML form is made available as a *collection*, which is a named list of key/value pairs. For developers familiar with languages like Perl or Awk, collections are analogous to hashes or associative arrays.

Form variables are included in the **Form** collection of the ASP **Request** object.

Instead of referring to a form variable with a Perl construction like this:

```
print $my_form('home_address');
```

you can refer to a form variable as shown in the following VBScript example:

```
MyAddress = Request.Form("home_address")
```

The following is the same instruction written in PerlScript:

```
$MyAddress = $Request->Form('home_address')
```

There is no need to parse the input, as was required in the CGI example, because the ASP environment takes care of parsing.

Although ASP applications do an excellent job of eliminating form-processing drudgery, developers sometimes need to access the unprocessed input stream. The following VBScript fragment writes the unprocessed input stream back to the output stream:

```
Response.Write Request.Form
```

In addition to the **Form** collection, the **Request** object includes a collection called **QueryString**. This collection contains form elements sent in response to the GET method of the HTML <FORM> tag. The elements of the **QueryString** collection are accessed in the same way elements of the **Form** collection are accessed.

Often, quite a bit of CGI code is devoted to determining whether a given form variable exists within the Query_String environment variable or in the standard input stream. However, within the ASP environment, you can avoid testing many of these conditions simply by referring to a variable by name:

```
MyVariable = Request("home_address")
```

In this case the Web server will search the Query_String first, then search the form, in order to find the correct variable. This feature can eliminate the need to rewrite code that is used for searching both the query string and the form.

To encode URLs or HTML, you can use the built-in **HTMLEncode** and **URLEncode** methods of the **Server** object.

Environment Variables

CGI applications that inspect environment variables can continue to do so when converted to ASP. Environment variables are provided as part of the **Request** object **ServerVariables** collection.

In a CGI application, you might access the Server_Name environment variable using a line of C code like the following:

```
serverName = getenv("SERVER_NAME");
```

or this line, written in Perl:

```
serverName = $ENV{'SERVER_NAME'};
```

In ASP, you can use the following instruction, written in VBScript:

```
serverName = Request.ServerVariables("SERVER_NAME")
```

You can iterate through collections such as **ServerVariables** in order to examine all the values they contain. The following ASP code, written in VBScript, generates an HTML table containing all the variables contained in the **ServerVariables** collection, with their values:

```
<%@ LANGUAGE=VBSCRIPT %>
<HTML>
  <BODY>
    <TABLE BORDER=1>
    <% For Each Name in Request.ServerVariables %>
    <TR>
      <TD><%= Name %></TD>
      <TD><%= Request.ServerVariables(Name)%></TD>
    </TR>
    <% Next %>
    </TABLE>
  </BODY>
</HTML>
```

Business Logic

Usually, once much of the "plumbing" is removed from a CGI application, some business logic remains to be ported. Approaches to migrating business logic include:

- Taking business logic written in a language such as C, C++, or Perl, and rewriting it as an ASP-supported scripting language such as VBScript, JScript, or PerlScript. This is a good approach when the business logic is fairly simple and if it is useful in only one or two applications.
- Encapsulating the business logic within a component. If the logic is extensive, requires more functionality than is available in a scripting environment, or is very general, this is a good approach. You can write components in any language that supports COM, including C, C++, Java, Visual Basic, Delphi, and even some implementations of COmmon Business Oriented Language (COBOL). The advantage of using components is that they can be reused in other business applications both within and outside the Web environment.

  Note Server-side components run completely within the server environment. Their use has no effect on whether an application can support a particular browser on a specific platform.

External Gateway and Database Logic

CGI applications were originally created to give Web clients access to applications outside the Web environment. Most CGI applications interface with databases of some sort, and many CGI applications are used to access services such as electronic mail servers. This section describes techniques for using ASP to access databases and other external services.

Databases

Microsoft® ActiveX® Data Objects (ADO) is a database access method you can use within the ASP environment. ADO is designed to be highly efficient in a multithreaded environment, where many instances of the ADO code execute simultaneously. Thus, ADO is ideal for Web applications, particularly when they need to scale to many concurrent users.

ADO exposes an object model to abstract the ideas of connecting to, and executing commands against, a remote database. The database need not support SQL, but if it doesn't, a database-specific piece of software called an *OLE DB provider* is required. If your database vendor supplies an ODBC driver for your database, you can use a provider for ODBC data sources that is supplied with IIS 5.0.

Perl developers commonly access a database by using a package such as Dbperl or the more current DBI package. No special package is required within the ASP environment. For example, the following simple ASP page, written in PerlScript. uses the ODBC data source **ADOSamples** to dump the contents of a table called "Orders":

```
<%@ LANGUAGE=PerlScript %>
<HTML>
  <BODY>
    <P>
    <%
       $Conn = $Server->CreateObject("ADODB.Connection");
       $Conn->Open( "ADOSamples" );
       $RS = $Conn->Execute( "SELECT * FROM Orders" );
    %>
    <TABLE BORDER=1>
    <TR>
    <%
       $count = $RS->Fields->Count;
       for ( $i = 0; $i < $count; $i++ )
    {
          %><TD><B><%= $RS->Fields($i)->Name %></B></TD><%
       }; %> </TR> <%
       while ( ! $RS->EOF )
    {
          %> <TR> <%
          for ( $i = 0; $i < $count; $i++ )
    {
             %><TD VALIGN=TOP>
             <%= $RS->Fields($i)->value %></TD><%
          };
          %> </TR> <%
          $RS->MoveNext;
       };
       $RS->Close;
       $Conn->Close;
       %>
    </TABLE>
  </BODY>
</HTML>
```

An additional benefit of working within the ASP environment is the ability to take advantage of Component Services. Not only does Component Services provide support for process isolation, as discussed previously, it also enables developers to easily build scalable Web applications based on components. Component Services takes care of all the plumbing issues such as transactions, connection management, and thread management, thus freeing the developer to concentrate primarily on building business logic.

For more information about ADO and Component Services, see "Data Access and Transactions" in this book and see http://www.microsoft.com/com/.

For more general information about ADO, ODBC, and Microsoft's overall data access strategies, see http://www.microsoft.com/data/.

For more information about the Perl-based DBI package and for documentation, including ideas for porting existing DBI code to the Win32 platform, see http://www.symbolstone.org/.

Maintaining State

There are several schemes commonly used in CGI applications for maintaining state information about the client session on the Web server. Some of these involve scripts that save and restore data from server-side text files. Others involve embedding state in the client-side HTML (for example, using hidden form fields). Some CGI applications even maintain state by setting themselves up as "mini Web servers" for the duration of the session, intercepting incoming HTTP requests. Many Web developers have moved to using browser-based cookies for storing state information.

ASP provides built-in state management through the **Session** collection of variables, making session management transparent to the Web developer. When a client first connects to the application, the server can send a *session cookie* that identifies the new session. As the browser accesses other pages in the application, the cookie is retrieved by the system and is used to manage state.

For example, consider an ASP page in which the user's login name is retrieved and assigned to a session variable:

```
Session("login_name") = Request.Form("login_name")
```

The value of `login_name` will be available to all other pages within the application, until the session times out (the default is 20 minutes, but it is configurable) or is explicitly abandoned.

Output Handling

A key difference between ASP and CGI applications is that in an ASP page you can interweave industry-standard HTML and server-side scripting code to deliver the appropriate HTML to the client. By doing so, you combine the most powerful elements of both environments.

Using standard HTML as part of the application code is a benefit that cannot be overemphasized. With this feature, the developer can emit any HTML that the client browser supports, whereas utilities such as Cgi.pm are necessarily limited by the tags they implement. With new developments such as DHTML, tools such as Cgi.pm must be modified, perhaps heavily, to support new tags or other new client-side features.

In addition, with ASP, Web designers and nonprogrammers can author in pure HTML. On the other hand, having to use a special HTML dialect, as required by CGI tools such as Cgi.pm makes it difficult for these users to modify the generated HTML.

Many CGI applications emit HTML by using the output facilities of the language in which they're written. For example, a simple Perl-based CGI application might look like this:

```
#!/usr/local/bin/perl
print "Content-type: text/html", "\n\n";
$server_name = $ENV('SERVER_NAME');
print "<HTML><BODY>Your server name is ", $server_name;
print "</BODY></HTML>";
exit(0);
```

Note that the HTML is embedded within the print function. The following is the corresponding ASP page, with code written in VBScript:

```
<HTML>
  <BODY>
    Your server name is <%= Request.ServerVariables("SERVER_NAME") %>
  </BODY>
</HTML>
```

Note that the entire page is just simple HTML, except for the VBScript expression

```
<%= Request.ServerVariables("SERVER_NAME") %>
```

For some applications you must specify HTTP header information, rather than using the default MIME type of text/html. In the ASP environment, you can accomplish this by using the **ContentType** property of the **Response** object. For example, the following instruction sets the content type to "text/plain":

```
<% Response.ContentType = "text/plain" %>
```

Many Perl-based CGI applications use the Cgi.pm or Cgi-lib.pl modules for formatting HTML output, as well as for performing other tasks. For example, the following script uses Cgi.pm to generate a form that collects the user's address; once the form is submitted, the form is redisplayed with the address beneath it:

```
use CGI qw(:all);
print header;
print start_html('Enter your address'),
   h1('Enter your address'),
   hr,
   p,
   start_form,
   table(
      Tr(td("Street"),td(textfield('street'))),
      Tr(td("City"), td(textfield('city'))),
      Tr(td("State"), td(textfield('state'))),
      Tr(td("Zip"), td(textfield('zip')))
   ),
   submit,
   end_form,
   hr;

if (param())
{
      print
         "Street is: ", param('street'), p,
         "City is: ", param('city'), p,
         "State is: ", param('state'), p,
         "Zip is: ", param('zip'), p,
         hr;
}
```

The preceding script provides a convenient way to lay out a form if you aren't able to generate the HTML as part of the source code. The drawback to these utilities is that they require the developer to master a "pseudo-HTML" dialect in order to generate the page.

The corresponding ASP page looks almost exactly like the actual HTML page that will be sent to the browser, except for the VBScript that is interwoven with the HTML:

```
<HTML>
  <HEAD>
    <TITLE>Enter your address</TITLE>
  </HEAD>
  <BODY>
    <H1>Enter your address</H1>
    <HR>
    <P>
    <FORM METHOD=POST>
    <TABLE>
       <TR><TD>Street</TD><TD><input name="street"></TD></TR>
       <TR><TD>City</TD><TD><input name="city"></TD></TR>
       <TR><TD>State</TD><TD><input name="state"></TD></TR>
       <TR><TD>Zip</TD><TD><input name="zip"></TD></TR>
    </TABLE>
    <INPUT TYPE=SUBMIT>
    </FORM>
    <HR>

    <%   If Request.Form.Count > 0 %>
    Street is: <%= Request.Form("street") %> <P>
    City is:  <%= Request.Form("city")  %> <P>
    State is: <%= Request.Form("state") %> <P>
    Zip is:  <%= Request.Form("zip")  %> <P>
    <HR>
    <% End If %>

  </BODY>
</HTML>
```

The File System

CGI applications frequently interact with the file system. There are several approaches to accessing the file system from within the ASP environment. The easiest method is to use the **FileSystemObject** and **TextStream** objects, which provide high-level access to the file system. For example, the following ASP page, written in VBScript, creates a text file, and writes the contents of the environment variable Server_Name to its own line:

```
<%@ LANGUAGE=VBScript %>
<HTML>
  <BODY>
    <%
       Dim objFS, objFile, append
       append = 8
       Set objFS  = Server.CreateObject("Scripting.FileSystemObject")
       Set objFile = objFs.OpenTextFile("c:\servlist.txt", append, True )
       objFile.WriteLine Request.ServerVariables("SERVER_NAME")
       objFile.Close
    %>
    Done
  </BODY>
</HTML>
```

If the supplied **FileSystemObject** and **TextStream** objects do not meet your needs, file system access can be encapsulated with a COM component. To learn more about components, see the "COM Concepts" topic in the IIS 5.0 section of the SDK documentation on MSDN, and see http://www.microsoft.com/com/.

Reproducing Common CGI Services

It is easy to reproduce many common CGI services, such as e-mail message delivery and page counters, by using ASP intrinsic objects or a comparable Win32 application. This section describes how to reproduce some of the most common CGI services on IIS 5.0.

Electronic Mail Delivery

Many CGI applications, such as sendmail, format RFC-822 e-mail messages for delivery to a mail server. You can easily reproduce this functionality by using ASP. Or you can make minor modifications to your sendmail script, and then run a comparable Win32 mail application, such as Blat.exe.

Using ASP to Send E-mail Messages

IIS 5.0 includes a SMTP service and exposes all of its functionality through the Collaboration Data Objects (CDO) object model. When the SMTP service is installed and configured, you can use the **CDONTS** object included with IIS 5.0 to send an e-mail message from an ASP page, as shown in the following example, written in JScript:

```
<%@ LANGUAGE=JScript %>
<%
   var msg;
   msg = Server.CreateObject("CDONTS.NewMail");
   msg.From = "someone@microsoft.com";
   msg.To  = "someone@microsoft.com";
   msg.Subject = "Sample message";
   msg.Body = "This is a sample message.";
   msg.Send();
%>
```

or in VBScript:

```
<%@ LANGUAGE=VBScript %>
<%
    Dim objMail
    Set objMail = CreateObject("CDONTS.NewMail")

    objMail.Send "someone@microsoft.com","someone@microsoft.com","Message subject","Message
    body"
%>
```

The following is an ASP page written in VBScript that illustrates how to capture the Http_Referer, Remote_User environment variables from a form and return the message, "Thank you *User_Name* for sending mail."

```
<%@ LANGUAGE=VBScript %>
<%
    '*** Load Constants
    SENDTO = "someone@microsoft.com"
    SUBJECT = "Form Reply-Response Example"
    USERNAME = Request.ServerVariables("LOGON_USER")

    '*** Post values to the form
    name = Request.Form("name")
    email = Request.Form("email")
    telephone = Request.Form("telephone")
    category = Request.Form("category")

    '*** Build message body
    BodyString = BodyString & "Name =  " & name & VbCrLf
    BodyString = BodyString & "Email =  " & email & VbCrLf
    BodyString = BodyString & "Telephone =  " & telephone & VbCrLf
    BodyString = BodyString & "category =  " & category & VbCrLf

    '***Begin Send mail to customer***
      Set myMail =

    Server.CreateObject("CDONTS.NewMail")
      myMail.From = email
      myMail.To = SENDTO
      myMail.Subject = SUBJECT
      myMail.Body = BodyString
      myMail.Send
      Set myMail = Nothing
    '***End Send mail to customer***

    '***Build confirmation body***
      OutputString = "<P></BODY></HTML>"
      OutputString = OutputString & "<P>  <P>Thank you " & USERNAME & " for
      sending mail<P>" & VbCrLf

    '***Begin Build footer and send to customer***
    OutputString = OutputString & "<P></BODY></HTML>"
    Response.Write OutputString
    '***End Build footer and send to customer***
%>
```

The following example, also in VBScript, shows how to send an e-mail attachment.

```
att_file="c:\Reports\wklyRpt.txt"
f_name="WklyRpt.txt"
Set objMail = CreateObject("CDONTS.NewMail")
objMail.From="someone@microsoft.com"
objMail.To="someone@microsoft.com"
objMail.Subject="Weekly Report"
objMail.Body="Attached below is my weekly report file."
objMail.AttachFile att_file,f_name
mailres=objMail.Send()
```

Migrating a UNIX Mail Script

A common CGI application in UNIX uses commands to call a Perl script, which then pipes messages to the "mail" e-mail application. In the following example, the Perl script is named MAILPROGRAM:

```
#Open a pipe to the MAILPROGRAM script located in /usr/local/bin:
Open (MAILPROGRAM, "|/usr/local/bin/mail someone@microsoft.com");
#Write the e-mail message:
Print MAILPROGRAM $emailMessageText;
#Close the pipe and send the message:
Close MAILPROGRAM;
```

When migrating to Windows 2000 Server, you can modify the Perl MAILPROGRAM script to call a Win32-compatible e-mail application, named Blat.exe. This is a simple, command-line SMTP e-mail application that works very much like mail on UNIX. You can obtain Blat.exe on the Resource Kit companion CD, or see http://www.shareware.com.

Install Blat.exe in the Web site applications directory, and be sure to configure it in the IIS 5.0 snap-in. Also, in the Services Control Panel, disable the SMTP service that comes with Windows 2000 Server.

Here is the Perl MAILPROGRAM script as used on UNIX:

```
Open(MAILF,"|mail-s\"Calendar activity $what $boss");>>print MAILF"For$today";
Print MAILF",and continuing for $days days";
Print MAILF"$val{$what}was";
Print MAILF"added to the schedule:\n";
Print MAILF"$val{$from}$val{$till}$work";
Print MAILF"removed from the schedule:\n";
Print MAILF"(originally was $gonzo)";
Close MAILF;
```

To run this script on Windows 2000 Server, replace `mail` with `blat` in the first line, as follows:

```
Open(MAILF,"|blat-s\"Calendar activity $what $boss");>>print MAILF"For$today";
```

Page Counters

To improve the performance and efficiency of your Web site, you can replace CGI-based page counters with the ASP Page Counter Component included with IIS 5.0. This creates a **PageCounter** object that counts and displays the number of times a Web page has been opened. It also writes the number of page hits to a text file. The following ASP script, written in PerlScript, demonstrates how to query the component and display the hit count:

```
<%@ LANGUAGE=PerlScript %>
<HTML>
  <BODY>
    <%
       $counter = $Server->CreateObject("MSWC.PageCounter");
       $hits  = $counter->Hits;
    %>
  You are visitor number <%= $hits %> to this page.
  </BODY>
</HTML>
```

For more information about ASP components, see the "Module 2: Using COM Components" topic in the ASP Tutorial, found in the IIS 5.0 online product documentation.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server, all of which can help you perform migration.

IIS 5.0 Migration Tools

IIS Migration Wizard

The Resource Kit companion CD includes the IIS Migration Wizard, which automatically migrates most configuration settings to IIS 5.0 from the following Web servers:

- Netscape Enterprise Server 3.5 (Windows version). (You can find the wizard to migrate from Netscape Enterprise Server 2.*x* and 3.*x* to IIS 4.0 in the *Microsoft® Internet Information Server 4.0 Resource Kit.*)
- Apache HTTP Server 1.3 (UNIX version).
- IIS 4.0.
- IIS 5.0. (This feature is useful for replicating an IIS 5.0 server.)

Note The IIS Migration Wizard is provided for educational purposes only and is not supported by Microsoft.

Application Migration Tools

http://msdn.microsoft.com/developer/sdk/

The Microsoft Platform Software Development Kit (SDK)—January 1998 Edition: Current and emerging Microsoft technologies including tools, headers, libraries, and sample code, as well as information about developing ISAPI extensions. This is the successor to the Win32 SDK, and includes components that have been distributed separately in the Microsoft® Win32 SDK, the Microsoft® BackOffice® SDK, the Microsoft® ActiveX/Internet Client SDK, and the Microsoft® DirectX SDK.

http://msdn.microsoft.com/scripting/

Information about Microsoft Windows script technologies, including JScript, VBScript, and Microsoft® Windows® Script Host (WSH).

http://www.activestate.com/

ActiveState provides three Perl interpreters of interest to IIS developers: Perl for Win32, Perl for ISAPI, and PerlScript, an Active Scripting interpreter that runs PerlScript in ASP pages. In addition, it provides PerlEX, a plug-in you can use with IIS 5.0 that dramatically improves the performance of Perl CGI scripts and allows you to embed Perl code within HTML.

http://www.interix.com/

To Windows 2000 Server and IIS 5.0, Interix adds the ability to host complete Internet applications developed for UNIX systems. It provides full support for sendmail, Perl and TCL/TK, UNIX commands and shells, including rcommands, and remote access with telnetd and rlogind.

http://www.datafocus.com/

This Porting Guide by DataFocus Inc. describes how to use Nutcracker to migrate UNIX and X/Motif applications to Windows 2000, Windows NT, and Microsoft® Windows® 95 as native Win32 applications.

http://www.mks.com/

Mortice Kern Systems makes the MKS Toolkit, which provides many utilities, such as htdiff, htsplit, url, and Web, to help you automate Web development and maintenance tasks. The Toolkit includes a version of Perl, as well as Pscript, an Active Scripting interpreter that allows you to use PerlScript code in an ASP page.

http://www.iasoft.com/defroster/

Defroster by IntraActive converts Cold Fusion applications to ASP applications. At the time of publication of this book, versions exist for IIS 3.0 and IIS 4.0.

http://www.python.org/windows/

Provides Python language products for the Windows operating system.

http://www.scriptics.com/

Provides Tcl scripting language products for the Windows operating system.

http://sourceware.cygnus.com/cygwin/
Includes GNU development tools for the Windows operating system.

http://cgi-lib.stanford.edu/cgi-lib/
Provides information about the Cgi-lib.pl library.

http://stein.cshl.org/WWW/software/CGI/
Provides information about the Cgi.pm utility.

http://www.symbolstone.org/technology/index.html/
Provides a Perl-based DBI package and documentation, including ideas for porting existing DBI code to the Win32 platform.

Server Administration and Interoperation Tools

http://www.microsoft.com/ntserver/nts/exec/overview/sfu.asp
Windows NT Services for UNIX, a suite of interoperability tools and utilities for Windows NT Server and UNIX. At the time of this writing, a version of this tool suite is under development for Windows 2000.

http://www.microsoft.com/ntserver/ntserverenterprise/techdetails/compares/WindowsNTUNIX.asp
This white paper lists Microsoft interoperation initiatives and partners.

http://www.netmanage.com/
NetManage, Inc. provides PC connectivity solutions, such as Chameleon UNIX Link 97, version 8.0, with significant enhancements to X Windows and NFS applications, extended browser access to UNIX, AS/400, and mainframe systems, as well as SupportNow technology. Chameleon UNIX Link 97 supports Windows 3.*x*, 9*x*, and Windows NT.

http://www.itribe.net/virtunix/
Windows versions of popular UNIX tools.

http://www.shareware.com/
Offers a tool called Ws_ftp to perform recursive FTP for the Windows operating system.

Reference Books

Migrating to Windows NT by Steve Heath, 1997, Houston: Digital Press.

Windows NT & UNIX, Administration, Coexistence, Integration, & Migration by G. Robert Williams and Ellen Beck Gardner, 1998, Reading: Addison Wesley Longman, Inc.

Windows NT, UNIX, NetWare Migration and Coexistence, A Professional's Guide by Raj Rajagopal, 1998, Boca Raton: CRC Press LLC.

Windows NT & UNIX Integration Guide by David Gunter, Lola Gunter, et al, 1997, Berkeley: Osborne McGraw-Hill.

Security Resources

http://msdn.microsoft.com/workshop/c-frame.htm?920326470354#/workshop/server/
An article on IIS security, "Untangling Web Security: Getting the Most from IIS Security."

http://www.microsoft.com/security/default.asp
The Microsoft Security Advisor.

In addition, the following Internet resources provide general information about security:

CERT Advisory
ftp://cert.org/.message/

Usenet News
Alt.comp.security

Search Yahoo
You'll find many listings under the general search term "security."

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 4

Capacity Planning

Capacity planning for a Web server installation involves determining the current and future needs of the installation, then choosing hardware and software that can meet *current* estimated needs, but that can also be expanded or upgraded to meet *new* needs as they arise. Because there are so many variables, and because needs change so rapidly, capacity planning is more of an art than a science, and typically requires an iterative approach.

This chapter is intended primarily for people who decide what equipment and software to acquire for Web sites, and for site administrators. It explains some aspects of Web server capacity, provides ways to determine where bottlenecks are likely to occur, and shows how much network bandwidth you will need. It also contains a case study that provides insight into large-site issues, many of which can and do arise at smaller sites as well.

In This Chapter

Capacity Planning Issues

Determining Your Installation's Requirements

A Capacity Planning Checklist

Capacity Planning Scenarios

A Large-Site Case Study: microsoft.com

Additional Resources

Capacity Planning Issues

Creating and maintaining a Web site involves managing traffic with hardware, software, and network bandwidth. This first section explores traffic.

Traffic

Servers send out pages in response to requests. In order to issue a request, a browser first establishes a Transmission Control Protocol (TCP) connection with the server, and then sends the request via that connection. Traffic, as the term applies to Web servers, is the resulting mixture of incoming requests and outgoing responses.

In general, traffic occurs in bursts and clumps that are only partly predictable. For example, many intranet sites have activity peaks at the beginning and end of the day, and around lunchtime; however, the exact size of these peaks will vary from day to day and, of course, the actual traffic load changes from moment to moment. Not surprisingly, there is a direct relationship between the amount of traffic and the network bandwidth needed. The more visitors your site has and the larger the pages it provides, the more network bandwidth your server requires.

To start with a simple example, imagine a server that displays static Hypertext Markup Language (HTML), text-only pages that average 5 kilobytes (KB) in size, which is about equivalent to a full page of printed text. The server is connected to the Internet through a DS1/T1 line, which can transmit data at 1.536 megabits per second (Mbps). Inherent overhead makes it impossible to use the full-rated T1 bandwidth of 1.544 Mbps. How many pages per second can the server send out under optimum conditions? To answer this question, it is necessary to look at the way the information travels between computers.

Data travelling on a network is split into packets. Each packet includes, in addition to the data it carries, roughly 20 bytes of header information and other network protocol information (all this extra information constitutes "overhead"). The amount of data in a packet is not fixed, and thus the ratio of overhead to data can vary. Most incoming HTTP requests are small. A typical request (for example, `GET http://www.microsoft.com/default.asp`), including the Transmission Control Protocol/Internet Protocol (TCP/IP) overhead, consists of no more than a few hundred bytes as it travels across the network. For a 5-KB file like the one in this example, protocol overhead is significant, amounting to about 30 percent of the file's size. For larger files, the overhead accounts for a smaller percentage of network traffic. Overhead can become an important consideration when you are estimating your site's bandwidth requirements and deciding how fast a connection you'll need. If you are close to the limits of your connection's capacity, an extra 20 percent for overhead may mean that you will require the next fastest type of connection.

Table 4.1 shows the traffic generated by a typical request for a 5-KB page. Note that all the figures for overhead are estimates. The precise number of bytes sent varies with each request.

Table 4.1 Traffic Generated by a Request for a 5-KB Page

Traffic Type	Bytes Sent
TCP Connection	180 (approx.)
GET Request	256 (approx.)
5-KB file	5,120
Protocol overhead	1,364 (approx.)
Total	**6,920**

To find the number of bits, multiply the number of bytes sent by 8 bits per byte: 6,920 x 8 = 55,360. As stated previously, a T1 line can transmit 1.536 Mbps. Dividing bits per second by bits per page, 1,536,000/55,360, provides a maximum rate of just under 28 pages per second. (Because modems add a start bit and a stop bit to each byte, they are slower than the raw numbers appear to indicate.)

Table 4.2 illustrates the relative speeds of several network interface types, using the small, text-only page just mentioned. Numbers of pages transmitted at speeds faster than the standard Ethernet rate of 10 Mbps are rounded.

Table 4.2 Relative Network Interface Speed

Connection Type	Connection Speed	5-KB Pages Sent per Second
Dedicated PPP/SLIP via modem	28.8 Kbps	Roughly half of 1 page
Frame Relay or fast modem	56 Kbps	Almost 1 page
ISDN	128 Kbps	Just over 2 pages
Typical DSL	640 Kbps	Almost 11 pages
DS1/T1	1.536 Mbps	26 pages
10-Mb Ethernet	8 Mbps (best case)	(Up to) 136 pages
DS3/T3	44.736 Mbps	760 pages
OC1	51.844 Mbps	880 pages
100-Mb Ethernet	80 Mbps (best case)	(Up to) 1,360 pages
OC3	155.532 Mbps	2,650 pages
OC12	622.128 Mbps	10,580 pages
1-Gbps Ethernet	800 Mbps (best case)	(Up to) 13,600 pages

If you add a small graphic to the 5-KB page, the results will be considerably different. An image, in the form of a .jpg file that appears on screen as, perhaps, a 1-by-2-inch rectangle (the actual physical size depends on monitor settings), takes up about as much disk space as the original text file. Adding one such image file to each page nearly doubles the average page size. This increased size reduces the number of page requests that the server can send to the Internet on a DS1/T1 line to a maximum of about 15 pages per second, regardless of how fast the computer itself runs. If there are several images on each page, if the images are relatively large, or if the pages contain other multimedia content, they will take considerably longer to download.

Given a page of moderate size and complexity, there are several ways to serve more pages per second:

- Remove some images from the page.
- Use smaller pictures if you currently send large ones, or compress the existing pictures (if they are already compressed, compress them further).
- Offer reduced-size images with links to larger ones, and let the user choose; use images of a file type that is inherently compact, such as a .gif or a .jpg file, to replace inherently large file types such as a .tif.
- Connect to the network by using a faster interface.

The last option resolves the issue at the server but not necessarily at the client, as will be shown later in this section.

A site that serves primarily static HTML pages, especially those with simple structure, is likely to run out of network bandwidth before it runs out of processing power. On the other hand, a site that performs a lot of dynamic page generation, or that acts as a transaction or database server, uses more processor cycles and can create bottlenecks in its processor, memory, disk, or network. There are no hard and fast rules that apply to all sites (one of the reasons why a comprehensive discussion of capacity planning is difficult), but the general relationship between bandwidth and CPU use for static versus dynamic pages is shown in Figure 4.1.

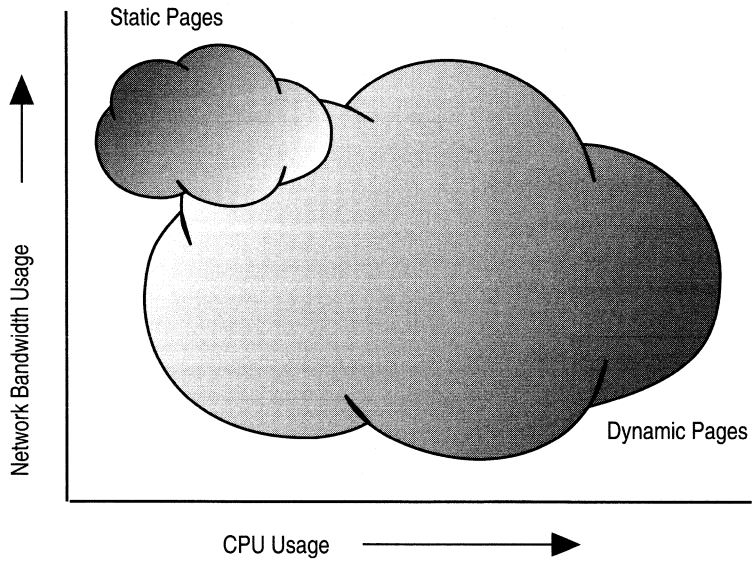


Figure 4.1 Relative Demands of Static vs. Dynamic Content for a Page of a Given Size

Browser Download Time

The number of pages a server can send is one half of the bandwidth equation. The other half is the time it takes a browser to download a page.

Consider how much time a browser needs to download a page that, including overhead, amounts to 90 KB or so—which equals about 720 kilobits. (Pages of this size are not at all unusual.) Ignoring latencies, which typically add a few seconds before any of the data arrives, it takes roughly 25 seconds to download 720 kilobits through a 28.8 kilobits per second (Kbps) connection if everything is working perfectly. On the other hand, if there's any blocking or bottlenecking going on at the server, if the network is overloaded and slow, or if the user's connection is slower than 28.8 Kbps (because of poor line quality, for example), the download will take longer.

If the client computer has a higher-bandwidth connection on an intranet, for example, the download time should be much shorter. If your Web site is on the Internet, however, you cannot count on a majority of users having faster connections until the next wave of connection technology becomes well-established. At the time of writing, a 56 Kbps modem standard has been adopted, but many (if not most) telephone lines are too noisy to allow full-speed connections with 56 Kbps modems. In addition, cable modem and Digital Subscriber Line (DSL) technologies are just beginning to appear in enough areas to compete in earnest. For this reason, it is not possible to tell which connection mode will take the upper hand or, for that matter, whether some other technology will appear and supersede both.

From the Server Side

It takes about 52 connections at 28.8 Kbps to saturate a DS1/T1 line. If no more than 52 clients simultaneously request the page used in the preceding example, and if the server can keep up with the requests, the clients will all receive the page in the 25 seconds calculated in the example (again, ignoring the typical delays).

If 100 clients simultaneously request that same page, however, the total number of bits to be transferred will be 100 times 737,280 (720 kilobits). It takes between 47 and 48 seconds for that many bits to travel down a DS1/T1 line. At that point the server's network connection is the limiting factor, not the client's.

Figure 4.2 shows the relationship between concurrent connections and saturation for DS1/T1 and DS3/T3 lines, assuming all clients are using a modem transmission speed of 28.8 Kbps and are always connected. A DS3/T3 line carries nearly 45 Mbps, about 30 times as much capacity as a DS1/T1 line, and it takes more than 1,500 clients at 28.8 Kbps to saturate its bandwidth. Moreover, the increase in download time for each new client is much smaller on a DS3/T3 line. When there are 2,000 simultaneous 28.8 Kbps connections, for example, it still takes less than 33 seconds for a client to download the page.

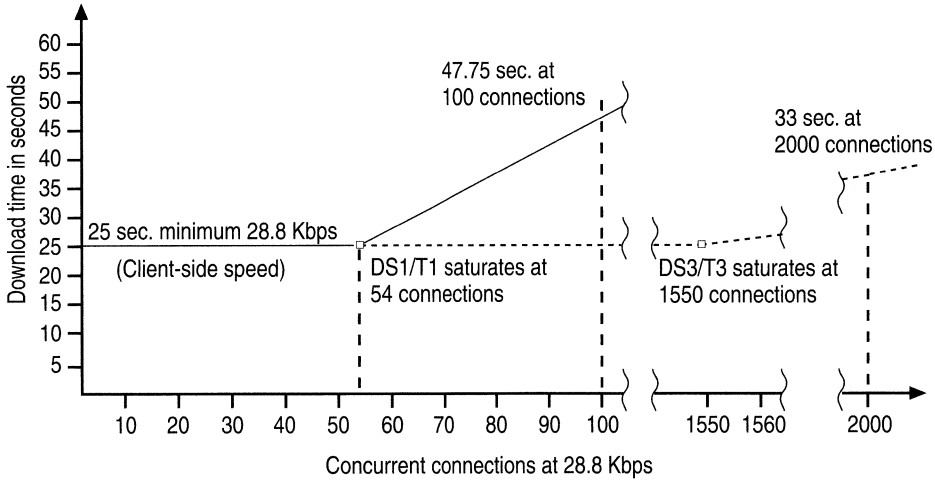


Figure 4.2 Download Time vs. Server Network Bandwidth

This example assumes that the server is capable of performing the relevant processing and handling of 2,000 simultaneous connections. That's not the same as handling 2,000 simultaneous users: Users occasionally stop to read or think, and typically spend only a modest percentage of their time downloading, except while receiving streaming multimedia content. Because of this difference between users and connections, the number of users that IIS 5.0 can support is larger than the figures would seem to indicate. A Web server on a DS1/T1 line can typically handle several hundred users connecting at 28.8 Kbps, and with a DS3/T3 line the number typically climbs to 5,000 or more. While these numbers are derived from actual servers, you can expect performance to vary with differing content types and user needs, and with the number and type of services being performed by a particular computer.

Essentially, the network performance differences shown here scale linearly, and the scaling continues at larger data-transfer rates. If you have two DS3/T3 lines, for example, you can serve approximately twice as many clients as you can with one, provided that you have enough processor power to keep up with user demand and that no bottlenecks prevent your servers from maximizing their processing power. For an example of such a bottleneck, see "A Large-Site Case Study: microsoft.com" later in this chapter.

Perceived Time

It is important to remember that the amount of time a user perceives that a page appears is not identical to the (measurable) amount of time it takes for the page to fully display. If the first thing the user sees upon reaching a given page is a set of buttons allowing further navigation, the user may never know or care that the rest of the page takes over a minute to download. If, on the other hand, the page takes over a minute to download and the navigation buttons don't appear until after the rest of the page, users probably won't bother to wait unless they are forced to by circumstances. An acceptable delay depends to some extent on the kind of information provided by the page, but it is ordinarily no more than 30 seconds. If the information is important, your users will be more likely to wait longer for it, albeit reluctantly.

Other Services and Activities

If your server is also acting as a Dynamic Host Configuration Protocol (DHCP), Windows Internet Name Service (WINS), or Domain Name System (DNS) server on a local area network (LAN), or as an e-mail and news group server, you must take into account the network bandwidth used by these services when planning your bandwidth budget (as well as your processing budget). If you anticipate heavy traffic, you should move these other services to a different server computer.

When testing new applications, particularly on a site that is actually in service, it's a good idea to run them out of process, despite the fact that this introduces additional computing overhead. For more information about running out-of-process applications and services, see "Data Access and Transactions" in this book.

Considerations

It is important to consider both server-side and client-side bandwidth when choosing your server's network connection(s). If your server connects only to an intranet, this question may be moot, except when planning a major upgrade; but if your site connects to the Internet, there are many possible types and speeds of network connection. The speed you require depends on the amount and type of traffic your site generates. The 5-KB static Web page discussed in the section on traffic earlier in this chapter is representative of many short, text-only pages, but relatively few Web pages contain only text. In the worst case, a page containing a substantial number of graphic elements can require one GET request per graphic. These requests add up quickly and have an impact on performance.

Web pages are increasingly being built as applications, and as a result are more processor-intensive. This by itself does not necessarily have much effect on bandwidth requirements. Content, however, does have an effect. Streaming multimedia, for example, is inherently bandwidth-intensive, and unless you can guarantee that your server will have only a small number of users on a high-speed intranet, you may find that spikes in user volume will easily overwhelm your network bandwidth capacity. Pages that perform extensive lookups in a database, on the other hand, may put a heavy load on the link from the Web server to the database server (and may place a really heavy load on the number of CPUs in the server computer for the database). However, these pages will not place much demand on the link from the Web server to the intranet or Internet, unless the returned datasets are quite large.

HTTP 1.1

IIS 5.0 automatically determines the size of any objects (such as text, files, or graphics) on a static page and the size of the page itself. When the client issues a GET request, the server uses a "Content-Length" header entry to report the size of the requested object or page. This HTTP 1.1 header entry has minimal cost in terms of overhead, and allows the browser to determine approximately how long the active connection will be used, thus affecting the browser's connection strategy. A browser such as Microsoft® Internet Explorer creates a new connection only when an existing one is "blocked." If the order of requests is such that larger .gif files are downloaded first, or if the connection speed is slow, more connections may be necessary.

Caching and the Refresh Process

In a refresh request, Internet Explorer (the client) tells the server the datestamp of the version of a file it has in its cache, using the "If-Modified-Since" header. IIS 5.0 then determines whether the file has been modified since that time. If it has not, IIS 5.0 replies with a brief "Not Modified" response.

A static HTML page is not retrieved during a screen refresh if it has not been updated. Some publication processes copy files that haven't been modified, which gives them new timestamps and thus "updates" them as far as the system is concerned. These files are downloaded even though they haven't really changed. When you set up your publication process, you should make every effort to avoid this waste of resources.

By default, IIS 5.0 sets HTTP cache-control to prevent browsers from caching processed page scripts in Active Server Pages (ASP), because there is no way to guarantee that an ASP page will be the same the next time it is requested. (IIS 5.0 caches compiled, ready-to-process scripts in ASP pages in its Script Engine Cache.)

Thus, under ordinary circumstances, just changing a file's extension from .htm to .asp, without putting any script on the page, causes a screen refresh to take longer. Because IIS 5.0 checks for this condition, the extra time is minimized but not entirely eliminated.

Secure Sockets Layer

When a browser makes a Secure Sockets Layer (SSL) request for a page, a delay occurs while the server encrypts the page. When the server sends the page, SSL adds additional network overhead. To enhance security, SSL also disables proxy and browser caching, so the considerable performance gains these allow are lost. The transaction time with SSL can be as much as a full order of magnitude longer. Finally, once the browser has received the requested files, the user must wait while they are decrypted, causing the download time to be even longer.

For these reasons, you should use SSL only when it is really necessary, such as when you need to ensure the security of financial transactions. Also, you should use it only on the specific pages to which it applies, rather than on an entire site.

Web Application Performance

If Web applications are an important part of your site, the performance of those applications is critical in determining the site's capacity. Testing is the only way to find out the capacity and performance of a Web application. The Web Capacity Analysis Tool (WCAT) and Web Application Stress Tool utilities included on the *Microsoft® Windows® Server Resource Kit* companion CD are useful testing tools. Before writing an application, however, it's useful to have a sense of the performance capabilities of different Web application types. In IIS 5.0, Internet Server Application Programming Interface (ISAPI) applications running as in-process dynamic-link libraries (DLLs) generally offer the best performance. Next best are ASP pages, followed by Common Gateway Interface (CGI) applications.

For most applications, the recommendation is to use scripting in ASP pages to call server-side components. This strategy offers performance comparable to ISAPI performance, with the advantage of more rapid development time and easier maintenance. For more information about Web application development strategy, see "Developing Web Applications" in this book.

It would be helpful here to look at some comparative performance data. Tests compared uniprocessor and multiprocessor performance for several tasks that were designed to be as similar as possible. These tests also measured and compared the costs of ISAPI, CGI, and static content, and ASP. In the test, each program takes a URL passed in the query string, maps it to a physical file on the server, and sends back that file. (For static content the URL is passed directly to the server, which responds normally.)

The ISAPI version achieves most of its speed by having much less overhead (and functionality) than the others; it uses the *ServerSupportFunction(HSE_REQ_TRANSMIT_FILE)* function. Even though the CGI version is nearly a line-for-line copy of the ISAPI version, it is slower because a new process must be started each time the CGI program is executed. Static content is fastest of all because IIS 5.0 is highly optimized for transmitting static content. In-process ISAPI and ASP applications execute faster than (pooled) out-of-process applications. Note that the test programs, while equivalent, are idiomatic. That is, each uses the best and most natural choice of tools from its framework. The ISAPI uses TransmitFile, which is extremely fast but is not available to ASPs or CGIs.

The following code sample is the ASP version, stripped of error handling:

```
<% @ EnableSessionState = false %>
<% ' Usage: http://server/test/sendfile.asp?test/sample/somefile.txt
   strPhysicalFilename = Server.MapPath(Request.QueryString)
   Set oFS = Server.CreateObject("Scripting.FileSystemObject")
   Set oFile = oFS.OpenTextFile(strPhysicalFilename)
   Response.ContentType = "text/plain"
   Response.Buffer = true'Default in IIS 5.0, but not in IIS 4.0.
   Response.Write oFile.ReadAll
   oFile.Close
   Response.End
%>
```

The entire Sendfile Test Suite is included on the Resource Kit companion CD, with instructions on setting up and running the tests.

The scripts and executables were tested with Beta 3 Release of IIS 5.0 and Microsoft® Windows® 2000 Server, using WCAT and the Web Application Stress Tool. The tests were run for 60 seconds, with 10 seconds of warm-up time and 10 seconds of cool-down time.

Table 4.3 shows the performance of the different application types, running on uniprocessor and multiprocessor kernels. Table 4.4 shows the hardware and software used for the tests.

The figures given in Table 4.3 are the actual numbers of pages per second that were served during testing, and in that narrow sense they are absolute. In several other senses, however, they are relative. First, the software being tested was not the final release version. Second, different computer types will, of course, provide different performance for the same test. Finally, the performance of different application types depends greatly on the application's task. Because this particular task is a relatively light load, it maximizes the differences among the various methods. Heavier tasks result in performances differences that are smaller than those reflected in the table.

Table 4.3 Performance of IIS 5.0

Test	Non-SSL 1 CPU	Non-SSL 2 CPUs	Non-SSL 4 CPUs	SSL 1 CPU	SSL 2 CPUs	SSL 4 CPUs
ISAPI In-Process	517	723	953	50	79	113
ISAPI Out-of-Process	224	244	283	48	76	95
CGI	46	59	75	29	33	42
Static file (file8k.txt)	1109	1748	2242	48	80	108
ASP In-Process	60	107	153	38	59	83
ASP Out-Of-Process	50	82	109	28	43	56

All numbers denote requests per second. Each test repeatedly fetched the same 8-KB file.

Table 4.4 Setup for the Performance Test

Server	Compaq Proliant 6500 (4 x Pentium Pro 200MHz) with 512 MB of 60ns RAM
Clients	16 Gateway Pentium II machines, 350MHz with 64 MB RAM
Network	Each client: one Intel Pro100+ 10/100MB network adapter Server: four Intel Pro100+ 10/100MB network adapters Four separate networks were created to distribute the workload evenly for the server, with four clients per network. Two Cisco Catalyst 2900 switches were used, each having two Virtual LANs (VLANs) programmed.
Software	Server: Windows 2000 Beta 3 Advanced Server (Build 2031), IIS 5.0 Clients: Windows 2000 Professional (Build 2000.3) Testing: Web Application Stress Tool (Build 236)

Note These tests were conducted with "out-of-the-box" computers and programs. No additional registry changes or performance enhancements were administered.

Multiprocessor Scaling

IIS scaling to multiple processors is improving, but when all processors in one computer must share the system bus and other resources, symmetric multiprocessor (SMP) scaling simply cannot map 1:1 with the number of processors. Multiple system buses can help with this issue; but for computers in which all processors share a single bus, a pair of two-processor machines can provide more throughput than a single four-processor machine, and may have a similar price tag.

Reliability

Some sites can afford to fail or go offline; others cannot. Many financial institutions, for example, require 99.999 percent or better reliability. Site availability takes two forms: The site might need to remain available if one of its servers crashes, and it might need to remain online while information is being updated or backed up.

Even if your requirements are less rigorous than those of a major financial institution, you will probably want to use Redundant Arrays of Independent Disks (RAID). You can also consider creating a "Web farm" with Network Load Balancing; in addition, you can create subsystem redundancy by clustering component servers.

Server Clustering

The term "failure" commonly brings to mind the idea of a system crash, but in fact many system failures are deliberate: the administrator brings a server down for routine maintenance or for hardware installation or an upgrade. Clustering makes it possible to take a server down for maintenance or service without causing the site itself to fail, and also provides reliability in the event of an unscheduled failure.

Microsoft supports two clustering solutions. The first of these is Windows 2000 Server clustering; the second is Network Load Balancing. The next two sections describe them.

Windows 2000 Server Clustering

With Windows 2000 Server clustering you can set up applications on two servers (nodes) in a cluster and provide a single, virtual image of the cluster to clients, as shown in Figure 4.3. If one node fails, the applications on the failed node become available on the other node. That is, the actual content and applications are shared so that both machines have full access to them. Failover times range from 20 seconds to 2 minutes.

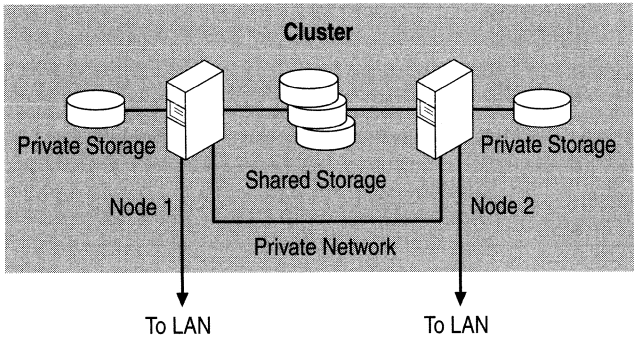


Figure 4.3 A Windows 2000 Server Cluster

Clustering is available with Windows 2000 Advanced Server. To use the Windows 2000 clustering feature, you must have two servers that are connected by a high-speed private network. Each of the servers must have at least one shared Small Computer Systems Interface (SCSI) bus, with a storage device connected to both servers, and at least one storage device that is not shared. For the most reliability, each computer should have its own uninterruptible power supply.

Windows 2000 Server clustering provides mirroring with rapid failover, and is an excellent way to ensure reliability on, for example, a Microsoft® SQL Server™ connected to either a single Web server or a group.

Network Load Balancing

The Network Load Balancing feature of Windows 2000 Advanced Server allows you to create server clusters containing up to 32 machines. Network Load Balancing is fully distributed, and is entirely software-based; it does not use a centralized dispatcher, and does not require any specialized hardware.

This feature transparently distributes client requests among the hosts in the cluster, using virtual Internet Protocol (IP) addresses. You must run IIS 5.0 or another TCP/IP service on each host, and the hosts must serve the same content so that any of them can handle any request. You can copy updated pages to local disks on the hosts, or use commercial file-replication software to perform updates. Network Load Balancing allows you the option of specifying that all connections from the same client IP address be handled by a particular server (unless, of course, that server fails). It also permits you to allocate all requests from a Class C address range to a single server. In addition, Network Load Balancing supports SSL.

DNS Round-Robin Distribution

Domain Name System (DNS) round-robin distribution is an earlier and less sophisticated technique for allocating requests among servers in a group. Consider a scenario in which there are four IP address entries for the same host on a DNS server:

```
domain.com A 172.17.21.31
domain.com A 172.17.21.35
domain.com A 172.17.28.41
domain.com A 172.17.28.52
```

If a client sends a query, the DNS server returns all four IP addresses, but typically the client uses only the first one it receives. With round-robin distribution, the next time the DNS server receives a query for this host the order of the list is changed in a cyclic permutation or "round-robin" (the address that was first in the previous list is last in the new list). Thus, when the client chooses the first IP address in the list, it connects to a different server.

This technique distributes incoming requests evenly among the available IP addresses, but does not fully balance the load because it is not interactive. That is to say, the DNS server neither checks on the loading of an IP address, nor whether a particular server is currently running. Nonetheless, round-robin distribution can be a useful starting point or a low-cost alternative for small groups of servers. You should, if you are using round-robin distribution, keep close tabs on your servers so that you can quickly remove any failed machines from the distribution list.

Determining Your Installation's Requirements

If you are building a new server site, you must develop a sense of its requirements in order to decide what hardware and software to acquire. Consider what the site will be required to do, and what services it will provide. Will it:

- Serve a known set of clients on an intranet or intranet/extranet connection to the Web? (That is, can you predict the number of clients with any accuracy, and is it possible to get a good sense of their usage behavior?)
- Allow people to download and possibly upload files with File Transfer Protocol (FTP) service?
- Provide e-mail with Simple Mail Transfer Protocol (SMTP) service?
- Provide news with Network News Transfer Protocol (NNTP) service, which is typically coupled with large storage requirements?
- Allow clients to access a database or large index using SQL or other query language, or with Indexing Service? Allowing this kind of access will entail large storage requirements for the database itself.
- Perform transactions using Microsoft® Component Services, which is processor-intensive?
- Provide name service?
- Function as a domain controller?
- Provide file and print service?

As discussed earlier, there are also page construction and Web application issues. Will your server or servers primarily send out static HTML pages? (Doing so involves memory for caching files.) Will the server run scripts in ASP pages (ASP service involves queue-length optimization issues as well as cache space), ISAPI, and CGI applications? (ASP, ISAPI, and CGI are all processor-intensive, and CGI applications are inefficient under Windows 2000 Server.) Do you need to run some of your pages out of process, for testing or other purposes? (This is also processor-intensive.) In addition, you must figure in the constraints of budget, facilities, and staffing.

If you can find a site with requirements that are similar to yours and examine its history, you may be able to discover (and possibly avoid) some of these potential pitfalls.

If you are working with an existing site, you can monitor the site server and log its performance. This will give you a sense of current conditions and will let you see how well the existing hardware and software are meeting current user needs. (See "Monitoring and Tuning Your Server" in this book.)

A Capacity Planning Checklist

This checklist provides an in-depth look at your site and helps you anticipate where bottlenecks are likely to occur.

Purpose or Type of Site

First, decide whether your site will be transactional; that is, determine whether customers will be retrieving and storing information, typically in a database. A transactional site involves both reliability and security requirements that do not apply to most nontransactional sites.

Complexity Level

Next, consider whether the content on your site will be static. A site that makes use of SQL, ASP, ISAPI, CGI, or multimedia requires much more processing capability than a static site.

Customer Base

Consider the size of your customer base and its potential for expansion, on at least the following time scales:

- Current to one month
- Six months
- One year

Finding Potential Bottlenecks

Find out what is likely to break first. Unless your site is extremely small, you'll need a test lab to determine that. (There are suggestions for building and using such a lab in the following list.)

To determine potential bottlenecks

1. Draw a block diagram showing all paths into the site. Include, for example, links to FTP download sites as well as other URLs.
2. Determine what machine hosts each functional component (database, mail, FTP, and so on).
3. Draw a network model of the site and the connections to its environment. Define the topography throughout. Identify slow links.
4. For each page, create a user profile that answers the following questions:
   - How long does the user stay on the page?
   - What data gets passed to (or by) the page?
   - How much database activity (or other activity) does the page generate?

- What objects live on each page? How system-friendly are they? (That is, how heavily do they load the system's resources? If they fail, do they do so without crashing other objects or applications?)
- What is the threading model of each object? (The "agile" model, in which objects are specified as both-threaded, is typically preferable, and is the only appropriate choice for application scope.)

5. Define which objects are server-side and which are client-side.
6. Build a lab. You'll need at least two computers, because if you run all the pieces of WCAT on one computer, your results will be skewed by WCAT's own use of system resources. Monitor the performance counters at 1-second intervals. When ASP service fails it does so abruptly, and an interval of 10 or 15 seconds is likely to be too long—you'll miss the crucial action. Relevant counters include CPU utilization, Pool nonpaged bytes, connections/sec, and so on. (For more information about counters, see "Monitoring and Tuning Your Server" in this book.)
7. Throw traffic at each object, or at a simple page that calls the object, until the object or the server fails. Look for:
   - Memory leaks (steady decrease in pool nonpaged bytes and pool paged bytes).
   - Stop errors and Dr. Watsons.
   - Inetinfo failures and failures recorded in the Windows® Event Log.
8. Increase the loading until you observe a failure; document *both* the failure itself and the maximum number of connections per second you achieve before the system tips over and fails.
9. Go back to the logical block diagram, and under each block fill in the amount of time and resources each object uses. This tells you which object is most likely to limit your site, presuming you know how heavily each one will actually be used by clients. Change the limiting object to make it more efficient if you can, unless it is seldom used and well off the main path.
10. Next, traceroute among all points on the network. Clearly, you can't traceroute the entire Internet; but you can certainly examine a reasonable number of paths between your clients and your server(s). If you are operating only on an intranet, traceroute from your desk to the server. This gives you a ballpark estimate of the routing latencies, which add to the resolution time of each page. Based on this information, you can set your ASP queue and Open Database Connectivity (ODBC) timeouts. (For more information about ASP queuing, see "Monitoring and Tuning Your Server" in this book.)

Note If the first seven steps appear to bear some resemblance to an inverted version of the Open Systems Interconnect (ISO) "layer cake" model, there's a reason. The ISO model is a highly useful lens through which to examine server behavior.

Network Bandwidth

Once you determine how many customers/clients you want to serve during a given time period, you have the lower limit for your network connection bandwidth. You need to accommodate both normal load and usage spikes; you can average these to get a reasonable estimation of network capacity.

Of course, the type of site you operate has a large effect on this issue. For example, if you are largely or entirely subscriber-based, or if your site is only on an intranet or an intranet/extranet combination, you probably already have a good idea of the maximum spike size. If, on the other hand, you issue software revisions to an audience of unknown size on the Web, there may not be a good way to predict the size of resulting spikes. You might, in fact, have to measure one or more actual occurrences to decide whether your bandwidth is sufficient.

A Note for Internet Service Providers

It's probably a good idea to set some standards. Tell customers that certain (tested and approved) objects are permitted on your servers and that others (tested and failed, or untested) are not. If they want to use untested objects or others known to be bad, they can do so, but you will likely want them to have their own server computer(s). Provide them with a page on one of your servers that they can log into through a secure connection, in order to reset their server remotely if it goes down.

Capacity Planning Scenarios

While all Internet sites differ in how they are implemented, they do follow standard capacity planning profiles that can be categorized into the following scenarios:

- The Intranet Web Site
- The Internet Marketing Web Site
- The Internet Transactional Web Site
- The Internet Commerce Web Site

It is important to understand that while the technology within IIS 5.0 is the same for each scenario, there are external technical and business decisions that influence the way a Web site is deployed and then expanded.

The Intranet Web Site

Many of today's Web sites exist only within corporations, and are not connected directly to the Internet. This scenario addresses the issues involved in establishing corporate intranets.

Purposes

Frequently, intranets are built to fulfill one or both of the following functions:

- Provide information to internal employees.
- Provide different groups with access to their own content areas.

Capacity Planning Issues

Each type of Web site has its own set of parameters and issues. Here are some issues that are specific to intranet sites:

- This site is usually put up on existing hardware and, as it becomes more valuable to the company, quickly expands to the maximum limitation of the platform.
- The site usually offers any kind of content that management requires.
- The site may be called upon to offer downloads or custom-built components as necessary.

Initial Model

As shown in Figure 4.4, the initial model for an intranet site consists of a single server running IIS 5.0, usually with FTP service enabled so that departments can add their own information as necessary.

Figure 4.4 Initial Model for an Intranet Site

Growth Models

The main options for expanding an intranet site fall into two categories:

1. Content Striping

   With this option, multiple servers have the same content installed and a load-balancing algorithm (such as Network Load Balancing) is implemented to serve multiple customers. This requires that the content be updated equally on all servers at once.

   This option is good when there is no clear organizational division among the components.

2. Content Specialization

   With this option, content or components are separated on dedicated servers. The user typically engages a primary intranet server, and is redirected to a dedicated server if necessary.

   This option is usually good when there is a clear organizational division for a particular set of content. It is also a good choice when resource-intensive content will impact the performance of the entire site if it is not segregated. When combined with content striping, this option is also good for extremely large sites.

Suggested Capacity Planning Methods

The following list presents some key issues for planning an intranet Web site:

1. Determine the site's audience.
2. Determine the type of information the site will offer (static content, active content, file downloads, and so on).
3. Determine additional services (newsgroups, chat, knowledge management) the site will host.
4. Based on this information, install hardware that will, in its *minimum* configuration, meet the current needs of the site. (This allows for maximum growth potential before expansion of the site is necessary.)
5. As the site grows, use content striping to handle increased customer requests or the decrease in Web server performance that comes with more complex Web content.
6. As individual departments request that the site host more information, you can implement content specialization (that is, install more servers and divide the content among them). Content specialization will allow those departments to meet their requirements without impacting the overall performance of the site.

The model for a larger site would look something like the one shown in Figure 4.5:

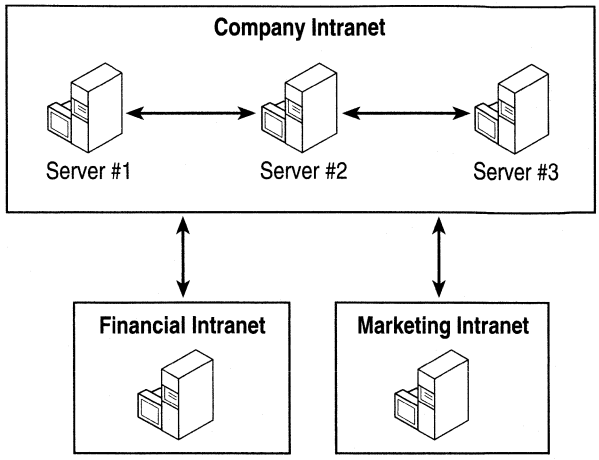


Figure 4.5 Mature Model for an Intranet Site

The Internet Marketing Web Site

An increasing number of sites conduct marketing (as opposed to commerce) on the Internet. A typical marketing site is relatively straightforward, involving HTML or simple ASP pages and graphics. It does not usually require an extensive database or much input from customers. This section discusses the issues involved in setting up such a site.

Purposes

A marketing site is likely to fulfill one or more of the following needs:

- The company wants to have an Internet presence available to its current and potential customers.
- The company wants to provide an inexpensive alternative to catalog shopping by showing customers its latest product offerings.
- The company wants to allow customers who have Internet access to provide feedback on products and company business practices.

Capacity Planning Issues

The following issues are commonly encountered in the process of developing a marketing site:

- The company has no initial idea of the number of customers it can expect to access its site.
- The company has no gauge of what speed or from where customers will be connecting.

Initial Model

As shown in Figure 4.6, the initial model for an Internet marketing site consists of a single server running IIS 5.0, usually with FTP service enabled (to allow internal divisions to add their own information as necessary), as well as some e-mail functionality for customer feedback.

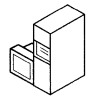


Figure 4.6 Initial Model for an Internet Marketing Site

Growth Models

The growth models for an Internet marketing site are the same as those for a corporate intranet site. Both include content striping and specialization. However, instead of separating the site's content by organizational unit, it is useful to segregate content either by functionality (Web, e-mail, newsgroups, and so on) or by product line. A sports products company, for example, would probably list gear for summer sports on one set of pages, and gear for winter sports on another.

Suggested Capacity Planning Methods

Here are some guidelines for capacity planning for a marketing site that does not already have a large customer base:

1. Put the initial Web server on a medium-size platform.
2. Examine the IIS 5.0 performance logs to determine how heavily the server's resources are loaded by incoming traffic.
3. Examine the IIS 5.0 server logs to determine the number of customer visits the site experienced.
4. Using the results of steps 2 and 3, plot out the growth metrics of the site, with special emphasis on periods when the site was being marketed heavily or was more visible to Internet customers than usual (for example, just prior to a new product launch).
5. As the Web server's resource utilization level reaches approximately 70 percent, use either content striping or specialization to add additional capacity to the site.

The Internet Transactional Web Site

The term *transactional*, as it is used here, indicates a bidirectional flow of data, typically to and from a database. (It does not indicate e-commerce transactions.) Customers need to be able not only to look up information already stored at the site, but also to add new information of their own. Transactional sites tend to use Web applications that are more complex and extensive than those of marketing sites, and thus require careful development and testing.

Purposes

A transactional site is likely to fulfill one or more of the following needs:

- The company wants to communicate with its customers online.
- The company wants to personalize its site to better meet the needs of its customers.
- The company wants to provide a forum to help online customers with complaints and issues.

Capacity Planning Issues

There are two main issues involved in setting up a transactional site:

- The site will almost always rely on a database, which gathers customer information. But usually the database cannot be replicated in the same way that content can be striped. As a result, adding Web servers as the site grows will increase the traffic load on the database server.
- As the amount of data and traffic grows with an increasing number of customers, additional components need to be implemented to deal with customer connection issues. Some of these issues include ODBC time-outs, WINS errors, and server failures. In fact, connection issues can encompass anything that keeps a customer from putting information into the database or retrieving information from it. The additional overhead of these components has effects on the overall performance of the site, which must be taken into account as the site expands.

Initial Model

As shown in Figure 4.7, the initial model for a transactional Internet site consists of a single server running IIS 5.0, usually with FTP service enabled (to allow divisions to add their own information as necessary), as well as some e-mail functionality for customer feedback. In addition, the Web server has some connectivity (such as ODBC or OLE DB) that allows it to offer dynamic content and to store customer information. Typically, the Web server is connected to a separate database server.

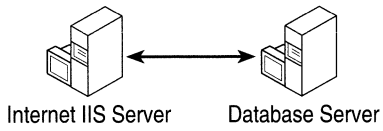


Figure 4.7 Initial Model for an Internet Transactional Site

Growth Model

This site can use the same options as the Internet marketing site, with the additional option of dividing the database server according to whether the data is effectively static (comes from the company) or dynamic (is entered by the customer). Separating the static and dynamic databases provides optimum performance on the Web servers, without the complex code necessary to handle data redundancy and multimastering issues.

Suggested Capacity Planning Methods

A transactional site has the potential for moderately heavy traffic, both outgoing and incoming, and requires more capacity at the outset than is required for either of the previous scenarios. Here are some guidelines:

1. Put the initial Web server on a high-end platform.
2. Examine the IIS 5.0 performance logs to determine how heavily used the server's resources are. Focus on CPU utilization, ASP requests per second (if you are using ASP), and memory paging.
3. Examine the IIS 5.0 server logs to determine the number of customer visits the site experienced.
4. Examine the logs of the database server to determine the average and maximum numbers of database connections.
5. Using the results of steps 2 and 3, plot out the growth metrics of the site. As with the marketing site, put special emphasis on periods when the site was more visible to customers than usual.
6. As use of the Web server's resources reaches approximately 70 percent, employ either content striping or specialization to add additional capacity to the site.
7. Using the results of step 4, determine when the database server is close to bottlenecking either from additional customer database requests, or from database requests that have become more complex. Use more server resources, and segregate the database as necessary.

The model for a transactional site that has grown to accommodate increased traffic looks something like the one shown in Figure 4.8:

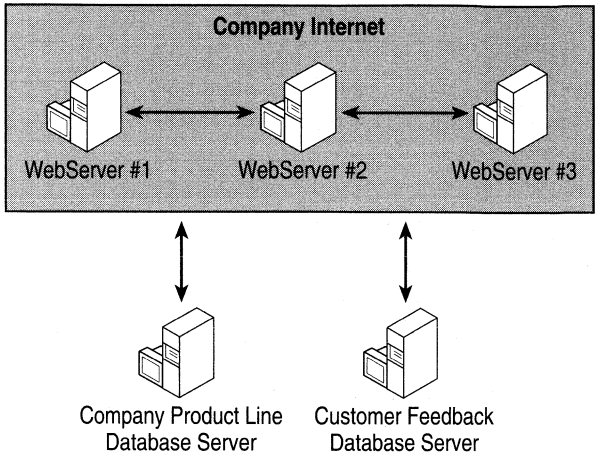


Figure 4.8 Mature Model for an Internet Transactional Site

The Internet Commerce Web Site

In 1998, e-commerce was the most rapidly expanding sector of activity on the Web. There is no reason to think that this will change in the immediate future.

Purposes

Internet commerce is complex. It involves several kinds of data that must be handled in appropriate ways at the right time. Some of the controlling considerations for an Internet commerce site include:

- The company wants to sell products or services to customers via the Internet.
- The company requires that the customer transactions be both secure and reliable.
- The site will usually have data feeds and batch jobs between the site and back-end systems (accounts receivable and payable, inventory, fulfillment vendors, and so on).

Capacity Planning Issues

The Internet commerce site has special requirements that affect capacity planning in specific ways. These involve performance, security, and availability:

- The site, because of its "real-time" nature, must be designed to handle the maximum number of concurrent customer connections that can be expected, with a good safety margin so that it can smoothly handle spikes in usage.
- The site designer must implement a high level of security and potentially some form of data encryption. This can easily add 15 to 20 percent in system resource overhead to each transaction.
- It is possible for the site's performance to be constrained by a downstream resource. For example, the Web server that handles catalog content may be able to handle 200 transactions per second; but if the sales tax or credit card components can only handle 50 transactions per second, there is a potential for a serious bottleneck to develop. It is crucial for site designers and administrators to be aware of these external constraints where they occur.

Initial Model

The initial model for this site consists of multiple servers running IIS 5.0 with some database connectivity enabled. Other components, such as certificate servers, tax, and credit card validation are common.

Growth Model

This type of site can expect growth from both increased customer numbers and additional online commerce offerings on its Web server(s). These factors are, to some extent, multiplicative, and can over time cause exponential increases in server loading. Another key to this type of site is the fact that while the physical hardware of the site itself may support a certain performance rating, very often the components that interact with legacy and third-party systems are much slower.

Suggested Capacity Planning Methods

Designing a site for commerce is more complex and difficult than designing for the other scenarios presented here. Here are some guidelines:

1. Prior to purchasing any equipment, create a logical diagram of the site and identify all components and major systems with which the Web site must interact.
2. Determine how many transactions per second the Web site will be expected to handle during its peak traffic time.
3. Using the information from steps 1 and 2, determine which components will be the most frequently used and rate them in terms of their use of server resources. (It is probably sufficient, at this stage, to focus on three parameters: connections per second, CPU utilization, and memory utilization.)

4. As the components are installed, test each of them by running simulated traffic into the component and recording how many transactions per second it can handle. At the same time, observe (using PerfMon, which is a performance monitoring tool that measures memory use system-wide) how much of the system's resources are used by each component.

5. Using the information from steps 3 and 4, create a table similar to Table 4.5 (the numbers in this example are arbitrary):

Table 4.5 Transaction Component Performance Comparison

Component Name	Component "Rating"	Peak Transactions Expected	Maximum Transactions per Server	Server Resources Used per Transaction
Sales Tax	1	200	70	1.4%
Inventory	2	150	120	0.23%
Customer Service	3	80	100	0.45%

6. Using the table you generated in step 5, determine which components are the most resource-intensive, and whether any will fail to deliver the peak transaction numbers expected, on a per-server basis.

7. Multiply the maximum simultaneous transactions per server by the server resources utilized per transaction to find the sum result for all components that run on the same server. Any total that exceeds about 70 percent indicates that you should either content stripe by adding servers, separate components onto their own servers, or both.

8. As the Web site is deployed, analyze periodic PerfMon performance reports on metrics like connections per second, memory, and CPU utilization to determine how accurate your chart is.

9. Update the chart as necessary with production data and add additional servers (using either component striping or specialization) to maintain the Web site, so that the peak transactions per second will only use approximately 70 percent of the resources of any one server computer.

A Large-Site Case Study: microsoft.com

At the time of writing, microsoft.com, one of the largest and most active sites on the Web, was receiving well over 200 million hits per day and hosting more than 24 gigabytes (GB) of content. Building a high-traffic Web site of this kind involves careful planning and constant monitoring in order to achieve a balance between user demand and the site's key components: hardware, software, and network infrastructure. These components need to be in balance to efficiently handle the content on the site, the number of hits to the site, and spikes in usage.

Despite the size of microsoft.com and the special needs that follow from its size, the processes that the Microsoft team follows in order to plan, deploy, and maintain the site are relevant to many other sites. These processes are worth examining, even if your Web server is only connected to a small intranet.

A Snapshot of the Site

The microsoft.com site started in 1994 as a single computer under a developer's desk. It sustained as many as a million hits a day. Today, the site is one of the largest on the Internet. It has far more server hardware and bandwidth than necessary for day-to-day use, but all available capacity is needed for the inevitable spikes that occur when thousands of users concurrently download, register, or participate in some type of online activity, such as with a major product release.

Figure 4.9 shows the site, with its hardware and internal connectivity:

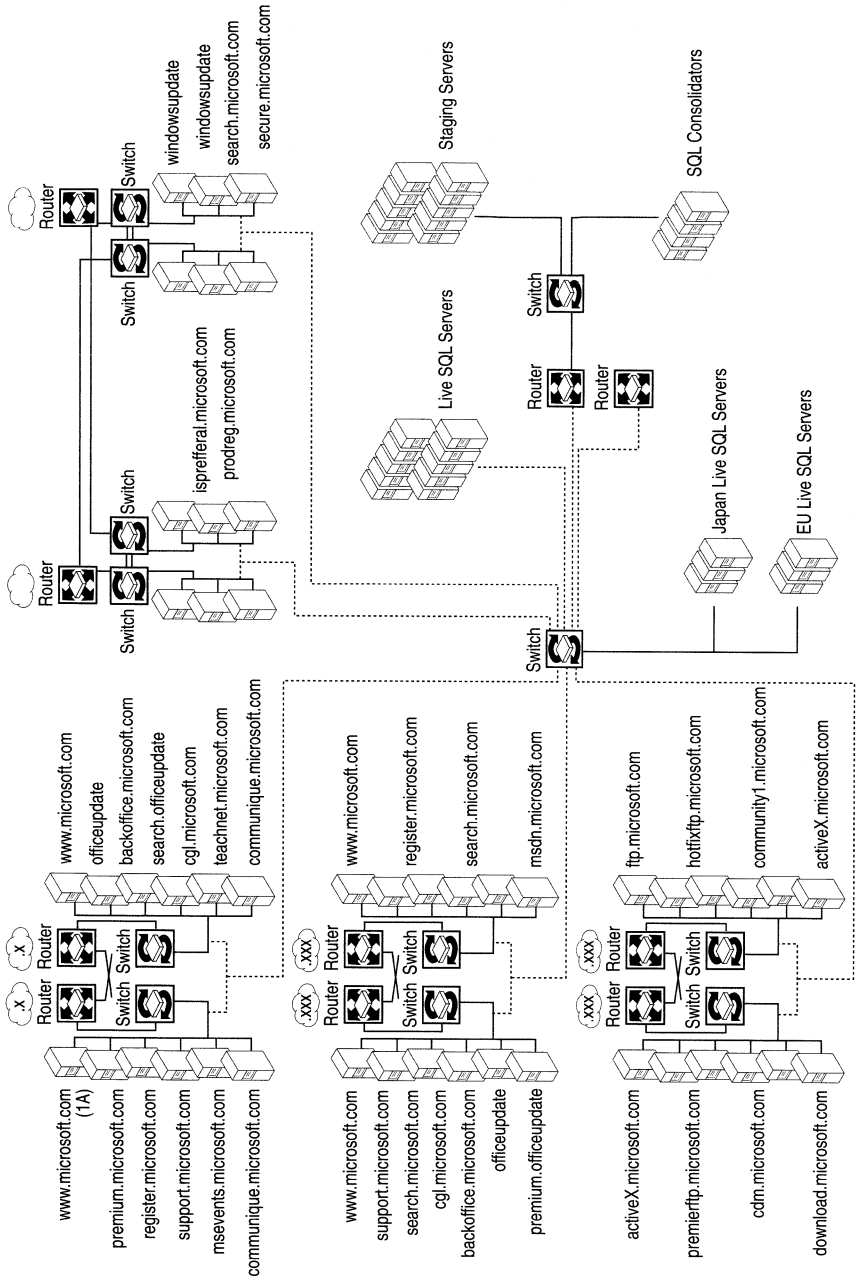


Figure 4.9 The microsoft.com Web Site

Current statistics and resources include:

Traffic

- 43 million page views per day.
- 240 million hits per day.
- 2.5 million users per day.
- 15 million users per month.
- More than 6 GB of successful downloads per day.

Growth

- 115 percent increase in page views between July 1997 and July 1998.
- 14 percent increase in users during that twelve-month period.

Content

- 24 GB of content.
- 324,000 HTML or ASP files.
- 307,000 .gif and .jpg files.
- About 18 GB of files available for download.
- Content updated every three hours worldwide.

Hardware

- Over 100 servers for all parts of the site: Compaq Proliant 5000s, 5500s, and 6500s, each with four Pentium Pro processors and 512 megabytes (MB) of RAM. (There are also 30 more servers at other U.S. locations, as well as overseas mirror sites.)
- Six internal Ethernets provide 100 megabits per second (Mbps) of capacity each.
- 2 OCI2 SONET fiber optic lines provide 1.2 gigabits per second of capacity to the Internet.

Software

- Microsoft® Windows® NT Server 4.0.
- IIS 4.0.
- Microsoft® Site Server 3.0.
- Microsoft® SQL Server 7.0.
- Other Microsoft tools and applications.

Planning for Spikes

The site's involvement in the Microsoft® Windows® 95 launch led to a surge of activity inside Microsoft, as product and marketing groups added content to it. During 1996, the number of hits per month on the site grew from 118 million to over *2 billion* (it is now considerably higher). The product groups' increased focus on Web-based marketing has fed the ramp-up, as additional users access more content on the site.

Beyond continuing growth and regular periods of peak usage, irregular spikes and special events place much larger burdens on many Web sites and servers. For Microsoft, many of these coincide with software product releases. The release of Microsoft® Internet Explorer 4.0 in October of 1997 is a good example: In one week, more than 2 million users downloaded an average of 18 MB each. Peak usage exceeded 6 terabytes per day.

While it is true that Microsoft's site has some special requirements, companies of every size need to plan for spikes, which can occur when they start a new ad campaign, appear in a news article, or are linked to a popular Web site. Even a single Web server connected only to a small LAN can experience spikes in activity. However, if the server supports only a small number of clients and their needs are not urgent, the impact won't be catastrophic. At the same time, if it is important to the enterprise that even a few of those clients be able to access the Web server without delay under any circumstance, or if it is crucial that the site be available all the time, the site manager make sure that it has the necessary capacity and reliability.

When Planning Is Not Enough

Spikes are not the only pitfalls. The continued expansion of day-to-day operations and the ongoing change in orientation from static HTML to dynamically generated pages and increased interactivity led to unforeseen difficulties at microsoft.com. Toward the end of 1996, for example, problems began to emerge at http://home.microsoft.com as users found error messages appearing on their browsers. By May of 1997, the servers had begun to show signs of blocking, as thousands of delayed access requests backed up. Performance of the Internet Start page went downhill gradually for two or three weeks, then suddenly took a nosedive. The site's existing hardware and software technology was simply unable to keep pace with demand.

Working Toward a Resolution

Because the Internet Start page at http://home.microsoft.com is the default home page for the Internet Explorer browser, the hit rate at the site is continually increasing. At the time of this writing, each of the servers was handling between 2,000 and 4,000 viewers per second. In May of 1997, as access to http://home.microsoft.com became increasingly difficult, a task force was formed and given the charter to find a solution. The group met daily for six weeks, working to return the site to nominal performance.

The first step was to examine the hardware, beginning with the servers—Internet connection bandwidth was not saturated, and therefore could not be the central issue. With the number of viewers increasing rapidly, the servers were adding viewer requests to the queue faster than they were delivering responses. More servers were added to handle the load, but this alone was not sufficient. The task force began probing the connections among servers, databases, and viewers to find and eliminate possible signal breakdowns or bottlenecks. They also initiated a process of streamlining the site's HTML and ASP code to make it as lean and efficient as possible. By mid-June some progress had been made, but it was clear that something else was wrong.

A Case of External Dependency

The task force began to look at the objects—text files, graphics, and other files that make up individual Web pages—that were being called by the servers. Some of those objects are housed in the Properties Database (PD), which receives and stores individual user preferences: news headlines, personalized home-page options, and stock quotes, for example. Through a process of elimination, the task force came to suspect the Properties Database of being the main culprit behind the ongoing slowdowns and failures. All of the servers were drawing on the same central repositories or databases for page elements, like many wells drawing on a common aquifer. In other words, the problem was external to the servers themselves; adding more of them just made it worse, because doing so only increased the loading on the bottlenecked component.

Part of the solution was to cut down the number of requests to that particular set of databases. New, less demanding ways of personalizing pages were developed, and the server software was adjusted to work better for the rapidly increasing numbers of viewers using the site. In addition, the database servers were moved physically closer to the Web servers in order to allow the team to eliminate intervening routers and other hardware. Meanwhile, the team continued to examine the entire process of creating Web pages and of serving them to visitors.

A Case of Inefficient Access

A new Internet Explorer download page was deployed just before the release of Internet Explorer 4.0. This page was complex, and the path to it led through two other processor-intensive pages, one of which was the microsoft.com home page. Almost immediately the site began to suffer processor bottlenecks. Analysis showed that a large number of clients were following a single path into the site, straight to the Internet Explorer download page, and that there were serious inefficiencies along that path. Streamlining the ASP code was not sufficient to resolve the problem. Nor was adding hardware the answer: Site engineers estimated that even an order-of-magnitude increase in the number of servers wouldn't be enough to handle the demand. Instead, temporary measures were put into place:

- Pages involving frames, other than the site's home page, were reduced to single panels.
- The download page and both pages leading to it were temporarily redesigned using only static HTML, rather than as dynamic pages.

This solution demonstrates another way to deal with spikes: Reduce the load by decreasing functionality, thereby lessening overhead. Once the load returns to normal levels, the original functionality can be restored.

Finding a Balance

Building an Internet or intranet site requires balancing server hardware and software, content, and network resources, and then maintaining that balance as site traffic grows. It is important for site managers to be on the lookout for hidden assumptions and external dependencies that interfere with the balance. Constant monitoring keeps the balance in view.

The Hardware

Hardware is a necessary part of site capacity and performance, and it is important to select hardware that can be scaled or upgraded easily when demands increase. The microsoft.com team continually reviews its hardware to be sure that capacity is staying ahead of demand. The current CPU utilization for microsoft.com is as high as 40 percent per server; it is deliberately kept well below capacity so that the site can handle spikes.

The Software

In addition to the Microsoft software used to run the site, the microsoft.com team has developed a few internal content management utilities by using Microsoft® Visual Basic® and Microsoft® Access. For example, a content-tracking tool scans the site each day and fills a database with information about each of the 250,000 HTML and ASP pages, including where the links on each page point. When a page is to be deleted, a check of the database shows other pages that point to that page. Links on the other pages can be changed before the page is deleted, so someone viewing them won't find broken links.

Another document management tool keeps track of who is responsible for each page and when the page was sent to the servers. Even with thorough testing, problems—such as errors in ASP pages—sometimes get through and are difficult to pinpoint among 250,000 pages of content. The tool provides a list of pages sent to the servers shortly before the problem appeared, narrowing the number that have to be investigated.

The Network

In another move to increase network capacity at Microsoft, system engineers installed two network adapters in each server. One is for the Fiber Distributed Data Interface (FDDI) ring that carries Internet traffic, and the other is for the corporate network, which handles administration and content replication. With this arrangement, administrative traffic doesn't take bandwidth away from Internet traffic.

The Content

Just as the microsoft.com team constantly reviews its hardware capacity to ensure quick response for the user, it also studies the way it configures its servers to manage content. Currently, the content is updated eight times a day, with 5 to 7 percent of the total content changing each day. Because each of the microsoft.com Web servers contains a complete copy of the site's content, each set of changes has to be replicated to every server. The team is now using Network Load Balancing to accomplish this.

Ongoing Changes

Groups producing content for microsoft.com want to use the latest Web publishing features to make their content more eye-catching and interesting to users. The result is a tug-of-war between content providers, who want users to have the richest experience possible, and the team running the Web site, which wants to increase download speeds and the site's capacity. The site currently has a size limit of 100 KB per page, including graphics, but preferably each page should be smaller than 60 KB. The bottom line is that content needs to strike a balance between being visually interesting and being as small as possible in order to improve download performance.

The User Experience Counts

The amount of content a site delivers does not necessarily indicate user satisfaction. To monitor the user download experience, the microsoft.com team periodically tests the access, search, and download times for specific pages, using 28.8 Kbps modems stationed in various U.S. cities. The test that was performed in January of 1997, showed that users averaged 50 seconds per task. A more recent test of download times indicated a 20-second improvement to 30 seconds per task, thanks in part to the new network design.

Summary

Microsoft.com team members are hard-pressed to define a formula for how much server power, software, and network bandwidth is needed to build a high-traffic Web site. Instead, the team relies on constant monitoring, worst-case planning, and powerful servers. Their experience and recommendations can be summarized in four rules of thumb:

- Learn as much as you can about your audience, potential content, marketing plans, and future initiatives, and then take your best guess.
- When in doubt, choose bigger and faster. You may overbuild today, but you'll likely need the extra capacity sooner than you expect.
- Once your site is in operation, watch it closely. Find out which of the three components—hardware, software (including Web applications), and network bandwidth—is lagging behind, and bring it into balance with the others. Watch for changes in content as well.
- Know that nothing will stay static for long. Keep an eye on growth and usage trends, and be prepared to react (that is, to add or shift hardware, software, or network capacity) quickly.

General Guidelines

To conclude, here are some useful things to keep in mind:

- Inform yourself about the way IIS 5.0 works, so that you can track down trouble before it starts.
- Remember the old rule of thumb: Computing needs rise to fill available resources in about six months, on average. You should be planning for at least six months out.
- Whatever you get, even if it seems sufficient now, is going to require upgrading sooner or later; in today's climate, it's more likely to be sooner than later.
- Don't skimp on RAM. A server computer running IIS 5.0 should have a minimum of 128 MB of RAM, though you can get by with less if you can tolerate some impact on performance. It's probably impossible to set up a server with "too much" RAM.
- If you are concerned with the speed and reliability of disk access, consider caching controllers and "RAID 01" arrays. However, a caching controller is pointless unless lots of your clients tend to request the same information.
- Be extremely aware of security issues. For more information, see "Security" in this book.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server.

Web Links

http://www.microsoft.com/ntserver/ntserverenterprise/exec/feature/wlbs/faq.asp
This contains Network Load Balancing Frequently Asked Questions.

http://www.microsoft.com/backstage/
Microsoft® Backstage has various articles and information about the microsoft.com Web site.

http://home.microsoft.com/reading/archives/sitebldr-4-13-98.asp
and
http://home.microsoft.com/reading/archives/sitebldr-2-16-98.asp
These two sites contain articles by Richard Maring on performance and expansion.

http://www.capacityplanning.com/whitepapers.html
This is a commercial capacity planning site.

http://www.realtime-info.be/encyc/techno/terms/7/66.htm
This site describes the Open Systems Interconnect seven-layer model.

http://www.softwareqatest.com/qatweb1.html
This page provides a wide variety of load and performance tools for download.

http://msdn.microsoft.com/workshop/server/feature/server042798.asp
This site describes InetMon, a capacity planning tool that you can use when testing your site. It supports a wide variety of protocols, and can monitor both services and hardware.

Books

Capacity Planning for Web Performance: Metrics, Models, and Methods by Daniel A. Menasce and Virgilio A. F. Almeida, 1998, Upper Saddle River: Prentice Hall.

Web Performance Tuning by Patrick Killelea; Linda Mui, Editor,1998, Sebastopol: O'Reilly & Associates.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 5

Monitoring and Tuning Your Server

Monitoring is an essential part of server administration. By monitoring the server's performance, you can spot problematic trends as they develop, and take steps to prevent them from turning into emergencies. Monitoring also helps you decide when to upgrade your hardware, and whether upgrades are actually improving your server's performance appropriately. Also, while some server problems are obvious, some develop over time; and unless you have a baseline against which to judge and compare, you can be blind-sided by a problem that you might have avoided. This chapter is intended to help you use the System Monitor and other tools to develop both a baseline and a monitoring strategy to help you track the performance of your server.

Tuning your server improves the client experience, helps you avoid some bottlenecks, and sometimes extends the interval between hardware upgrades. You can use the information from your monitoring program to decide how you should best tune your server.

This chapter does not discuss client connections or client-side performance, except to the extent that the client's performance is controlled by the server.

In This Chapter

Using This Chapter

Memory

Preventing Processor Bottlenecks

Network I/O

Web Applications

Monitoring Security Overhead

Tools

Additional Resources

Using This Chapter

This chapter suggests several approaches to monitoring and improving the performance of a site running Internet Information Services (IIS) 5.0 with Microsoft® Windows® 2000 Server or Microsoft® Windows® 2000 Advanced Server. It is intended as a technical guide for administrators, but should be useful for anyone with one or more servers running IIS 5.0. The sections in this chapter discuss the following topics:

- **Memory** Memory is discussed first because memory shortages often manifest themselves as poor performance in other components, especially processors and disks. It is important to rule out a memory shortage before investigating other components.

  This section presents an overview of some techniques for determining whether a computer has adequate physical memory to be an efficient Internet server. It discusses how IIS 5.0 uses physical memory, including the IIS Object Cache, IIS Template Cache, and IIS Script Engine Cache, as well as the operating system's File System Cache, In addition, it includes suggestions for tuning some server parameters to improve memory use. It also covers Transmission Control Protocol (TCP) sockets.

- **Preventing Processor Bottlenecks** This section presents a brief discussion of tools and strategies for monitoring processor use in single-processor and multiple-processor computers running IIS 5.0, and for preventing the processor from becoming a system bottleneck. It provides information about the role of the processor in servicing client connections and requests, and it includes tips on tuning in order to balance processor load.

- **Network I/O** This section offers some techniques for measuring how much data a network configuration is able to transmit during periods of average and peak use. It includes some suggestions for capacity planning based on the number of files and connections the server is expected to handle.

- **Web Applications** Applications can be responsible for poor performance and inefficient processor and memory use. This section builds on the monitoring techniques discussed in the memory and processor sections. It suggests some ways to monitor the basic aspects of application performance, and explains why some applications perform better than others on IIS 5.0. It also includes some suggestions for optimizing your Web applications, and for mitigating the effects of poor application performance.

- **Monitoring Security Overhead** This section presents a brief discussion of some techniques for monitoring the performance overhead associated with some common security strategies, such as integrated Windows authentication and the Secure Sockets Layer (SSL) protocol.

- **Tools** This section is a short list of some monitoring and performance-testing tools that you can use with Windows 2000 Server and IIS 5.0.

Memory

With memory, more is almost always better. But just pouring more RAM into your server won't necessarily solve your problems, and certainly won't optimize your server. It is important to determine whether you are making good use of the memory you have.

Memory Allocation

The largest "chunk" of activity in Windows 2000 Server is a *process.* The physical RAM available to a process is called its *working set.* If the needs of a process exceed the amount of available RAM, that is, if the process is not able to store all of its code and frequently-used data in physical memory, some of the information must be stored elsewhere, usually on disk (as virtual memory). This causes an increase in the amount of disk activity, which slows down the server. Some memory contains information (typically code) that is used often, and it should not be sent off to disk because this will result in serious performance degradation. A memory region of this type is designated as "nonpageable."

Within each process, Windows 2000 uses *threads* to accomplish particular tasks. The memory occupied by threads is part of the working set of the process, except for some special threads that run in nonpageable memory. Threads that service connections in IIS 5.0, for example, are not pageable. Sockets also use nonpageable memory. If too many sockets and nonpaged threads are created, the system runs out of nonpaged pool memory and a "bottleneck" occurs. (This can happen, for example, during some forms of denial-of-service attack, in which the server is barraged with requests. For more information, see "Security" in this book.

Memory Management

The Windows 2000 Server memory system is largely self-tuning. The Virtual Memory Manager and the Cache Manager adjust cache sizes, the working sets of processes, the paged and nonpaged memory pools, and paging files on disk to produce the most efficient use of memory. Similarly, IIS 5.0 regulates the size of the IIS Object Cache. Therefore, the primary purpose of monitoring memory in a Windows 2000 operating system–based server running IIS 5.0 is not to adjust the size of each memory component, but rather to make sure that the system as a whole has enough memory and is using it efficiently.

There are several major areas to look at when monitoring memory usage on a Web server. The next section discusses them.

Memory Requirements of a Server Running IIS 5.0

Web servers use memory in several ways. A server running IIS 5.0 needs to have enough physical memory to accommodate the following:

- **The Program Code for IIS 5.0** This code occupies about 2.5 megabytes (MB) of memory with all services running simultaneously. It is part of the working set of the IIS 5.0 process, Inetinfo.exe, and can—but for performance reasons should not be—paged to disk. (By default, Internet Server Application Programming Interface (ISAPI) dynamic-linked libraries (DLLs) run in a separate memory space in IIS 5.0.)

  Note In addition, each connection adds approximately 10 kilobytes (KB) to the size of the working set. Thus, when there are 1,000 simultaneous connections, the working set expands by about 10 MB.

- **The IIS Object Cache** Objects that are costly to retrieve and frequently reused by the service, such as file handles, file directory listings, and so on, are stored in the IIS Object Cache, a cache maintained by the IIS 5.0 service. The IIS Object Cache is also part of the working set of the IIS 5.0 process, Inetinfo.exe, and can be paged to disk.

- **The IIS Template Cache and the IIS Script Engine Cache** The IIS Template Cache is a relatively small area of memory that holds, for example, pointers to Script Engines. The IIS Script Engine Cache holds precompiled scripts in Active Server Pages (ASP) ("Script Engines") that are ready to run. The IIS Template and Script Engine caches are part of the dllhost space by default in IIS 5.0, because scripting is run in the out-of-process pool. (Instances of dllhost in IIS 5.0 are equivalent to instances of Mtx.exe in IIS 4.0.) If this setting is changed, however, and scripting is brought in process, they become part of Inetinfo.exe. For more information about running applications in the out-of-process pool, see "Administering an ISP Installation" in this book.

- **The File System Cache** Web page files that are frequently accessed are stored in the operating system's File System Cache, an area of physical memory reserved for items that are used frequently and repeatedly. The remaining files are stored on disk until needed.

- **Overall Memory Usage by the Server.** (This is particularly important on multipurpose servers.) This category includes:
  - **IIS 5.0 Log Files** IIS 5.0 maintains one memory-mapped file for each site that has logging enabled. These files are mapped in 64-KB chunks. The mapped segments of the log files are part of the working set of the IIS 5.0 process. Logged data also appears in the File System Cache because data is cached when the file-mapping objects are written to disk.
  - **TCB Table** TCP maintains a hash table, a scheme for providing rapid access to data items, of Transmission Control Blocks (TCBs) that store data for each TCP connection. A control block is attached to the table for each active connection and is deleted shortly after the connection is closed. The TCB table is part of the operating system's *nonpaged memory pool*. As such, the TCB table must remain in physical memory; it cannot be paged to disk. TCP *sockets* are also maintained in nonpaged memory.
  - **HTTP Connection Data Structures** HTTP maintains pageable data structures to track its active connections. When these data structures are in physical memory, they are counted as part of the working set of the IIS 5.0 process.
  - **Pool Threads** The threads that execute the code for IIS 5.0 services are stored in the *nonpaged pool* in physical memory, along with other objects used by the operating system to support IIS 5.0. Threads in the nonpaged pool must remain in physical memory; they cannot be paged to disk.

Cache sizes, of course, vary depending on the files used by your server and its clients, and on the amount of physical memory present in the server.

Memory consumption is greater for dynamic pages and depends heavily on the type of application involved. For example, an ASP page that formats database tables containing thousands of rows consumes far more memory than even a fairly large static page.

In general, it's good to have at least 256 MB of RAM; you'll want more in a server that is performing memory-intensive tasks.

Monitoring Overall Server Memory

IIS 5.0 is well integrated with the Windows 2000 operating system. Because of this, IIS 5.0 services derive many benefits from the system architecture, including the Windows 2000 Server security model, remote procedure call (RPC) communication, messaging, the file systems, and other operating system services. Thus, monitoring memory for IIS 5.0 begins with monitoring overall server memory, particularly on a multipurpose server.

Monitoring the physical memory of a server running IIS 5.0 involves measuring the size of the areas in physical memory used by IIS 5.0 and assuring that enough space is available to contain the elements IIS 5.0 needs to store. The physical memory space should be sufficient for normal operation and for routine peaks in demand; but your site may also encounter occasional spikes. If it does, you must decide how much degradation in performance (if any) you will allow at those times. Routine peaks on most sites reach about twice the average amount of utilization, whereas spikes can easily be a full order of magnitude beyond the average.

The Windows 2000 virtual memory system is designed to be self-tuning. The Virtual Memory Manager and Cache Manager within Windows 2000 adjust the size of the File System Cache, the working sets of processes, the paged and nonpaged memory pools, and the paging files on disk in order to produce the most efficient use of available physical memory. Similarly, the IIS 5.0 service regulates the size of the IIS Object Cache. Therefore, the primary purpose of monitoring memory in a Windows 2000–based server running IIS 5.0 is to make sure that the server has enough physical memory, not to adjust the size of each memory component, as might be the case with other operating systems.

Memory shortages frequently show up as or appear to be problems in other components. Thus, when your server has a problem (its response time increases, for example, or it cannot handle all of the requests it is receiving), it's a good idea to check memory first.

Memory Monitoring Specifics

This section is a brief review of the most important memory monitoring techniques. Many performance monitoring tools measure memory use system-wide and according to each process. These tools include Task Manager as well as the System Monitor, which is usually referred to as PerfMon (*Perfmon.msc,* supplanting the older *Perfmon.exe*). These are Windows 2000 Server administrative tools built into Windows 2000 Server.
In addition, there are various tools included on the *Microsoft® Windows® 2000 Server Resource Kit* companion CD or in Windows 2000 itself, the latter of which include Process Monitor (Pmon.exe), Process Viewer (Pviewer.exe), Process Explode (Pview.exe), and Performance Data Log Service, which is usually referred to as PerfLog, (Pdlcnfig.exe and Pdlsvc.exe). PerfLog and PerfMon can measure memory use over time. PerfMon, in particular, can read and log data from software performance counters that are built into IIS 5.0 and the Windows 2000 operating system. For more information about PerfMon, see "The System Monitor" in this chapter.

- **Monitor Available Memory** Compare the total physical memory that is available to Windows 2000 with the available memory remaining when you are running all server services. To gather more reliable data, log this value over time, making certain to include periods of peak activity. The system attempts to keep the amount of available bytes at 4 MB or more, but it is prudent to keep at least 5 percent of memory (rather than a specific number of MB) available for peak use.

  To track available memory, log the Computername\ Memory\ Available Bytes counter. ("Computername" indicates the name of the computer you are monitoring, if it isn't the same one on which PerfMon is running. This indicator is not explicitly shown in the names of most of the counters listed in this chapter, but is generally applicable.)

- **Monitor Paging** Continuous high rates of disk paging indicate a memory shortage.

  - **Number of Hard Page Faults** If a process requests a page in memory and the system cannot find it at the requested location, this constitutes a page fault. (If the page is elsewhere in memory the fault is called a soft page fault. If the page must be retrieved from disk, the fault is called a hard page fault.) The system also counts a page fault on a file access if the requested page is not found in the File System Cache and must be retrieved from storage. The page fault counters do not distinguish between hard and soft faults, so you must combine counter information to deduce the number of hard faults.

    To track paging, log the following counters: Memory\ Page Faults/sec, Memory\ Cache Faults/sec, and Memory\ Page Reads/sec. The first two of these track working sets and the File System Cache. The Page Reads counter helps you track hard page faults: a high rate of page faults coupled with a high rate of page reads (these also show up in the Disk counters) indicates a high rate of hard faults.

- **Monitor the File System Cache** The File System Cache is the working set of the file system. This cache is a reserved area in physical memory where the file system stores its recently used and frequently used data. By default, the system reserves about 50 percent of physical memory for the File System Cache, but the system trims the cache if it is running out of memory. A server running IIS 5.0 functions like a specialized file server, and a large and effective File System Cache is vital to its efficient operation. For more information, see "Monitoring the File System Cache" in this chapter.

- **Monitor the Size of the Paging Files** The paging files on disk back up committed physical memory. The larger the paging file, the more memory the system can commit. Windows 2000 itself creates a paging file on the system drive; you can create a paging file on each logical disk, and you can change the sizes of the existing files. In fact, striping a paging file across separate physical drives (use drives that do not contain your site's content) improves paging file performance. Remember, in any case, that the paging file on the system drive should be at least twice the size of physical memory, so that if a crash occurs the system can write the entire contents of RAM to disk. This so-called "core dump" (a term that originated when memory was built of discrete magnetic cores, long before chips were invented) can help you recover data and determine the cause of the crash. To monitor the paging files, log the Computername\ Process\ Page File Bytes: Total counter.

- **Monitor the Size of the Paged and Nonpaged Memory Pools** The system's memory pools hold objects created and used by applications and the operating system. The contents of the memory pools are accessible only in privileged mode. That is, only the kernel of the operating system can directly use the memory pools; user processes cannot. On servers running IIS 5.0, threads that service connections are stored in the nonpaged pool, along with other objects used by the service, such as file handles.

  To monitor the pool space for all processes on the server, log the Computername\ Memory\ Pool Paged Bytes and Computername\ Memory\ Pool Nonpaged Bytes counters. To monitor the pool space used directly by IIS 5.0, log the Computername\ Process\ Pool Paged Bytes: Inetinfo and Computername\ Process\ Pool Nonpaged Bytes: Inetinfo counters.

Tuning TCP Sockets

In IIS 4.0, TCP sockets were allocated on a per-Web-site basis. This meant that if you created thousands of sites on one server, performance suffered. By contrast, in IIS 5.0 sockets are allocated on a per-port basis. Regardless of how many Web sites you place on any particular port, they share the sockets allocated to that port. Of course, if you put a huge number of sites on one port, performance can still suffer; if you have a mission-critical site, you can disable socket pooling for that site to avoid the risk. The relevant property is a metabase entry (**MD_DISABLE_SOCKET_POOLING**), which you can set at /LM/W3SVC/*X* (where *X* is the number of the site) to revert back to the IIS 4.0 default behavior.

If you choose to disable socket pooling, you should do so only at the site level, so that other (noncritical) sites can continue to take advantage of this feature. This property is not available in the IIS snap-in; you can only set it by scripting. For information about setting metabase properties, see the "Active Server Pages Guide" topic of the IIS 5.0 online product documentation.

If you have a real need to disable socket pooling globally, type the following at the command prompt:

```
c:\inetpub\adminscripts\cscript adsutil.vbs set w3svc\disablesocketpooling true
```

The command prompt will reply:

```
disablesocketpooling : (BOOLEAN) True
```

> **Note** If pooling is enabled and you set bandwidth throttling for a site in the pool, the setting affects all the other sites as well. For this reason, it's a good idea to disable pooling for any throttled site(s).

As an alternative to using the command line, you can change metabase settings with Metabase Editor, a tool that is distributed on the Resource Kit companion CD.

Monitoring the Working Set of the IIS 5.0 Process

A server running IIS 5.0 must have enough physical memory to support the IIS 5.0 working set.

The system continually adjusts the size of the working set of a process as the process runs. The amount of space the system provides to the working set depends on the amount of memory available to the system and the needs of the process.

This next section suggests methods for monitoring the size of the IIS 5.0 working set over time and explains how to determine whether the working set of the IIS 5.0 process is large enough.

About the Inetinfo Working Set

The Inetinfo working set does not have a fixed size. The requirements of the working set vary depending upon the number of connections maintained, the number and size of files, and the use of other supporting services, such as security features and logging.

IIS 5.0 runs in a pageable user-mode process called Inetinfo.exe. When a process is pageable, the system can remove part or all of it from RAM and write it to disk if there isn't enough free memory. If part of the Inetinfo process is paged to disk, the performance of IIS 5.0 suffers. It's extremely important to be sure that your server or servers have enough RAM to keep the entire Inetinfo process in memory at all times. The Web, File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP) services run in the Inetinfo process. Each of the current connections is also given about 10 KB of memory in the Inetinfo working set.

ISAPI DLLs, including ASP, can run in or out of process; in IIS 5.0, ASP processing is done in pooled out-of-process mode by default, and should certainly be run out of process during development and testing. The out-of-process pool is part of the dllhost space, the size of which you can also monitor.

The working set of the Inetinfo process should be large enough to contain the IIS Object Cache, data buffers for IIS 5.0 logging, and the data structures that the Web service uses to track its active connections.

Using the System Monitor to Monitor the IIS 5.0 Working Set

You can use PerfMon to monitor the working set of Inetinfo.exe. Because ISAPI DLLs run in the out-of-process pool by default, you'll need to monitor them separately from Inetinfo unless you've changed that setting; and you need to be aware that out-of-process counter information is summed. This makes it difficult to single out any one process or application. (If your site uses custom ISAPI DLLs, those DLLs should incorporate their own counters so that you can monitor them individually.) To track ISAPI DLLs, you should monitor dllhost counters in the Process counter set. You can also monitor the IIS Object Cache.

For more information about running applications in the out-of-process pool, or separately out of process, see "Administering an ISP Installation" in this book.

Table 5.1 lists relevant counters for monitoring the Inetinfo process.

Table 5.1 Counters for Monitoring the IIS 5.0 Working Set

Counter	Indicates
Computername\ Memory\ Available Bytes	The amount of physical memory remaining and available for use, in bytes. This counter displays the amount of memory not currently used by the system or by running processes. It displays the last observed value, not an average. The operating system attempts to prevent this value from falling below 4 MB. It often trims the working sets of processes to maintain the 4 MB minimum available memory.
Computername\ Process\ Working Set: Inetinfo	Size of the working set of the process, in bytes. This counter displays the last observed value, not an average over time.
Computername\ Process\ Page Faults/sec: Inetinfo	Hard and soft faults in the working set of the process.
Computername\ Memory\ Page Faults/sec	Hard and soft faults for all working sets running on the system.
Computername\ Memory\ Page Reads/sec	Hard page faults. This counter displays the number of times the disk is read to satisfy page faults. It displays the number of read operations, regardless of the number of pages read in each operation. A sustained rate of 5 reads/sec or more can indicate a memory shortage.
Computername\ Memory\ Pages Input/sec	One measure of the cost of page faults. This counter displays the number of pages read to satisfy page faults. One page is faulted at a time, but the system can read multiple pages ahead to prevent further hard faults.

You should log this data for several days. You can use Performance Logs and Alerts in PerfMon to identify times of unusually high and low server activity.

Analyzing the Working Set Data

The following sections describe how to use data about the Inetinfo working set to determine if the server has enough memory to support IIS 5.0 efficiently.

Available Bytes and the Inetinfo Working Set

When the amount of available memory falls below about 4 MB, the system attempts to provide more available bytes by taking memory away from the working sets of processes. This strategy increases the rate of page faults, because each process must now retrieve data that was once in its working set either from disk or from elsewhere in memory. When the rate of page faults for a particular process rises, however, the system attempts to expand the working set of that process to lower the page fault rate. As the system tries to maintain this balance, the size of the working set of each process fluctuates accordingly.

If the system has sufficient memory, it can maintain enough space in the Inetinfo working set so that IIS 5.0 rarely needs to perform disk operations. One indicator of memory sufficiency is how much the size of the Inetinfo process working set varies in response to general memory availability on the server.

As Table 5.1 indicated, you can use the Computername\ Memory\ Available Bytes counter as an indicator of memory availability and the Computername\ Process\ Working Set: Inetinfo counter as an indicator of the size of the IIS 5.0 working set. Be sure to examine data collected over time, because these counters display the last value observed, rather than an average.

> **Tip** You may want to have PerfMon alert you if the number of available bytes dips below about 5 percent of the amount of physical RAM on the server, or below a minimum of about 10 MB on small servers.

Page Faults and the Inetinfo Working Set

When you look at page faults, compare your data on the size of the Inetinfo working set to the rate of page faults attributed to the working set. As Table 5.1 showed, you can use the Computername\ Process\ Working Set: Inetinfo counter as an indicator of the size of the working set, and the Computername\ Process\ Page Faults/sec: Inetinfo counter to indicate the rate of page faults for the IIS 5.0 process. When you have reviewed data on the varying size of the Inetinfo working set, you can use its page fault rate to determine whether the system has enough memory to operate efficiently. If the system is not able to lower the page fault rate to an acceptable level, you should add memory to improve performance.

ASP Caching

In addition to various static files, there are two special ASP caches, the IIS Template Cache and the IIS Script Engine Cache. By default, scripts run in the out-of-process pool, so the IIS Script Engine Cache is part of dllhosts rather than Inetinfo. This is also true if you run ASP separately out of process. If you move ASP processing in process, however, these two caches become part of the Inetinfo process. The final HTML that results from executing Script Engines is not cached, because in many cases that output can vary each time the script is executed. For more information, see "Tuning the ASP Queue and Thread Pool" in this chapter.

The IIS Object Cache

The IIS Object Cache is part of the working set of the IIS 5.0 process in physical memory. Because of this, the IIS Object Cache can be paged to disk if memory is not sufficient to support a large enough working set for the IIS 5.0 process. It is important either to provide enough physical memory to maintain the IIS Object Cache in the working set, to reduce **ObjectCacheTTL** so that the cache is flushed more often, or both.

Using PerfMon to Monitor the IIS Object Cache

There are five performance objects for measuring IIS 5.0 performance:

- Internet Information Services Global
- Web Service
- ASP (though, by default, ASP processing is external to Inetinfo)
- FTP Service
- SMTP Service

Table 5.2 lists the counters recommended for monitoring the IIS Object Cache. You should log the counters over time to record trends in the size, content, and effectiveness of the IIS Object Cache.

Table 5.2 Counters for Monitoring the IIS Object Cache

Counter	Indicates
Internet Information Services Global\ Cache Hits	A measure of the efficiency of the IIS Object Cache. These counters demonstrate how often data sought in the IIS Object Cache is found.
Internet Information Services Global\ Cache Misses	Internet Information Services Global\ Cache Misses indicates how often the system must search elsewhere in memory or on disk to satisfy a request.
Internet Information Services Global\ Cache Hits %	The first two of these counters (Cache Hits and Cache Misses) display totals since the service was started. Internet Information Services Global\ Cache Hits % displays an instantaneous value, not an average over time.
Internet Information Services Global\ Cache Flushes	How many times an object was deleted from the IIS Object Cache, either because it timed out, or because the object changed.
Internet Information Services Global\ Objects	The total number of objects currently stored in the IIS Object Cache.
Internet Information Services Global\ Directory Listings	Directory Listings is a subset of the Objects counter. At any given time, the difference between the total number of objects and the number of Directory Listings is equal to the number of other objects stored in the cache. The Directory Listings counter is most important to servers running the FTP service.

Analyzing the Performance of the IIS Object Cache

As Table 5.2 shows, cache performance is judged by how often objects sought in the cache are found. Frequent cache misses result in disk I/O and decreased performance. There are no fixed standards for cache performance, although a value of 80 to 90 percent for Cache Hits % is considered to be excellent for sites with many static files. If Cache Hits and Cache Hits % are very low, or if Cache Misses is quite high, the cache could be too small to function effectively. Adding memory increases the size of the cache and should improve its performance.

Cache flushes can also affect the performance of the IIS Object Cache. Cache flushes are regulated, in part, by an internal timer. The timer activates the *object-cache scavenger,* which deletes expired objects. Objects are flushed from the cache if they change, or if they time out before they are reused.

To measure cache flushes, compare the number of cache flushes over time to the number of cache misses and to the rate of page faults in the Inetinfo process (as indicated by the Process\ Page Faults/sec: Inetinfo counter). It is important to observe these values over time. Like the other global IIS 5.0 counters, Cache Flushes displays an instantaneous value, not an average. If a high rate of cache flushes is associated with elevated cache misses and page faults, it is possible that the cache is being flushed too frequently.

If you suspect that cache flushes are occurring too often or not often enough, you can change the rate at which unreferenced objects are flushed from the IIS Object Cache. Make sure your server has ample memory before you increase the time between flushes. The cache flush time defaults to 30 seconds; to change it, add the **ObjectCacheTTL** key to the registry if it is not already present. Put this key in the following registry location:

```
HKEY_LOCAL_MACHINE
    \System
        \CurrentControlSet
            \Services
                \Inetinfo
                    \Parameters
```

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft® Management Console (MMC) whenever possible.

If you increase **ObjectCacheTTL**, the cache size increases because files stay in the cache longer. Be sure your server has enough physical memory to support the increased load.

Monitoring the File System Cache

The File System Cache is maintained by the Cache Manager in Windows 2000 for use by all processes. This section explains how IIS 5.0 uses the File System Cache, how to monitor the size and efficiency of the cache, and how to interpret the data you collect.

How IIS 5.0 Uses the File System Cache

IIS 5.0 relies on the operating system to store and retrieve frequently-used Web pages and other files from the File System Cache. The File System Cache is particularly useful for servers of static Web pages, because Web pages tend to be used in repeated, predictable patterns.

Also, IIS 5.0 always reads sequentially. Sequential reading takes advantage of a Cache Manager feature called a *read ahead*. This occurs when the Cache Manager's predictive algorithms detect sequential reading and begin to read larger blocks of data in each read operation. Read aheads can provide a significant performance boost to a process.

IIS 5.0 uses both the File System Cache and the IIS Object Cache, sometimes together. When a thread belonging to an IIS 5.0 service needs to open a file, the thread requests a file handle from the operating system. When it receives the handle, the thread uses the handle to open the file. Then, if space permits, the thread stores the handle in the IIS Object Cache and the system stores the file data in the File System Cache. Later, if that thread (or any other thread) needs the file, the file handle can be retrieved from the IIS Object Cache and the file contents can be retrieved from the File System Cache.

Using PerfMon to Monitor the File System Cache

There are several counters in the Memory and Cache performance objects that you can use to monitor the size and effectiveness of the File System Cache. Table 5.3 lists these counters.

Table 5.3 Counters for Monitoring the File System Cache

Counter	Indicates
Memory\ Cache bytes	The size of the cache, in bytes. This counter displays the last observed value; it is not an average.
Memory\ Cache faults/sec	How often data sought in the File System Cache is not found there. The count includes faults for data found elsewhere in memory, as well as faults that require disk operations to retrieve the requested data. This counter displays the number of faults, regardless of the number of pages retrieved in response to the fault.
Cache\ Copy Reads/sec	The frequency of reads from pages of the File System Cache that involve a memory copy of the data from the cache to the application's buffer. This is a method used by the LAN Redirector, the LAN Server (for small items), and the disk file systems.

Continued

Table 5.3 Counters for Monitoring the File System Cache *(continued)*

Counter	Indicates
Cache\ Fast Reads/sec	The frequency of reads from the File System Cache that bypass the installed file system and retrieve the data directly from the cache. Normally, file I/O requests invoke the appropriate file system to retrieve data from a file. However, this path permits direct retrieval of data from the cache without file system involvement, if the data is in the cache. Even if the data is not in the cache, one invocation of the file system is avoided.
Cache\ MDL Reads/sec	How often the system attempts to read large blocks of data from the cache. Memory Descriptor List (MDL) Reads are read operations in which the system uses a list of the physical address of each page to help it find the page. MDL Reads are often used to retrieve cached Web pages and FTP files.
Cache\ Pin Reads/sec	How often the system attempts to read recently accessed blocks of data from the cache. This counter is more accurate for ASP content than the MDL Reads/sec counter is. Pin counters display reads of cache data that is held because it has just been read or written. They reflect cache data that is used repeatedly.
Cache\ MDL Read Hits %	How often attempts to find large sections of data in the cache are successful. You can use the Cache\ MDL Read Hits % counter to calculate the percentage of MDL misses. Misses are likely to result in disk I/O.
Cache\ Pin Read Hits %	How often attempts to find recently accessed sections of data in the cache are successful. This counter is more accurate for ASP content than the MDL Read Hits % counter is. You can use the Cache\ Pin Read Hits % counter to calculate the percentage of misses. Misses are likely to result in disk I/O. Pin counters display reads of cache data that is held because it has just been read or written. They reflect cache data that is used repeatedly.
Cache\ Data Maps/sec	How often pages are mapped into the cache from elsewhere in physical memory or from disk. To measure the percentage of data maps from elsewhere in physical memory, use Cache\ Data Map Hits %. 100 minus the value of Cache\ Data Map Hits % is the percentage of data maps retrieved from disk.
Cache\ Read Aheads/sec	A measure of sequential reading from the cache. When the system detects sequential reading, it anticipates future reads and reads larger blocks of data. The read ahead counters are a useful measure of how effectively an application uses the cache.
Memory\ Page Faults/sec	Hard and soft faults in the working set of the process. This counter displays the number of faults, without regard for the number of pages retrieved in response to the fault.
Memory\ Page Reads/sec	Hard faults in the working sets of processes and in the File System Cache.

Note You should log this data over several days, because cache behavior varies over time.

Analyzing the File System Cache Data

You can use the data you collect with PerfMon to evaluate whether your server has enough memory to support an effective File System Cache.

To evaluate server memory

1. Determine how the size of your server's File System Cache varies over time.
2. Determine the extent to which cache performance varies with cache size.
3. Verify that the server has enough memory to support an effective cache even when the server is most active.

The following sections provide information to help you perform this evaluation.

Analyzing Cache Size Data

The Cache Manager in Windows 2000 adjusts the size of the File System Cache based on whether a computer is a workstation or a server, the amount of physical memory in the computer, and the applications and services the computer is supporting. In general, it is counterproductive to override the Cache Manager and manipulate the cache size directly. If the cache is too small to be effective, it is better to increase the amount of physical memory in the computer, or to redistribute memory-intensive applications to other servers.

Use the Memory\ Cache Bytes counter to monitor the size of the File System Cache. Task Manager also displays the size of the File System Cache in the **File Cache** field under **Physical Memory** on the **Performance** tab. A log of cache size reveals how the size of the File System Cache changes over time. Compare this data to a measure of general memory availability, such as data from the Memory\ Available Bytes counter. In general, when memory is scarce the system trims the cache, and when memory is ample the system enlarges the cache.

Note the points in the log when the cache is smallest. Keep track of how small the cache gets, and how often that happens. Also, note how much system memory is available when the cache size is reduced. This data is useful when associating the size of the cache with its performance. (To put this another way, if performance degrades when the cache is large, you should consider adding RAM. If performance degrades when the cache is small, you should consider leaving objects in the cache longer before you allow them to be flushed.)

Analyzing Cache Performance Data

Use the logged counter data to evaluate the performance of the File System Cache on your server. A *hit* is recorded when requested files are found in the cache. A *miss* or *fault* is recorded when requested files are not found. Misses and faults indicate how often the system needs to do extra work to retrieve files from somewhere other than the cache.

To evaluate the performance of your server's File System Cache, examine a log of Cache\ MDL Read Hits %. The higher the value, the better the File System Cache is performing. Values near 100 percent are not uncommon on IIS 5.0 servers with ample memory. Subtract the percentage of hits from 100 to determine the percentage of misses. Misses on MDL Reads usually require disk operations to find the requested data.

MDL Reads are the most common type of read used for retrieving many contiguous pages. Typically, MDL Reads are also the most common read operation on servers running IIS 5.0. To determine which type of cache read is most common on your server, use PerfMon to report the rates of different types of cache reads. Include Cache\ Copy Reads/sec, Cache\ Data Maps/sec, Cache\ Fast Reads/sec, Cache\ MDL Reads/sec, and Cache\ Pin Reads/sec.

You can also use the Memory\ Cache Faults/sec counter to indicate how often data sought in the File System Cache is not found there. This value should be as small as possible. The Memory\ Page Faults/sec and Memory\ Page Reads/sec counters are included to help you relate the fault rate of the cache to the fault rate in the system as a whole. Memory\ Cache Faults/sec is a component of Memory\ Page Faults/sec. The ratio of Memory\ Cache Faults/sec to Memory\ Page Faults/sec indicates the proportion of faults occurring in the cache, as opposed to the working sets of processes.

A high rate of cache faults can indicate a memory shortage. But it can also indicate that the organization of data on disk is inefficient. If the files used in sequence are stored on the same logical partitions of the same disk, they are more likely to benefit from the optimizing strategies of Cache Manager within Windows 2000, such as read aheads. The Memory\ Read Aheads/sec counter displays the rate of sequential reading from the cache.

Comparing Cache Size and Performance Data

To determine the extent to which the performance of the File System Cache correlates with the size of the cache, compare the size of the cache over time to the rate at which data sought in the cache is found there. Use the Memory\ Cache Bytes counter as an indicator of the size of the cache. Use Memory\ Cache Faults/sec as an indicator of the rate of cache misses, and Cache\ MDL Read Hits % as an indicator of the rate of cache hits.

If Memory\ Cache Faults/sec increases and Cache\ MDL Read Hits % decreases when the system has reduced the size of the File System Cache, the cache might be too small to really benefit the server. A less effective File System Cache is likely to degrade the performance of an IIS 5.0 server significantly, especially if the size of the cache is often reduced because of a general memory shortage.

If cache performance is poor when the cache is small, use the data you have collected to infer why the system reduced the cache size. Note the available memory on the server, and the processes and services running on the server, including the number of simultaneous connections being supported.

When you add physical memory to your server, the system allocates more space to the File System Cache. A larger cache is almost always more efficient, but typically it's a case of diminishing returns—each additional megabyte of memory adds less efficiency than the previous one. You must decide where the trade-off point is: the point at which adding more memory gets you so little improvement in performance that it ceases to be worthwhile.

In addition, defragmenting your disks makes it more likely for related pages to be copied into the cache together. This, in turn, improves the hit rate of the cache. Finally, consider reducing the workload on the server by moving some of the load to another server. For more information, see "Administering an ISP Installation" in this book.

Suggestions for Optimizing Memory Usage

Servers running IIS 5.0, like other high-performance file servers, benefit from ample physical memory. Generally, the more memory you add, the more the servers use and the better they perform. IIS 5.0 requires a minimum of 64 MB of memory; at least 128 MB is recommended. If you are running memory-intensive applications, your server could require a much larger amount of memory to run optimally (for example, most of the servers that service the microsoft.com Web site have at least 512 MB of memory).

Adding RAM to your system is not the only option, however. Here are a few suggestions for optimizing memory performance without adding memory:

- **Improve Data Organization** Keep related Web files on the same logical partitions of a disk. Keeping files together improves the performance of the File System Cache. Also, defragment your disks. Even well-organized files take more time to retrieve if they are fragmented.
- **Try Disk Mirroring or Striping** The optimum configuration is to have enough physical memory to hold all static Web pages. However, if pages must be retrieved from disk, use mirroring or striping to make reading from disk sets faster. In some cases, a caching disk controller may help.
- **Replace or Convert CGI Applications** CGI applications use much more processor time and memory space than equivalent ASP or ISAPI applications. For more information about ASP, ISAPI, and CGI applications, see "Web Applications" in this chapter.
- **Enlarge Paging Files** Add paging files and increase the size of the ones you have. The Windows 2000 operating system creates one paging file on the system disk, but you can also create a new paging file on each logical partition of each disk.

- **Retime the IIS Object Cache** Consider lengthening the period that an unused object can remain in the cache (use the **ObjectCacheTTL** setting in the registry, as mentioned earlier in this chapter, to accomplish this).

  Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

- **Change the Balance of the File System Cache to the IIS 5.0 Working Set** By default, servers running the Windows 2000 operating system are configured to give preference to the File System Cache over the working sets of processes when allocating memory space. Although IIS 5.0–based servers benefit from a large File System Cache, the setting **Maximize Throughput for File Sharing** often causes the IIS 5.0 pageable code to be written to disk, which results in lengthy processing delays. To avoid these processing delays, set Server properties to the **Maximize data throughput for network applications** option.

  To change Server properties

  1. On the desktop, open **My Computer** and select **Network and Dial-up Connections**.
  2. Right-click **Local Area Connection** and open its property sheet.
  3. Select **File and Printer Sharing for Microsoft Networks** and select **Properties**.
  4. On the **Server Optimization** property sheet, select **Maximize data throughput for network applications**.

- **Limit Connections** If your server doesn't have enough memory, limiting the number of connections on the server might help alleviate the shortage because some physical memory (about 10 KB per connection) is consumed by the data structures the system uses to keep track of connections.

  To control the number of current connections

  1. In the IIS snap-in, right-click a site, then choose **Properties** and select the **Web Site** tab.
  2. Select the **Limited To** check box in the **Connections** panel. Type into the field the maximum number of connections you want to allow.

- **Eliminate Unnecessary Features** You can also disable the performance boost for applications in the foreground. In addition, at times when you are not actively checking performance, you can disable performance-related logging in order to squeeze a bit more performance from your server.

Preventing Processor Bottlenecks

Servers running IIS 5.0 rely on the speed and efficiency of their processors. The IIS 5.0 code is multithreaded for efficient scaling on single-processor and multiprocessor computers, and is largely self-tuning. Nonetheless, *processor bottlenecks* are a potential problem on very active servers.

A processor bottleneck occurs when one or more processes take up nearly all the time of all processors on the computer. In a bottleneck, process threads ready to be executed must wait in a queue for processor time. All other activity comes to a halt until the queue is cleared. Processor bottlenecks can occur on multiprocessor computers even when only a single processor is fully loaded, if the work in the queue cannot be or is not distributed to the other processors. Because the problem is not centered in the other components (such as memory, disks, or network connections), upgrading or adding to those components does not overcome this performance problem, and can make it even worse.

Monitoring Server Processors

The processors in a server running IIS 5.0 must support the operating system and processes unrelated to Internet services, as well as IIS 5.0 processes. The processors must also support applications related to Internet services, such as those that assemble data from Microsoft® SQL Server™ databases or that generate dynamic Web pages.

There are several tools you can use to monitor processor performance. Web Capacity Analysis Tool (WCAT) and the Web Application Stress Tool are stress-testing tools that are both available on the Resource Kit companion CD. In addition, Task Manager, PerfLog, and PerfMon are commonly used to monitor Windows 2000–based servers. Remember that all monitoring tools use system resources. Monitor the processor use of the process in which the tool runs. Then, before you analyze your data, subtract the processor time of the tool process from the data.

Using PerfMon to Monitor Processor Activity

To monitor your server's processors, use PerfMon to log data from the counters listed in Table 5.4:

Table 5.4 Counters for Processor Activity Monitoring

Counter	Indicates
System\ Processor Queue Length	Threads waiting for processor time. If this value exceeds 2 for a sustained period of time, the processor may be bottlenecked.
Processor\ % Processor Time (Total instance)	The sum of processor use on each processor.
Processor\ % Processor Time	Processor use on each processor (#0, #1, and so on). In a multiprocessor server, this counter reveals unequal distribution of processor load.
Processor\ % Privileged Time	Proportion of the processor's time spent in privileged mode. In the Windows 2000 operating system, only privileged mode code has direct access to hardware and to all memory in the system. The Windows 2000 Executive runs in privileged mode. Application threads can be switched to privileged mode to run operating system services.
Processor\ % User Time	Proportion of the processor's time spent in user mode. User mode is the processor mode in which applications like IIS 5.0 services run.
Process\ % Processor Time	The processor use attributable to each processor, either for a particular process or for the total for all processes. (These are shown in the list of instances.)

False Indications

A memory bottleneck can sometimes look like a processor or disk bottleneck. If the system does not have enough physical memory to store the code and data that are needed, the processor spends substantial time paging. Before adding or upgrading processors or disks, you should monitor the memory in your server. For more information about monitoring memory, see "Memory" in this chapter.

Analyzing Processor Activity Data

Of the counters listed in Table 5.4, the System\ Processor Queue Length counter is probably the most important for analyzing processor activity data. This counter displays the number of threads waiting to be processed in the single queue shared by all processors. Sustained high rates of processor activity, which leave little excess capacity to handle peak loads, are often associated with processor bottlenecks. Processor activity itself only indicates that the resource is being used, not that maximum use of the resource is a problem. However, a long, sustained queue indicates that threads are being kept waiting because a processor cannot handle the load assigned to it.

A sustained processor queue length of two or more threads (as indicated by the System\ Processor Queue Length Counter) typically indicates a processor bottleneck. You can set a PerfMon alert to notify administrators when the processor queue length reaches an unacceptable value.

The Processor\ % Processor Time counter is most often used as a general measure of processor activity on both single-processor and multiprocessor computers. System\ % Total Processor Time is included for monitoring system-wide processor use on multiprocessor computers. On single-processor computers, System\ % Total Processor Time always equals Processor\ % Processor Time. On multiprocessor computers, System\ % Total Processor Time represents the active time of all processors divided by the number of processors.

If the server workload is shared equally among all processors, System\ % Total Processor time is an excellent measure of processor activity. However, this counter hides bottlenecks resulting from unequal processor loads. (If one processor is 100 percent busy and three other processors are idle, the % Total Processor Time is 25 percent.)

Windows 2000 Server is designed for efficient scaling and includes several strategies for balancing processor load. An application, however, can create an imbalance by setting a processor affinity, which binds a process to a single processor. For detailed processor monitoring, you need to chart Processor\ % Processor Time for each processor on the computer.

It's not unusual to encounter the following challenges in analyzing processor data:

- A large processor queue when all processors are busy. Create a histogram of Process\ % Processor Time for each process. To do this, click the **View Histogram** button on the toolbar in PerfMon. The histogram shows the processor time consumed by each process.
- A single bar in the histogram rises above all of the others. The process represented by the bar may be consuming a disproportionate share of processor time and causing a bottleneck. Consider replacing the application running in that process, or moving the process to another server.
- The processors are being shared equally by several processes. Consider upgrading or adding processors. (Multithreaded processes benefit most from additional processors.)

For more information about processor use by applications related to IIS, see "Web Applications" in this chapter.

Process Throttling

Process throttling limits the CPU usage of out-of-process applications and CGI applications on a per-site basis. The allowed amount is a percentage of the total CPU usage during a given interval, which defaults to one day.

There are four actions that can be taken when a limit is exceeded:

- The lowest level action, engaged whenever a limited site exceeds its allocation at all, is that IIS 5.0 logs the error.
- The second level action, engaged when the site exceeds 150 percent of its allocation, is that all out-of-process applications on the site have their CPU priority set to idle.
- The next-to-highest level action, engaged when the site exceeds 200 percent of its allocation, is that IIS 5.0 halts all processes on the site that are affected by throttling, such as all CGI applications and out-of-process applications. These must then be restarted.
- The highest level action is accessible only through the **CPULimitPause** property of the metabase. When this level is triggered, IIS 5.0 pauses the entire site and causes the server to issue a custom error message when any client attempts to connect to that site. (See the IIS 5.0 online product documentation for more information about this and other metabase settings.)

Process throttling also limits CGI processes to a certain amount of CPU usage. If your server performs CGI processing, you should probably lower the CGI timeout interval. For more information, see the IIS 5.0 online product documentation for more information. If you support a large number of long-running CGI applications, see "A Thread-Related CGI Program Issue" in this chapter.

Monitoring Connections

It is important to determine how your server responds when it is managing different numbers of connections. When you have collected data on connection trends, you can associate data about general server performance with the number of connections being served.

The next few sections discuss why connections have performance overhead, how you can use PerfMon and other tools to measure the overhead, and how to interpret the data you gather.

Performance Overhead of Connections

Each connection that an IIS 5.0 service establishes consumes some processor time. The network adapter card interrupts the processor to signal that a client has requested a connection. Further processing is required to establish and maintain the connection, to fulfill client requests sent on the connection and, when the connection is closed, processing is required to delete the structures that serviced the connection. Each time a connection is established, the load on the server increases.

One aspect of connection overhead is the time it takes to search the TCB table. TCP creates and maintains TCBs to store information about connections. This information can include, for example, data about the precedence of the connection, and its local and remote socket numbers. For efficient control, the TCBs are kept in a hash table, which is stored in the operating system's nonpaged memory pool.

IIS 5.0 includes several features to optimize its handling of connections. Among these features are *HTTP Keep-Alives*, which are different from and independent of Transmission Control Protocol/Internet Protocol (TCP/IP) Keep-Alives. While the latter are messages sent to determine whether an idle connection is still active, the former maintain a connection even after the connection's initial request is complete. The HTTP Keep-Alives feature keeps the connection active and available for subsequent requests. HTTP Keep-Alives, which both the client and the server must support, are implemented to avoid the substantial cost of establishing and terminating connections. They are supported by IIS version 1.0 and later, Microsoft® Internet Explorer version 2.0 and later, and Netscape Navigator version 2.0 and later.

HTTP Keep-Alives are enabled in IIS 5.0 by default. Although they significantly improve bandwidth performance on most servers and improve the response times for clients, you can modify or eliminate them if they are not needed. You can also measure their effect on the performance of your system. To test their effect on the server, you can disable them, but it is recommended that you re-enable them when the test is concluded, to maintain the performance of the server.

Use the IIS snap-in to enable or disable HTTP Keep-Alives. You'll find this setting on the **Web Site** tab of the property sheets for the Web site.

Tip When you use Microsoft Management Console (MMC), right-click to get the properties of an item in a listing.

Using the IIS 5.0 Log to Monitor Connections

You can use IIS 5.0 logging to monitor the number of connections your server makes and to track patterns of client demand for your server. For more information about configuring and interpreting IIS 5.0 logs, see the IIS 5.0 online product documentation.

Using PerfMon to Monitor Connections

Table 5.5 lists the counters that you use with PerfMon to monitor connections to IIS 5.0.

Table 5.5 Performance Counters for IIS 5.0 Service Connections

Counter	Indicates
Web Service\ Current Connections FTP Service\ Current Connections	The number of connections maintained by the service during the most recent sample interval
Web Service\ Maximum Connections FTP Service\ Maximum Connections	The largest number of connections maintained simultaneously since the server was started

Because these counters display the last value they observe, and not an average, you must log their values over time to collect reliable data.

Also, these counters can exaggerate the number of simultaneous connections. This is because, at any given moment, some entries may not have been deleted, even though their connections have been closed.

Analyzing Connection Data

By monitoring the number of connections, you can identify patterns of client demand for your server. Divide your PerfMon logs into regular time intervals, and look at the number of connections served during each interval. Observe the length of the processor queue and the processor use on each processor during periods of small, moderate, and large numbers of connections. This data shows how your configuration responds to each load level.

You can identify a processor bottleneck during a given interval by:

- A long, sustained processor queue (more than two threads).
- High use rates on one or more processors.
- A curve in the graph of the Current Connections counter on any IIS 5.0 service performance object that reaches a high value and then forms a plateau. This pattern often indicates that additional connections are being blocked or rejected.

To prevent processor bottlenecks, make certain that a lengthy processor queue isn't forming when you serve large numbers of connections. You can usually avoid a bottleneck during peak time by setting the connection limit to twice the average value of Current Connections. If the processor regularly becomes a bottleneck when servicing large numbers of connections, you might consider upgrading or adding processors, or limiting the maximum number of connections on the server. Limiting connections can cause the server to block or reject connections, but it helps to ensure that accepted connections are processed promptly.

Note The Active Server Pages, Web Service, FTP Service, and SMTP Server counters collect data at the Open Systems Interconnectivity (OSI) Application Layer. If any connections were blocked, rejected, or reset between the Transport and Application layers, counts of TCP/IP connections may not equal the sum of HTTP, FTP, and SMTP connections. For information about monitoring connections at lower layers, see "Network I/O" later in this chapter.

Monitoring Threads

IIS 5.0 runs in a set of multithreaded processes designed for efficient scaling on single-processor and multiprocessor systems. *Threads* are the sequences of execution in each process that run the process code on the processor. In the IIS 5.0 process, there is no simple association between threads and connections or between threads and requests, nor is there an easily quantifiable relationship between the optimum number of threads in the process and the number of files served, the number of requests filled, or the number of connections maintained.

The relationship between threads, connections, and requests is complex because IIS 5.0 uses the *worker thread* model, rather than the simpler, but less efficient, *thread-per-client* model. Instead of dedicating a thread to each connection or request, IIS 5.0 dedicates one pool of threads, the *worker threads*, to the task of accepting and monitoring all connections. This frees other threads to do the remaining work of the application, such as authenticating users; parsing client requests; locating, opening, and transmitting files, and managing internal data structures.

Even though you cannot associate individual threads with connections or requests, you can:

- Count the number of threads in the IIS 5.0 process.
- Measure the amount of processor time each thread gets.
- Associate the number of threads (and processor activity) with the number of current connections, number of files served, and other measures of server activity and performance.

Several tools monitor or enumerate the threads in a process, including Process Monitor, Process Viewer, Process Explode, PerfLog, and PerfMon. Individual threads are difficult to monitor, especially if they frequently start and stop. Threads are also costly to measure. Be sure to monitor the overhead (by using Process: % Processor Time) of the process in which your tool runs, and subtract it from the data you collect.

Table 5.6 lists the counters that monitor threads. You can add to this list any counters you use to associate numbers of threads with performance, such as Web Service\ Current Connections, Web Service\ Bytes/sec, or Server\ Logon/sec.

Table 5.6 Counters for Monitoring IIS 5.0 Threads

Counter	Indicates
Process\ Thread Count: Inetinfo	The number of threads created by the process. This counter does not indicate which threads are busy and which are idle. This counter displays the last observed value, not an average.
Thread\ % Processor Time: Inetinfo => *Thread #*	How much processor time each thread of the Inetinfo process is using.
Thread\ Context Switches/sec: Inetinfo => *Thread #*	How many times the threads of the IIS 5.0 service are switched onto and off processor. This counter indicates the activity of IIS 5.0 service process threads.

Analyzing the IIS 5.0 Thread Data

You can chart the Process\ Thread Count: Inetinfo value over time to see how many threads the Inetinfo process creates and how the numbers of threads vary. Then, observe the processor time for each thread in the process (Thread\ % Processor Time: Inetinfo => *Thread #*) during periods of high, medium, and low server activity (as indicated by the other performance measures).

You should also observe the patterns of *context switches* over time, which indicate that the kernel has switched the processor from one thread to another. A context switch occurs each time a new thread runs, and each time one thread takes over from another. A large number of threads is likely to increase the number of context switches. While they allow multiple threads to share the processor, content switches also interrupt the processor and might interfere with its performance, especially on multiprocessor computers. As long as processor utilization is under 70 percent, however, this is not an issue.

Optimizing Thread Values

By default, the IIS 5.0 process creates up to four threads per processor. IIS 5.0 continually adjusts the number of threads in its process in response to server activity. For most systems, this tuning is sufficient to maintain the optimum number of threads, but you can change the maximum number of threads per processor, if your system requires it. If the threads in the IIS 5.0 process appear to be overworked or underutilized, consider these tuning strategies:

- If nearly all of the threads of the IIS 5.0 process are busy nearly all of the time, and the processors are at or near their maximum capacity, consider distributing the workload among more servers. You can also add processors, but do so cautiously. Unnecessary or underused processors will degrade performance, not improve it.
- If nearly all threads appear busy, but the processors are not always active, consider increasing the maximum number of threads per processor. Do not increase the maximum number of threads unless you have processors with excess capacity. More threads on the same number of processors cause more interrupts and context switches, and result in less processor time per thread.

To adjust the maximum number of threads in the IIS 5.0 service process, use a registry editor to add the **MaxPoolThreads** entry to the registry. **MaxPoolThreads** does not appear in the registry unless it is added to the following:

```
HKEY_LOCAL_MACHINE
    \System
        \CurrentControlSet
            \Services
                \Inetinfo
                    \Parameters
```

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

MaxPoolThreads is calculated in units of threads-per-processor. Do not set this value below 2 or above 20. Continue monitoring the system carefully to make sure that changing the number of threads achieves the desired effect.

A Thread-Related CGI Program Issue

If your server simultaneously executes a large number of long-running CGI programs, they tie up the entire IIS 5.0 pool of I/O threads and the server hangs until one of the CGI programs finishes running, or crashes. There is a workaround that makes IIS 5.0 use a separate dedicated thread for each CGI application. To enable this workaround, create a DWORD value called **UsePoolThreadForCGI** in the following location, and set it to zero:

HKEY_LOCAL_MACHINE
\System
\CurrentControlSet
\Services
\W3SVC
\Parameters

Suggestions for Improving Processor Usage and Performance

If your data on processor performance indicates that processor queues are developing regularly or while a server is handling large numbers of connections, start by monitoring the server's memory. Rule out a memory bottleneck or add more memory before (or in addition to) adding or upgrading processors.

In addition to the suggestions for optimizing thread values mentioned earlier in the chapter, consider doing the following:

- **Upgrade the L2 Cache** When adding or upgrading processors, choose processors with a large secondary (L2) cache. File server applications, such as IIS, benefit from a large processor cache because their instruction paths involve many different components. A large processor cache (2 MB or more if external, up to the maximum available if on the CPU chip) is recommended to improve performance on active servers running IIS 5.0.
- **Improve DPC Handling** Deferred Procedure Calls (DPCs) are similar to interrupts except that they have a lower Interrupt Request Level (IRQL). Unlike interrupts, DPCs can be delayed, allowing the processor to complete higher priority work. Platforms that distribute interrupts to all processors do not benefit from the Windows 2000 Server default DPC affinity. If you are administering a multiprocessor computer that distributes interrupts symmetrically, such as an Intel Pentium or Pentium Pro (P6) system for Windows 2000 Server, set the value of the **ProcessorAffinityMask** entry in the registry to zero. DPCs will be handled by the same processor that handled the interrupt from which the DPC evolved.

  Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

- **Add Network Adapters** If you are administering a multiprocessor system that does *not* distribute interrupts symmetrically, you can improve the distribution of the processor workload by adding network adapters so that there is one adapter for every processor. Generally, you only add adapters when you need to improve the throughput of your system. Network adapters, like any additional hardware, have some intrinsic overhead. However, if one of the processors is nearly always active (that is, if Processor\ % Processor Time = 100) and more than half of its time is spent servicing DPCs (if Processor\ % DPC Time > 50), then adding an adapter is likely to improve system performance, as long as the available network bandwidth is not already saturated.
- **Upgrade Network Cards** If you are adding or upgrading network adapters, choose those with drivers that support interrupt moderation. Interrupt moderation prevents the processor from being overwhelmed by bursts of interrupts. Consult the driver manufacturer for details. Also, if at all possible, use adapters that support TCP checksum offloading.
- **Limit Connections** If you cannot upgrade or add processors, consider reducing the maximum number of connections that each IIS 5.0 service accepts. Limiting connections can result in connections being blocked or rejected, but it helps ensure that accepted connections are processed promptly.
- **Redesign the Web Site** You can improve performance and reduce the processor workload by optimizing database use, optimizing the design of ASP pages, calling compiled components from scripts in ASP pages, using ISAPI instead of ASP (and ASP or ISAPI instead of CGI applications), substituting static Web pages for dynamic pages, eliminating the use of SSL except where it is necessary, moving trusted out-of-process applications into the Inetinfo process, and eliminating large bitmapped images or optimizing them to reduce their size. (For more information about optimizing scripts in ASP pages, see "ASP Best Practices" in this book.) It is also advisable to set long expirations on infrequently modified static files.
- **Adjust the Maximum Number of Threads** IIS 5.0 tunes the number of threads in its process dynamically. The dynamic values are usually optimal. In extreme cases of very active or underused processors, however, it may help to adjust the maximum number of threads in the Inetinfo process. If you change the maximum number of threads, continue careful testing to make sure that the change has improved performance. The difference is usually quite subtle.

Network I/O

The main purpose of most Web servers, obviously enough, is I/O. Requests come in, and pages go out. This requires a certain amount of bandwidth, and uses other server resources as well. In addition to IIS 5.0, of course, network I/O involves TCP/IP, which is handled by the system.

Network Bandwidth Requirements of a Server Running IIS 5.0

The speed of a connection to the Internet is governed largely by need and budget. Remember that the bandwidth available to clients has a large impact on the bandwidth required by an Internet server.

In intranet-only cases, the available bandwidth is limited by the network. If more bandwidth is needed, the network must be upgraded or, in the case of shared-resource networks (Ethernet, for example), it must be broken into subnets. Some servers have two or more network interfaces. This is the case, for example, if a Web server is connected to a database server via a private network. Different interfaces need not necessarily run at the same speed.

Monitoring the Network Connection

The primary functions of IIS 5.0 are to establish connections for its clients, to receive and interpret requests, and to deliver files—all as quickly as possible. The pace at which these vital functions are performed depends, in large part, on two factors: the effective bandwidth of the link between the server and the network, and the capacity of this link and the server to support network resources.

Bandwidth and Capacity

Bandwidth is measured in several different ways:

- The rate at which bytes are transferred to and from the server.
- The rate at which data packages are sent by the server. Data packages include frames, packets, segments, and datagrams.
- The rate at which files are sent and received by the server.

Effective bandwidth varies widely depending upon the transmission capacity of the link, the server configuration, and the server workload. The values for a single server also change as it operates, in response to demand and to competition for shared network resources.

Network capacity is measured, in part, by the number of connections established and maintained by the server.

Network Specifics (Data Transmission Rates, File Transfers, TCP Connections)

The simplest measure of the effective bandwidth of a server is the rate at which the server sends and receives data. PerfMon displays counts of data transmissions that are collected by many components of the server computer. The components that collect data each reside in different Open Systems Interconnectivity (OSI) layers:

- Counters on the Web, FTP, and SMTP services performance objects measure data transmitted at the OSI Application Layer.
- Counters on the TCP object measure data transmitted at the Transport Layer.
- Counters on the IP object measure data at the Network Layer.
- Counters on the Network Interface object measure data at the Data Link Layer.

As a result of their different positions in the OSI stack, the counters display different data. For example, the counters at the Application Layer count the bytes sent before the data is divided into packets and prefixed with protocol headers and control packets. Counters at the Application Layer measure data in this way because the data is in this form when the application sends it. Counters at the Application Layer do not include retransmitted data.

In addition, the counters display the data in units native to the component measured. For example, the Web Service object displays data in bytes, and the TCP object displays data in segments.

For more information about TCP/IP and about the OSI "seven-layer" model, see "Additional Resources" at the end of this chapter.

Monitor Settings and Relevant Counters for Transmission Rates

The following tables list and describe some of the counters that can be used for measuring transmission rates. The counters in these tables display the transmission rate observed during the last sample interval. They do not display a rolling or cumulative average of the rate. Also, the counters that represent sums of other counters, such as IP\ Datagrams/sec, are simple sums of the other counters' values.

For more information about how counter values are calculated, check the counter type. A counter type determines how PerfMon calculates and displays that particular counter.

Table 5.7 lists and describes counters at the Application Layer.

Table 5.7 Counters for Measuring Transmission Rates at the Application Layer

Counter	Indicates
Web Service\ Bytes Sent/sec	The rate at which the HTTP server application is sending data, in bytes.
Web Service\ Bytes Received/sec	The rate at which the HTTP server application is receiving data, in bytes.
Web Service\ Bytes Total/sec	The rate at which the HTTP server application is sending and receiving data, in bytes; the sum of Web Service\ Bytes Sent/sec and Web Service\ Bytes Received/sec.
FTP Service\ Bytes Sent/sec	The rate at which the FTP server application is sending data, in bytes.
FTP Service\ Bytes Received/sec	The rate at which the FTP server application is receiving data, in bytes.
FTP Service\ Bytes Total/sec	The rate at which the FTP server application is sending and receiving data, in bytes; the sum of FTP Service\ Bytes Sent/sec and FTP Service\ Bytes Received/sec.
SMTP Service\ Bytes Sent/sec	The rate at which the SMTP server application is sending data, in bytes.
SMTP Service\ Bytes Received/sec	The rate at which the SMTP server application is receiving data, in bytes.
SMTP Service\ Bytes Total/sec	The rate at which the SMTP server application is sending and receiving data, in bytes; the sum of SMTP Service\ Bytes Sent/sec and SMTP Service\ Bytes Received/sec.

Table 5.8 lists and describes counters on the TCP object.

Table 5.8 Counters for Measuring Transmission Rates at the Transport Layer

Counter	Indicates
TCP\ Segments Sent/sec	The rate at which TCP segments are sent by using the TCP protocol.
TCP\ Segments Received/sec	The rate at which TCP segments are received by using the TCP protocol.
TCP\ Segments/sec	The sum of Segments Sent/sec and Segments Received/sec.
TCP\ Segments Retransmitted/sec	The rate at which segments are transmitted that contain one or more bytes TCP recognizes as having been transmitted before. Segments Retransmitted/sec is a proper subset of Segments Sent/sec and Segments/sec. To determine the proportion of transmissions caused by failed transmission attempts, divide Segments Retransmitted/sec by Segments Sent/sec.

Table 5.9 lists and describes counters on the IP object.

Table 5.9 Counters for Measuring Transmission Rates at the Network Layer

Counter	Indicates
IP\ Datagrams Sent/sec	The rate at which IP datagrams are sent by using the IP protocol. This counter does not include datagrams forwarded to another server.
IP\ Datagrams Received/sec	The rate at which IP datagrams are received from IP by using IP protocol. This counter does not include datagrams forwarded to another server.
IP\ Datagrams/sec	The sum of IP\ Datagrams Sent/sec and IP\ Datagrams Received/sec.
IP\ Datagrams Forwarded/sec	The rate at which IP datagrams are forwarded to their final destination by the server.

The sum of IP: Datagrams/sec and IP: Datagrams Forwarded/sec represents the rate at which all IP datagrams are handled by the server.

Table 5.10 lists and describes counters on the Network Interface performance object.

Table 5.10 Counters for Measuring Transmission Rates at the Data Link Layer

Counter	Indicates
Network Interface\ Bytes Sent/sec: Adapter#	The rate at which bytes are sent over each network adapter. The counted bytes include framing characters.
Network Interface\ Bytes Received/sec: Adapter#	The rate at which bytes are received over each network adapter. The counted bytes include framing characters.
Network Interface\ Bytes Total/sec: Adapter#	The sum of Network Interface\ Bytes Sent/sec and Network Interface\ Bytes Received/sec.

The Network Interface counters display data about the network adapters on the server computer. The first instance of the Network Interface object that you see in PerfMon represents the loopback. The *loopback* is a local path through the protocol driver and the network adapter. All other instances represent installed network adapters.

Analyzing the Data

The data provided by these counters is collected by different methods, is displayed in different units, and represents the view of different system objects. Some guidelines for interpreting the data follow:

- The IIS 5.0 service counters display the number of bytes transmitted on behalf of each service that server provides. To calculate the total number of bytes sent or received by all IIS 5.0 services, sum the values for each service. You can determine the proportion of bytes transmitted by each service by computing the ratio of bytes for one service to the sum of bytes for all services, or for the network.
- Data collected by the IIS 5.0 service counters underestimates the total number of bytes actually being transmitted to the network by the IIS 5.0 services. These values are collected at the Application Layer, so they measure data only. They do not measure protocol headers, control packets, or retransmitted bytes. In general, the bytes counted by the services represent approximately 60 to 70 percent of the total number of bytes transmitted by the services on the network. If the sum of bytes for all services accounts for two-thirds or more of total network bandwidth, you can assume your network is running at or near the total capacity of its communications link.
- Counters on the TCP and IP performance objects display the rate at which data is sent and received on a TCP/IP connection at the Transport and Network layers, but they do not count in bytes. Counters on the IP performance object display data in datagrams, and counters on the TCP performance object display data in segments. It is difficult to convert segments to bytes because the bytes per segment can vary from 8 KB to 64 KB; the number of bytes per segment depends upon the size of the TCP/IP receive window and the maximum segment size negotiated when each connection is established.
- Counters on the Network Interface performance object display the rate at which bytes are transmitted over a TCP/IP connection, by monitoring the counters on the network adapter at the Data Link Layer. The values of these Network Interface counters include all prepended frame header bytes and bytes that have been retransmitted. These values provide a relatively accurate estimate of the number of bytes transmitted over the network, but they do not measure the bytes transmitted by a specific IIS 5.0 service.

Despite the difficulty of comparing these counters to each other, they can all be related to other performance measures, such as the total number of connections served at a given bandwidth, or processor use at different throughput rates.

Counters and Settings for Monitoring File Transfers

Each successful request to IIS 5.0 results in the transfer of at least one file. Most static Web pages include multiple files, such as a file of text and one or more files of graphics. There are counters for each IIS 5.0 service, which display the number of files sent and received by the Web Service and the FTP service. The SMTP service is slightly more complex; its counters indicate messages sent and messages delivered, as well as messages received.

Table 5.11 lists and describes the file counters.

Table 5.11 Counters for Monitoring IIS 5.0 File Transfers

Counter	Indicates
Web Service\ Files Sent	The number of files or messages sent by the service since the service was started.
FTP Service\ Files Sent	
SMTP Service\ Messages Sent Total	
Web Service\ Files Received	The number of files or messages received by the service since the service was started.
FTP Service\ Files Received	
SMTP Service\ Messages Received Total	
Web Service\ Files Total	The number of files sent and received by the service since the service was started. Files Total is the unweighted sum of Files Sent and Files Received. The SMTP service lacks this counter.
FTP Service\ Files Total	

The file counters for a particular service can be used as indicators of the network activity of that service. They can also be associated with other performance measures to determine the effect of high and low rates of file activity on server components.

Note, however, that the file counters for an IIS 5.0 service display cumulative totals on all traffic since the service was started, regardless of when PerfMon was started. The counters do not display current values or the rate at which files are transmitted.

To calculate file transmission rates, you can use PerfLog to log the file counters. PerfLog automatically logs the time at which measurement is taken. After you have generated a log in PerfLog, you can enter the PerfLog output files into a spreadsheet that associates the time of the measurement and the file count, in order to derive the transmission rates.

Monitoring TCP Connections

If the bandwidth of your server is insufficient to handle its workload, it is likely that clients will be aware of it before the server is. Client requests to the server will be rejected or will time out, or the response will be delayed. On the server side, the indicators are less clear. The server will continue to establish connections, receive requests, and transmit data.

Bandwidth shortages are not uncommon. You can detect one on your server (perhaps even before clients do) by monitoring the success and failure of connections established and rejected by TCP. With ample bandwidth, the server can establish and serve connections before they time out. If it is not sufficient, the connections fail.

The counters on the TCP object are the best indicators of the success of connection requests. The counters on the Web Service and FTP Service performance objects monitor connections maintained by each IIS 5.0 service. The counters on these objects display only successful connection requests. They do not display failed attempts to connect to these IIS 5.0 services. Like all counters at the Application Layer, they do not have information about connections until the connections are established. Performance counters that display the number of simultaneous connections maintained by IIS 5.0 are discussed in the section "Preventing Processor Bottlenecks" earlier in this chapter.

Table 5.12 lists and describes the counters that monitor the success and failure of connections to TCP.

Table 5.12 Counters for Monitoring TCP Connection Successes and Failures

Counter	Indicates
TCP\ Connections Established	The number of simultaneous connections supported by TCP (at last observation). This counter displays the number of connections last observed to be in the ESTABLISHED or CLOSE-WAIT state. It displays the last observed value only; its value is not an average.
TCP\ Connection Failures	The number of connections that have failed since the service was started (regardless of when PerfMon was started). TCP counts a connection as having failed when it goes directly from sending (SYNC-SENT) or receiving (SYNC-RCVD) to CLOSED or from receiving (SYNC-RCVD) to listening (LISTEN).
TCP\ Connections Reset	The number of connections reset since the service was started (regardless of when PerfMon was started). TCP counts a connection as having been reset when it goes directly from ESTABLISHED or CLOSE-WAIT to CLOSED.

You can also use the HTTP Monitoring Tool or a similar tool to monitor the functioning of the server. For more information about the HTTP Monitoring Tool, see "Tools" in this chapter.

Analyzing the Data

At the TCP level, you should monitor the TCP\ Connections Established counter regularly. If you notice a pattern in which the counter value often reaches, but rarely exceeds, a maximum (that is, the graphed line rises and then reaches a plateau), the peak value is likely to indicate the maximum number of connections that can be established with the current bandwidth and application workload. If you observe such a pattern, the server probably cannot support any greater demand.

Failure to support current or increasing demand also might be evident from the number of connection failures and resets. The counters that monitor failures and resets show cumulative values, but you can set PerfMon alerts on the values or use PerfLog to log values over time. You can then use a spreadsheet to calculate the rates at which connections are rejected and reset. An increasing number of failures and resets or a consistently increasing rate of failures and resets can indicate a bandwidth shortage.

Be cautious when interpreting the number of reset connections shown by the TCP\ Connections Reset counter. Resets do not always indicate dropped or failed connections. Many browsers try to minimize connection overhead by routinely closing connections by sending a TCP reset (RST) packet, rather than by closing the connection using a normal close operation. The TCP\ Connections Reset counter does not distinguish between connections that are reset because they are dropped and those that are reset in order to close them abruptly.

Using the Network Monitor Program to Monitor Bandwidth

Network Monitor (NetMon) is a tool you can use to monitor data sent and received by the local computer. NetMon can:

- Capture or trace data, and filter it based on different attributes.
- Monitor throughput based on bytes or frames.
- Monitor bandwidth based on the percentage of the network used.
- Monitor errors, a possible consequence of an overloaded network.

Windows 2000 Server and Advanced Server include NetMon as an optional tool. For more information about NetMon, see the Windows 2000 Server product documentation.

For an overall view of bandwidth, use the **Network Monitor Frames Per Second** and **Bytes Per Second** status bars. Use the **% Network Utilization** status bar to view monitor network capacity used. The **# Frames Dropped** field indicates the number of frames that are not processed because the buffers on the network adapters are full. Frames are dropped when the processor cannot handle the traffic generated by the network.

Limiting Bandwidth for Static Files

If the bandwidth on your server is not sufficient to handle the load imposed by IIS 5.0, you can limit the amount of bandwidth IIS 5.0 uses for static HTML pages (files with .htm or .html extensions). This setting affects all services that route directly through IIS 5.0, but not those that are sent to other engines or applications (ASP, ISAPI, CGI, and SQL Server, for example).

Remember that per-instance bandwidth throttling is not supported for FTP service.

To enable bandwidth throttling

1. In the IIS snap-in, right-click a Web site, then click **Properties** and select the **Performance** tab.
2. Select the **Enable Bandwidth Throttling** check box. In the **Maximum network use** box, type the maximum amount of bandwidth you want IIS 5.0 to use for static HTML pages, in kilobytes per second.

You do not need to restart the server or the service to activate bandwidth throttling; it is enabled dynamically.

Monitoring Bandwidth Throttling

When you enable bandwidth throttling, IIS 5.0 activates a set of counters to monitor it. You can identify these counters by the presence of the phrase "Async I/O" in the counter name. These counters are active only when bandwidth throttling is enabled. (If bandwidth throttling is not enabled, the counters appear in PerfMon, but they always have a value of zero.)

The Async I/O counters are part of the IIS 5.0 Global performance object. They represent totals for all of the IIS 5.0 services. Bandwidth is not measured for each service. Table 5.13 lists and describes the Async I/O counters.

Table 5.13 Counters for Monitoring Bandwidth Throttling

Counter	Indicates
Internet Information Services Global\ Current Blocked Async I/O Requests	The number of requests blocked (that is, held in a buffer until bandwidth is available) by bandwidth throttling as reported during the most recent observation
Internet Information Services Global\ Total Allowed Async I/O Requests	The number of requests allowed by bandwidth throttling since the service was last started
Internet Information Services Global\ Total Blocked Async I/O Requests	The number of requests blocked (that is, held in a buffer until bandwidth is available) by bandwidth throttling since the service was last started
Internet Information Services Global\ Total Rejected Async I/O Requests	The number of requests rejected by bandwidth throttling since the service was last started
Internet Information Services Global\ Measured Async I/O Bandwidth Usage/	The number of bytes sent asynchronously, averaged over a one-minute period, as indicated by a sample taken by bandwidth throttling

Analyzing the Data

The bandwidth setting determines whether IIS 5.0 accepts or rejects a request for a static HTML page, based on periodic samples of the rate at which bytes are sent on the server.

- If the amount of bandwidth used (as indicated by the sample) approaches the maximum set by the user, bandwidth throttling blocks read requests, but allows write requests and transmission requests. Read requests are blocked first because they are likely to result in further requests.
- If the amount of bandwidth used exceeds the maximum set by the user, bandwidth throttling rejects read requests, blocks large write requests and transmission requests, and allows small write requests and transmission requests.

To determine how many requests are being blocked and rejected, monitor the Async I/O counters. These counters display cumulative totals, so it's best to use PerfLog to log the counter values. Alternatively, you can use a spreadsheet to calculate the rate over time. You can also set a PerfMon alert to notify administrators when the number of blocked or rejected requests exceeds a threshold.

No rule exists that sets a threshold or appropriate number of blocked and rejected requests. Tolerance for client delays and rejections is a business rule, not a performance measure. However, you can use the Async I/O counters to enforce your business standards, at least for static HTML pages.

Optimizing the Network Connection

If the bandwidth on your server is not sufficient to support demand, you can solve the problem by increasing overall server bandwidth. You can also increase the effective bandwidth of existing communication links. Some suggestions on how to do so follow; many involve setting parameters that can only be modified by editing the Windows 2000 registry or the IIS 5.0 metabase.

> **Caution** Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

It is frequently possible to reduce your use of bandwidth by optimizing Web application scripts or content. This option is well worth looking into as an interim solution, partly because it can also improve response time and hence the user experience; but if your client base is growing, you will eventually have to increase the bandwidth of your network connection anyway. For more information about scripted Web applications, see "ASP Best Practices" in this book.

Lengthening Connection Queues

You can sometimes effectively increase existing bandwidth by increasing the length of the connection queues. Requests for connections to IIS 5.0 services are held in queues until the service is available to respond to the request. A separate queue exists for each of the IIS 5.0 services, but all queues have the same maximum size. By default, each queue can hold up to 15 connection requests. If the queue to a service is full, any new connection requests are rejected.

The default queue length of 15 connection requests is sufficient for most servers. However, if your server is rejecting many requests when the services are most active, you can increase the maximum number of items in the queue. If you change the queue length, be sure to monitor server processor use, server memory use, and the connection counters to avoid creating a system bottleneck.

To change the maximum number of connection requests in the queue for each IIS 5.0 service, add the **ListenBackLog** key to the registry. Set the value of **ListenBackLog** to the maximum number of connection requests you want the server to maintain. You must place **Listen_Back_Log** in the registry at:

HKEY_LOCAL_MACHINE
\System
\CurrentControlSet
\Services
\Inetinfo
\Parameters

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

Using HTTP Keep-Alives

To ensure optimal bandwidth, you can also verify that HTTP Keep-Alives are enabled. HTTP Keep-Alives maintain a connection even after the initial request is complete. HTTP Keep-Alives are enabled by default, but can be disabled. To disable them, right-click a site in the Internet Services Manager, then click **Properties**; select the **Performance** tab and clear the **HTTP Keep-Alives Enabled** check box.

Disk I/O

IIS 5.0 writes its logs to disk, so there is almost always some activity, even when clients are hitting the cache 100 percent of the time. Under ordinary circumstances, disk activity, other than that generated by logging, serves as an indicator of issues in other areas. For example, if your server needs more RAM, you'll see lots of disk activity because there are lots of hard page faults. But there will also be lots of disk activity if your server houses a database or your users request many different pages.

Monitor Settings and Relevant Counters

Typically you use the Physical Disk counters to watch for spikes in the number of disk reads when the server is busy. If you have enough RAM, most connections will result in cache hits unless, of course, you have a database on the same server and clients are making dissimilar queries, which precludes caching.

If you don't have anything that is obviously disk-intensive on your server, but you see lots of disk activity anyway, you should check RAM use immediately, to make sure you have enough memory.

Monitoring a Remote Computer

When you are monitoring a server remotely, the Internet Information Services Global, Active Server Pages, Web Service, FTP Service, and SMTP Service performance objects and counters appear in PerfMon only when the related IIS 5.0 service is running on the computer being monitored. If you do not see these objects in the **Add to Chart** dialog box, close PerfMon on your computer, start the IIS 5.0 service on the computer being monitored, and then start PerfMon again.

If the objects and counters still do not appear in the **Add to Chart** dialog box, or if they appear intermittently, make certain that no one else is using PerfMon to monitor the remote computer. All remote users must restart PerfMon after the IIS 5.0 service is started, or none of them will see the IIS 5.0 counters. Also, check the Event Viewer application event log for errors. Errors in the service or in loading the counters can also prevent PerfMon from displaying the counters. (Errors are logged on the main machine, not on the remote machine.)

Web Applications

Web applications can be written haphazardly, in which case they may work but will be inefficient (for example, a haphazardly-written script may make many separate references to a database instead of a single comprehensive one). On the other hand, Web applications can be constructed with an eye toward speed and efficiency. In addition, a server can be configured to run Web applications better, or to serve static content better. For most applications that are not well adapted to the use of static pages, using scripts in ASP pages to call server-side components offers performance close to that of ISAPI, with the advantage of more rapid development time than with ISAPI. For more information about Web application development strategy, see "Developing Web Applications" and "ASP Best Practices" in this book.

If Web applications are an important part of your site, the performance of those applications is critical in determining the site's capacity. Testing is the only way to find out the capacity and performance of a Web application. WCAT and the Web Application Stress Tool, which are included on the Resource Kit companion CD, are useful stress-testing tools. ("Additional Resources" at the end of this chapter includes a source for more information about the Web Application Stress Tool.)

Before you write an application, however, it's useful to have a general sense of the performance capabilities of different Web application types. In IIS 5.0, ISAPI applications running as DLLs in the IIS 5.0 process generally offer the best performance. Next come ASP pages, followed by CGI applications.

Tuning the ASP Queue and Thread Pool

There have been significant changes in the handling of ASP queuing and related caching issues in IIS 5.0 relative to IIS 4.0:

- In IIS 4.0, the IIS Template Cache limit was set to -1 (that is, unlimited). In IIS 5.0 it is set to 256 files. You can, and almost certainly should, adjust this to suit your Web site's requirements, but do not set it to zero. For an explanation, see the next section, "How This Affects Server Administration."
- In IIS 4.0, the IIS Script Engine Cache limit was set to 30 files. In IIS 5.0 it is set to 120. As with the IIS Template Cache, you can change this limit.
- In IIS 4.0, **ProcessorThreadMax** was in the registry, and was set to 10. Administrators routinely changed this value themselves as needed. In IIS 5.0, the setting has been moved to the metabase, is set to 25, and is handled automatically.
- In IIS 4.0, the Request Queue length was set to 500 by default. In IIS 5.0, it is set to 3,000. You probably won't need to change this setting, but you can if you need to. In addition, you can use Custom Errors. There is also automatic client-connection testing so that an ASP page is processed and the output sent only if the client is still connected. (Unfortunately, many proxy servers keep the connection alive even if the client cancels it, defeating the purpose of this test.)

How This Affects Server Administration

With the IIS Template Cache limit set to -1, as it was in IIS 4.0, this cache could grow arbitrarily large. On Web sites with lots of ASP content, the IIS Template Cache tended to fill all of the RAM in the server. In contrast, this limit in IIS 5.0 is set by default to 256 files. Because each site has its own requirements, you should reset the limit to meet your site's particular needs.

Perhaps the easiest way to accomplish this is to monitor performance as you increase and decrease the value. Because an entry in this cache can point to one or more entries in the IIS Script Engine Cache, and because best performance occurs if the scripts in ASP pages are found in the IIS Script Engine Cache, you should never set the limit on the IIS Template Cache to zero. (Doing so prevents any hits on the IIS Script Engine Cache, because the IIS Script Engine Cache entry for a particular .asp file can only be referenced through its template. Thus, if the template for it is not cached, by definition, the IIS Script Engine Cache is rendered useless.) IIS Script Engine Cache hits provide better performance than hits on the IIS Template Cache, so if you make IIS Script Engine Cache hits impossible, performance suffers badly unless all your pages are static.

To monitor the IIS Template and Script Engine Caches, use the following counters in PerfMon: Active Server Pages: Templates Cached, Active Server Pages: Template Cache Hit Rate, and Active Server Pages: Script Engines Cached.

IIS 5.0 includes a mechanism that effectively adjusts the value of the **ASPProcessorThreadMax** metabase entry "on the fly" (the actual value is not changed). When processor utilization drops below 50 percent, which indicates that threads are blocked (perhaps while waiting for an external database to return the results of a query), IIS 5.0 increases the number of active threads so that other requests can be serviced in a timely manner. When processor utilization exceeds 80 percent, indicating a relatively unblocked situation, IIS 5.0 deactivates threads to reduce the amount of context switching. Both lower and upper bounds are settable; 50 percent and 80 percent are the default values.

You can still change the length of the ASP Request Queue if you need to, and you can use Custom Errors to return friendly messages if your server becomes too busy to handle all current requests. Remember that a custom "Server Too Busy" error message is returned only when the length of the queue is at its maximum and another request comes in. (For information about creating custom error messages, see the IIS 5.0 online product documentation.)

In sum, a lot of the tweaking that was necessary in IIS 4.0 is performed automatically in IIS 5.0, thus reducing the workload on server administrators. But administrators can still change settings to accommodate specific conditions they encounter on their servers.

Optimizing for Web Applications

Web applications have much higher overhead than static HTML pages. But this should not deter you from using them. By monitoring applications and estimating their overhead during periods of varying activity, you can make sure your server is prepared for the increased workload.

Here are some suggestions for optimizing your configuration for Web applications:

- **Upgrade Processors** Web applications benefit from faster processors.
- **Add Processors** Components called by ASP can (and should) be multithreaded, which means they can run simultaneously on multiple processors; adding a second processor to a single-processor system brings the most benefit. Even if the expense is not prohibitive, you shouldn't use more than four processors per system at this time.
- **Add Memory** Adding memory may help if applications are running within their own processes. (By default, ASP and ISAPI applications run within the IIS 5.0 process, but you can choose to run them within their own memory spaces.)
- **Defragment Your Disks** If your cache performance declines over time, defragment the disk(s). Your files may have become fragmented over time.
- **Redesign Your Static Pages** Running Web applications takes more time than serving static pages. If you are generating pages dynamically to satisfy user preferences, consider substituting as many as 10 or 20 different static variations for a single dynamically-generated page. If you are generating pages dynamically to provide frequently-updated data, consider redesigning your application so that it generates a single dynamic page on a fixed schedule and then stores that page for retrieval until the next update.
- **Convert CGI Scripts to Scripts in ASP Pages or to ISAPI Applications** Scripts in ASP pages and ISAPI applications are optimized to run on Windows 2000. CGI applications are much less efficient, because each invocation spawns an entire process. For more information about converting CGI scripts, see "Migrating a Web Server to IIS 5.0" in this book.
- **Convert Scripts in ASP Pages into COM Objects** Scripts in ASP pages are interpreted; if a script involves one or more loops, converting it to a COM object can improve its performance by a considerable margin.

Web applications are continuing to increase in popularity, constituting an ever-larger proportion of the average Web server file base. The challenge for administrators is to preserve speed and efficiency while using them.

Short-Term Problems and Temporary Fixes

If you expect a large spike in traffic on a particular page or set of pages, you can always change that page or set it to static HTML for a few days. This may involve some redesign, but the effort may prove worthwhile. When many thousands of people traverse three or four scripted pages to get to a particular download page during a short period, for example, your server is using a lot more resources than it really needs to. You could, instead, provide a static front page with simple navigation, on a temporary basis, and save yourself and your users a lot of headaches.

Tuning Tips

Here are some general tips for maintaining or improving performance.

- This may seem obvious, but sometimes people do forget: Make sure debugging for ASP is turned off.
- Set Expires for all images and for HTML wherever possible. Proxy servers and browsers will make fewer calls to the Web server as a result.
- Remove Microsoft® Visual Basic® objects from ASP Session state if they are Apartment threaded (not Java or most C++ objects, though).
- Cache output from ASP if possible; if not, cache inputs to ASP if possible.
- If you have a large installation with many Web sites, each attached to a different IP port, your server may create too many backlog monitor threads (the system creates a thread for every 64 Web sites). The workaround for this is to disable the backlog monitor. To disable the backlog monitor, set the following registry key to 1:

  HKEY_LOCAL_MACHINE
  \System
  \CurrentControlSet
  \Services
  \Inetinfo
  \Parameters
  \DisableBacklogMonitor

  Then restart the FTP and Web services.

  Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

Monitoring Security Overhead

Security is achieved only at some cost in performance. Measuring the performance overhead of a security strategy is not simply a matter of monitoring a separate process or threads. The features of the Windows 2000 security model and other IIS 5.0 security services run in the context of the IIS 5.0 process; they are integrated into several different operating system services. You cannot monitor security features separately from other aspects of the services.

Instead, the most common way to measure security overhead is to run tests comparing server performance with and without a security feature. The tests should be run with fixed workloads and a fixed server configuration, so that the security feature is the only variable. During the tests, you probably want to measure:

- **Processor Activity and the Processor Queue** Authentication, IP address checking, SSL protocol, and encryption schemes are security features that require significant processing. You are likely to see increased processor activity, both in privileged and user mode, and an increase in the rate of context switches and interrupts. If the processors in the server are not sufficient to handle the increased load, queues are likely to develop. Custom hardware may help here. For more information, see "Security" in this book.
- **Physical Memory Used** Security requires that the system store and retrieve more user information. Also, the SSL protocol uses long keys—40 bits to 1,024 bits long—for encrypting and decrypting the messages.
- **Network Traffic** You are also likely to see an increase in traffic between the IIS 5.0–based server and the domain controller used for authenticating logon passwords and verifying IP addresses.
- **Latency and Delays** The most obvious performance degradation resulting from complex security features like SSL is the time and effort involved in encryption and decryption, both of which use lots of processor cycles. Downloading files from servers using the SSL protocol can be 10 to 100 times slower than from servers that are not using SSL.

If a server is used both for running IIS 5.0 and as a domain controller, the proportion of processor use, memory, and network and disk activity consumed by domain services is likely to increase the load on these resources significantly. The increased activity can be enough to prevent IIS 5.0 services from running efficiently. It is a good idea to test such a server thoroughly before deploying it.

Measuring Security Overhead with WCAT

WCAT is a script-driven, command line–based application that tests your server configuration using a variety of predetermined, unvarying workloads. You can use WCAT to test how your server responds to different workloads or test the same workload on varying configurations of the server.

WCAT includes a folder of prepared test workloads; alternatively, you can use it to create your own workloads. WCAT also includes a special option, ssl.*testname*, which adds SSL protocol settings to any workload test.

A WCAT test simulates clients and servers communicating over a network, and ordinarily requires at least three computers for each test:

- At least one computer that simulates a client, which runs one or more *virtual clients*
- One computer that acts as a *server*
- One computer, called a *controller*, which initiates and monitors the test

Both client and controller can run on one computer. (In fact, all three functions can run on a single computer, but this produces skewed results.)

To produce a realistic test, it is best to associate four or more client computers, each running several virtual clients, with each server. The processors in the client computers should be at least as fast as the processors in the server computer. If they are not, more client computers should be associated with each server computer. WCAT works best if the network that connects the computers has little or no traffic that is not related to the test. It is preferable to use a link dedicated only to the test. A 100 Mbps or faster network is recommended.

Testing Security Features

To test a security feature, first run a WCAT test with the feature, then run the same test without it. It is important to run the "with feature" and "without feature" versions of the test on varying workloads. WCAT includes over 200 MB of prepared workloads ranging from 12 files to 1,600 files. You can create additional tests of workloads with 2,000 or more files.

WCAT has several options for collecting data on the tests:

- You can use WCAT's own log of performance data. Open this log, which is a text file, with any word-processing program. The WCAT user guide explains how to interpret a WCAT log.
- You can run IIS 5.0 logging in conjunction with WCAT to count the number of logons and file accesses.
- You can collect performance counter data with WCAT. The WCAT **run** command includes a **-p** switch to do this. You can select counters by entering the names of counters in a script file. WCAT even includes a sample counter file, Server.pfc. (Monitoring performance counters adds only minimally to the load on the system.) When you use this option, WCAT logs the counter data in its usual report, and also outputs a performance log file in tab-separated form, so that you can import the logged counter data into a spreadsheet or data-processing software.

You should repeat each test several times and average the results to eliminate unintended variations of the test conditions. Then, compare the results of the "with feature" and "without feature" tests. Consistent differences in the results of the tests are likely to indicate the overhead associated with the security feature. You can use these results for planning configuration changes, in order to handle the security overhead.

WCAT is the primary tool used for determining security overhead. However, PerfMon also includes a set of counters you can use to monitor one specific aspect of security: authenticating users.

Using PerfMon to Track Anonymous and Nonanonymous Connections, and Not-Found Errors

The Web Service and FTP Service performance objects include counters that display the number of anonymous and nonanonymous connections to each of these IIS 5.0 services. (The term *nonanonymous* is used instead of *authenticated* to account for custom authentication schemes that require data from the client other than, or in addition to, the user name and password, as is the case with authentication.)

By themselves, these counters help you determine the number and proportion of each type of connection. You can also use the counter values to estimate the effect of changing how you handle anonymous and nonanonymous users. For example, if the vast majority of connections are anonymous, prohibiting anonymous connections has a more significant impact than if most connections are nonanonymous.

Combining data from these counters with general measures of server performance, such as data on processor time, the processor queue, memory, disk reads and writes, and throughput, is even more useful. Using the combined data, you can associate varying numbers and proportions of anonymous and nonanonymous users with their effect on the performance of system components.

The counters that display the numbers of anonymous and nonanonymous connections are called Current Anonymous Users and Current NonAnonymous Users. These counters, however, actually display connections, not users. Users who connect more than once are counted each time they connect. The Anonymous and Nonanonymous User counters display the number of anonymous and nonanonymous connections to the IIS 5.0 service when the values were last observed. They do not report averages or rates. These counters can exaggerate the number of connections, because closed connections may not yet have been deleted when the counter is displayed.

Table 5.14 lists the counters for anonymous and nonanonymous connections. These counters are part of the Web Service and FTP Service performance objects.

Table 5.14 Counters for Anonymous and Nonanonymous Connections

Counter	Indicates
Web Service\ Anonymous Users/sec	How many anonymous and nonanonymous users connect to the IIS 5.0 service each second. (These counters provide instantaneous values rather than averages.)
Web Service\ NonAnonymous Users/sec	
Web Service\ Current Anonymous Users	How many anonymous and nonanonymous users are currently connected to the IIS 5.0 service.
FTP Service\ Current Anonymous Users	
Web Service\ Current NonAnonymous Users	
FTP Service\ Current NonAnonymous Users	
Web Service\ Maximum Anonymous Users	The maximum number of anonymous and nonanonymous users that have been connected simultaneously to the IIS 5.0 service since the service was last started.
FTP Service\ Maximum Anonymous Users	
Web Service\ Maximum NonAnonymous Users	
FTP Service\ Maximum NonAnonymous Users	
Web Service\ Total Anonymous Users	A running total of anonymous and nonanonymous connections to the IIS 5.0 service since the service was last started.
FTP Service\ Total Anonymous Users	
Web Service\ Total NonAnonymous Users	
FTP Service\ Total NonAnonymous Users	

The Current Anonymous Users and Current NonAnonymous Users counters operate based on the following definitions:

- The *Anonymous User* counters display the number of connections whose requests either did not contain a user name and password, or whose user name and password were ignored because authentication is not permitted on the server. If anonymous connections are not permitted on the server, the value of all anonymous user counters is always zero.
- The *NonanonymousUser* counters display the number of connections whose requests contained a valid user name and password, or whatever authentication is required by a custom authentication scheme. If authentication is not enabled on the server, and none of the applications that run on the server request or require authentication, then the value of all nonanonymous user counters is always zero.

These counters count successful connections only. If a client request for an anonymous connection is rejected and the client responds with valid authenticating data, the connection is counted as nonanonymous.

Counting Not-Found Errors

The Web Service performance object includes a counter that displays *not-found errors*. Not-found errors are client requests that could not be satisfied because they included a reference to a Web page or a file that did not exist. (These errors are sometimes described by their HTTP status code number, which is 404.)

Many not-found errors occur because Web pages and files are deleted or moved to another location. However, some can result from user attempts to access documents that they are not authorized to have. (The code number of these "Access forbidden" errors is 403, and most browsers report them differently from 404 errors. They do not show up in the Not Found Errors/sec counter results.)

You can use the Web Service\ Not Found Errors/sec counter to track the rate at which not-found errors are occurring on your server. Alternatively, set a PerfMon alert to notify the Administrator when the rate of not-found errors exceeds a threshold.

Table 5.15 is a brief description of the Web Service\ Not Found Errors/sec counter.

Table 5.15 Counter for Not-Found Errors

Counter	Indicates
Web Service\ Not Found Errors/sec	The number of client read requests that could not be satisfied because the URL did not point to a valid file. An increase in not-found errors might indicate that a file has been moved without its link being updated. However, it can also indicate failed attempts to access protected documents, such as user lists and file directories.

Analyzing the Data and Planning Upgrades

After you have collected data on the effect of adding security features to your server configuration, you can use the results to plan configuration changes. This new configuration will help you handle the additional workload required to support security features. The following approaches are recommended:

- **Upgrade or Add Processors** Security features are often very processor-intensive. Once again, it should be noted that the SSL protocol consumes a significant amount of processor time. Because Windows 2000 security features are multithreaded, they can run simultaneously on multiple processors. Thus, adding processors improves performance significantly and prevents the processors from becoming a bottleneck.

  For best results, choose processors with large (up to 2 MB) secondary (L2) cache space. When encrypting and decrypting data, the processor spends much of its time reading and writing small units of data to and from the main memory. If this data can be stored in the processor cache instead, the data can be retrieved much faster.

- **Add Memory** If security features cause increased paging or shortages in virtual memory, adding more memory will help. The physical memory used to support the security service consumes space that could be used to cache files. To accommodate peak use, you should allow for twice as much memory as required during times of average use, while still maintaining 10 MB of available memory.

- **Use Custom Hardware** Custom hardware can make a significant difference in the way your server handles security overhead. (Be sure to test any nonstandard hardware thoroughly, to be sure it is fully compatible with the other hardware and software you use.)

Do not, however, add disk space solely to support security features. Any increased disk activity associated with security features is likely to result from a shortage of physical memory, not from an actual need for more disk space. Security features, such as the SSL protocol, rely primarily on processors and physical memory, as opposed to disk space.

Tools

This section provides a description of the tools mentioned in this chapter. Some tools passively watch either the server or the network; others provide a way to actually probe and stress-test the server and its network connections. The following tools are provided with Windows 2000 Server, IIS 5.0, or the Resource Kit companion CD.

The System Monitor

The System Monitor, commonly referred to as PerfMon (Perfmon.msc), is a snap-in control for MMC. It reads software counters that are built into both IIS 5.0 and Windows 2000 Server. It also lets you log counter activity and set alerts. PerfMon is indispensable for watching your server. The startup PerfMon display is shown in Figure 5.1.

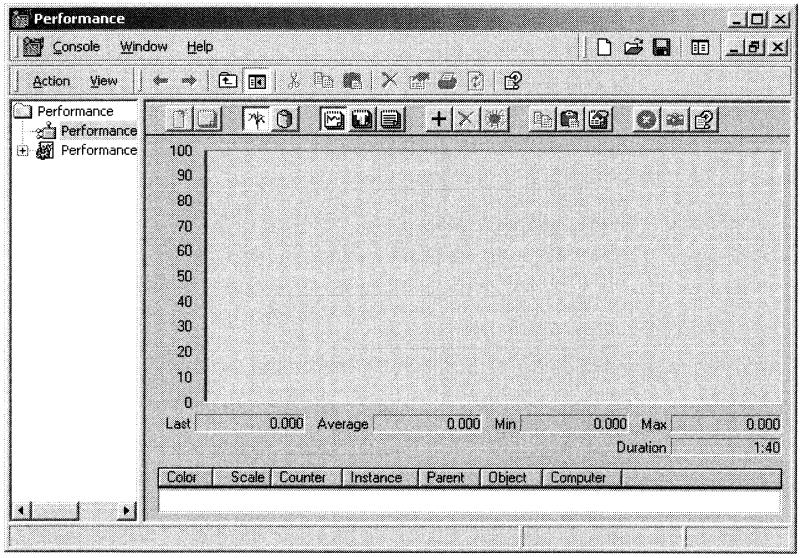


Figure 5.1 The PerfMon Snap-in

Performance Counters

Software performance counters are built into IIS 5.0 and the Windows 2000 operating system. Developers can also build them into custom ISAPI DLLs. These counters are frequently misrepresented as being "PerfMon Counters," but in fact they are built into IIS 5.0 and the operating system, not into PerfMon. These performance counters can be read directly by many applications, including PerfMon, the Web Application Stress Tool, and WCAT. PerfMon provides a handy user interface to performance counters, for the purposes of monitoring and logging.

Each performance counter is named by the object it collects data from (for example, the processor) and by the data it collects (for example, in the PhysicalDisk performance object, Current Disk Queue Length). The object and the specific counter name are separated by a backslash (\). If more than one instance of a particular counter exists, the instance name is separated from the counter name by a colon, and if more than one object or instance of a particular type exists (for example, if your server has more than one processor chip), a hash mark or number sign (#) and a number (the count of the specific object, starting at 0) are presented at the end of the counter or instance name (for example, Process\ % User Time: explorer#0). For some counters, "<No Instances>" is shown in grayed-out text in the list of instances, if there is no current instance.

For a description of a particular counter, click the + button in PerfMon, then select the counter you're interested in, and click the **Explain** button. Note that many of the IIS 5.0 counters display the last observed value or a cumulative count, not an average value or rate.

Different counters provide different kinds of information, at varying levels of granularity, so it's important to know which counters to watch. For example, there are both logical and physical Disk counters; the counters of the Processor object are replicated for each processor in the system, and so on. To choose a specific instance of a counter, use the list of instances on the right-hand side of the **Add Counters** dialog box, as shown in Figure 5.2. To reach this dialog box, click the + button in the toolbar on the right-hand side of the PerfMon console.

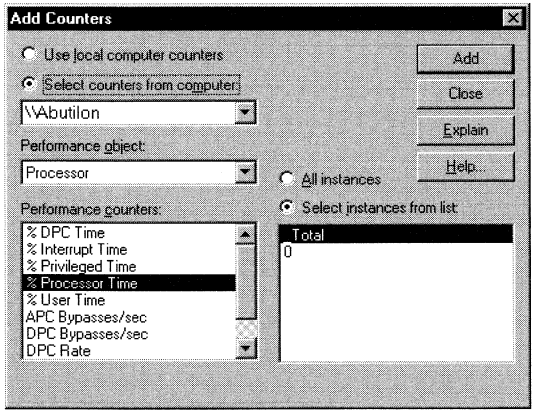


Figure 5.2 PerfMon Add Counters Dialog Box

PerfMon is not limited to the computer on which it is installed. Use the **Select counters from computer** field in the **Add Counters** dialog box to choose a different computer to read from. This is especially helpful when you don't want to add any more overhead to the operation of your server than is absolutely necessary. Remember that the service you want to monitor on the other computer must be running in order for its counters to be exposed. If the service is not running, you need to quit PerfMon, start the service, and run PerfMon again.

Performance Counter Check

Performance Counter Check, included on the Resource Kit companion CD, is a simple tool that allows you to read performance counters from within Microsoft® Windows® Script Host (WSH) or from within an ASP page. It is a scriptable COM object that reads one counter at a time. (To monitor more than one counter, make multiple instances of the object.)

The HTTP Monitoring Tool

The HTTP Monitoring Tool, included on the Resource Kit companion CD, monitors HTTP activity on your server and alerts you to dramatic changes in activity. If a supposedly active server stops serving pages, for example, it might have crashed and probably needs to be checked. The HTTP Monitoring Tool is composed of three main components:

- **Real-Time Sampling Service** is a Windows 2000 Service that makes HTTP requests to ASP pages on live machines.
- **SQL Reporting Server** uses FTP to retrieve sampling results, and loads this data into a SQL Server database that is used for reporting the results.
- **Client Monitor** uses ASP pages displayed in a browser to show status information about site availability, based on the data stored in the SQL Reporting Server.

NetStat and NetMon

These two tools collect information about the network.

NetStat is a command that you issue from the command line. It detects current network connections and lists them with information about protocol, local address, foreign address and state. NetStat goes hand in hand with the Net Use command, which lists network connections to other servers. For a listing of options, type **netstat /?** or **net use /?** at the command-line prompt.

NetMon (the Network Monitor) is one of the Windows 2000 Administrative Tools. Use it to monitor network traffic. NetMon is not installed by default, but you can install it yourself from **Control Panel**.

To install NetMon

1. Open **Add/Remove Programs** and select **Configure Windows**.
2. Click the **Components** button to start the Windows 2000 Components Wizard.
3. Select **Networking Options**, click the **Details** button, and select **Microsoft Network Monitoring Tools**.

Process Viewer, Process Explode, Process Monitor, and Event Viewer

These tools (Pviewer.exe, Pview.exe, Pmon.exe, and Eventvwr.msc respectively) are provided with the Windows 2000 operating system, but only the Event Viewer is available from the Start menu as an Administrative Tool. (Use the Start menu to reach the others.)

Process Viewer lets you examine each process in detail, set its priority and thread priority, and even stop the process if necessary. It can also give you detailed information about memory usage. You can use Process Viewer to connect to other computers and to view their processes.

Process Explode examines processes on the local system only. It lets you view each process in considerable detail, set thread priority, view and change security settings, and even terminate applications if necessary.

Process Monitor, like Process Explode, examines processes on the local system only. It lets you view each process in some detail, but does not permit you to set parameters.

Event Viewer examines the system's event logs. For example, if one or more services fail to start when you bring up a server, you can use Event Viewer to begin your investigation of the problem.

Web Application Stress Tool and WCAT

Included on the Resource Kit companion CD, both of these tools are intended for stress testing, and are superficially equivalent. But you'll find that they differ in some areas, and you are likely to prefer one or the other for a particular testing task. Although WCAT will not be discussed in this section, both tools have documentation on the CD. The next section of this chapter also includes a tutorial on the Web Application Stress Tool.

Getting Started with the Web Application Stress Tool

The Web Application Stress Tool is designed to realistically simulate multiple browsers requesting pages from a Web site. You can use this tool to gather performance and stability information about your Web application. This tool simulates a large number of requests with a relatively small number of client machines. The goal is to create an environment that is as close to production as possible, so that you can find and eliminate problems in the Web application prior to deployment.

Installing the Web Application Stress Tool

For instructions on installing the Web Application Stress Tool, see the Resource Kit companion CD.

Using the Web Application Stress Tool Sample Script

When you run the Web Application Stress Tool for the first time, it creates a script named Sample Script. Use this script to get a feel for the main features in the Web Application Stress Tool. The actual Web pages used in the Sample Script are stored in the folder where you installed the Web Application Stress Tool. Copy the \samples folder to the root directory of your Web site before continuing with this tutorial.

Script View, Script Items, and Script Item Details View

The left-hand window of the Web Application Stress Tool is known as the Script view, which displays all of the scripts stored in the current Web Application Stress Tool database. If this is a new database, the only items you see in the Script view are Defaults and Sample Script.

There are seven script items in the Sample Script, each of which utilizes a special feature of the Web Application Stress Tool. For example, notice that one of the script items is a POST. Double-click the row header to the left of the POST script item and double-click. This opens the Script Item Details view, shown in Figure 5.3. Using this view, you can edit the query string name-value pairs, change POST data, modify the header, enable Remote Data Service (RDS), and enable SSL.

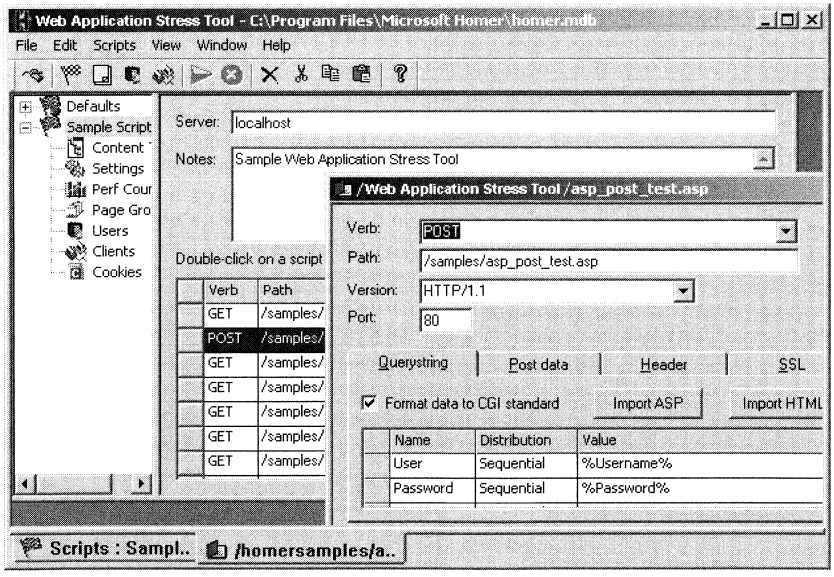


Figure 5.3 The Web Application Stress Tool Script Item Details View

Notice that the names **User** and **Password** on the **Querystring** tab are being passed in the query string. Also notice that **%Username%** and **%Password%** are the values being passed. These are not literal values; they are variables that tell the Web Application Stress Tool to pass the next available user and next available password from the list. The Web Application Stress Tool automatically cycles through the user names and passwords, passing the next set with each POST.

Close the Script Item Details view by clicking the **OK** button.

Page Groups

The last two script items in the Sample Script contain the text "adGrp" under the Group column. These are known as page groups. A page group is shown as "Default" unless you change it. The Web Application Stress Tool uses page groups to change the order in which it invokes the script items, and to change the number of times that it invokes each script item while a script runs.

Select the **Page Groups** node in the script tree view, shown in Figure 5.4, to see a list of all the page groups. You can also change the distribution percentages from this view. Notice that Keep-Alives are enabled for entire page groups at a time. For more information about page groups in the Web Application Stress Tool, see the Knowledge Base article at http://webtool.rte.microsoft.com/kb/hkb12.htm.

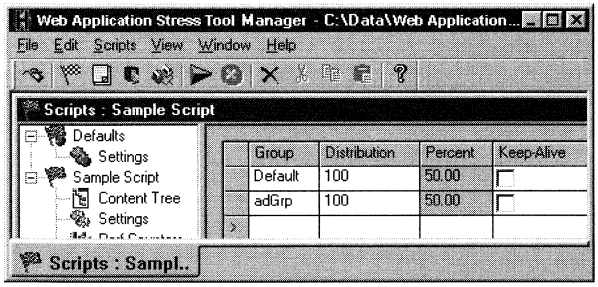


Figure 5.4 The Web Application Stress Tool Page Groups Node

Performance Counters

Select the **Perf Counters** node from the script tree and click the **Add Counter** button. (The first time you click this button, the Web Application Stress Tool may take a few seconds to load the **Add Counters** dialog box.) Add the following counters:

- Computername\ Web Service\ Get Requests/sec
- Computername\ Web Service\ Post Requests/sec
- Computername\ Processor\ % Processor Time_total
- Computername\ Active Server Pages\ Requests/sec

Change the Collection Interval, shown in Figure 5.5, to every 5 seconds. Capturing the correct performance monitor counters is central to obtaining the correct data. There are several important performance counters to choose from, based on the type of application you are testing. In addition to the lists of counters in this chapter, there is a Web Application Stress Tool online documentation topic titled "Common Performance Monitor Counters" that contains a list of the most common Web-related counters and an explanation of each.

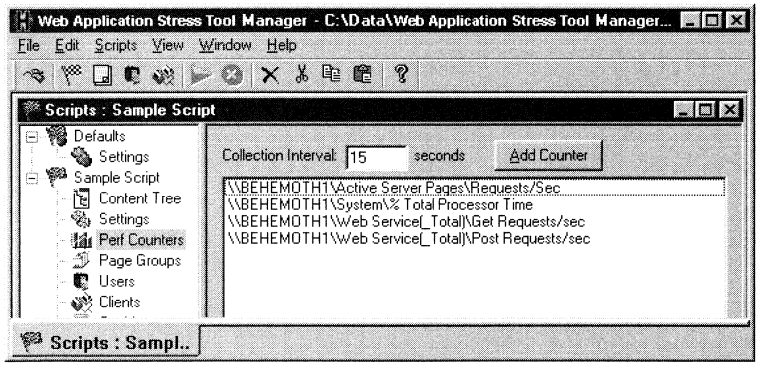


Figure 5.5 View of the Web Application Stress Tool Collection Interval Setting

Settings

Select the **Settings** node, shown in Figure 5.6, and change the Test Run Time to 1 minute (down from the default setting of 15 minutes). Leave the other settings as they are for the time being, but look over the other options in this view, to get an idea of what can be configured for a script. You can also change the default settings for all new scripts by selecting the **Defaults** node and changing those options. Remember, though, that the settings in the Defaults node will not affect existing scripts (for example, the Sample Script).

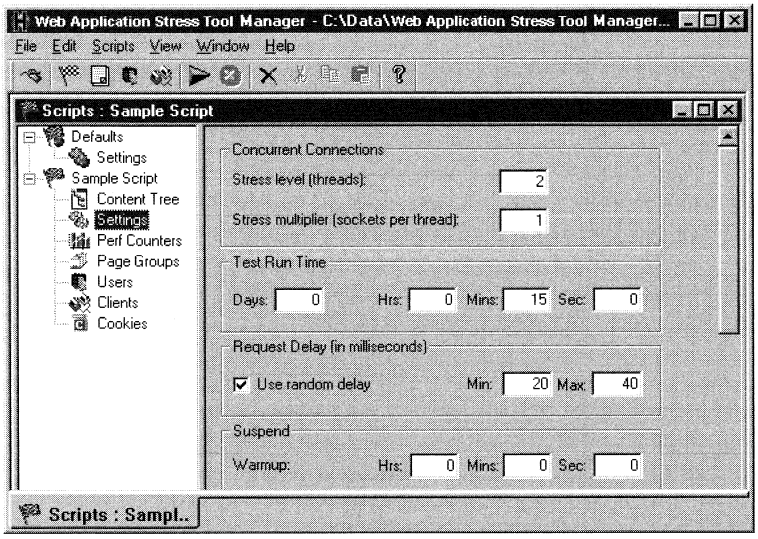


Figure 5.6 The Web Application Stress Tool Settings Node

Users

Select the **Users** node and double-click the **Default** user population, indicated by the **Default** icon. This opens the Users view where you can add and delete users from the default population and create new populations. Each Web Application Stress Tool user stores cookie information and authentication data. Notice that there are 20 users in the default population.

> **Note** The setting for Web Application Stress Tool users is not the same as the setting for Stress level (threads), which is located in the Concurrent Connections section of the Settings node.

Clients

To return to Script view, choose **Scripts** from the **View** menu. Select the **Clients** node under the Sample Script and double-click the **Default** client group, indicated by the **Default** icon. This opens the Clients view, where you can add and delete client machines from the current group or add new groups of client machines. Notice that "localhost" is the only client and that it has a check box next to it. This means that the current machine is acting as a Web Application Stress Tool client. For now, leave the Clients view as is, and select **Scripts** from the **View** menu to return to the Script view.

For more information about users and clients in the Web Application Stress Tool, see the Knowledge Base article called "Understanding Threads, Users, and Clients" at http://webtool.rte.microsoft.com/kb/hkb32.htm.

Running a Test

Once you have created a script and configured all the settings, users, and clients, you can start the test. Select **Sample Script** and choose **Run** from the **Scripts** menu. Allow the test to complete.

Reporting

To open the Reports view, shown in Figure 5.7, choose **Reports** on the **View** menu. Expand the **Sample Script** report to display all of the report nodes. There should be at least one node with a title that marks the date and time when your most recent test began. To view a summary of that test, expand and select the top level of this report node.

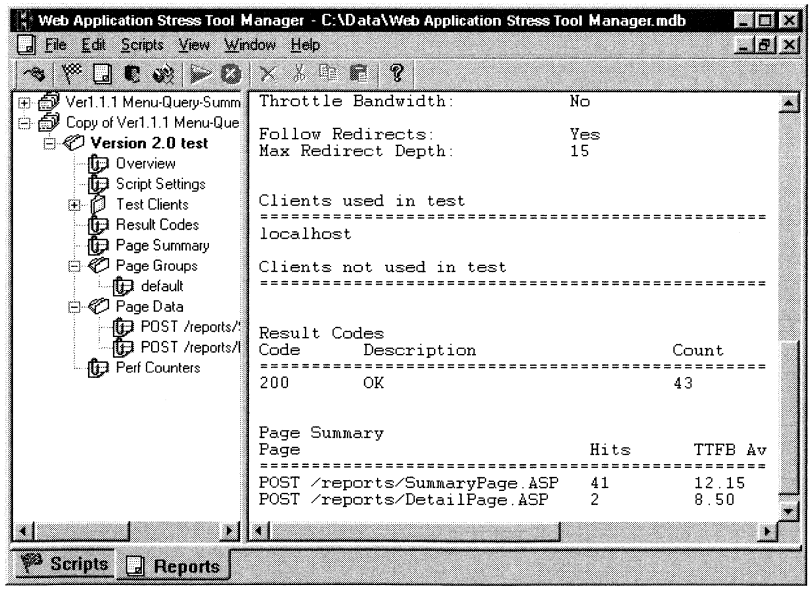


Figure 5.7 View of Web Application Stress Tool Reports

You can select a specific node of the report tree to view more in-depth information. For example, select the **Result Codes** node to display a sum of the HTTP result codes for every request in the test. Expanding the **Perf Counters** node displays the counters that have been collected during the test.

Expand the **Page Data** node and select the first script item. The pane on the right displays detailed information regarding this script item. The page data section of the report provides the response time for each requested page. This period of time, also known as Time To Last Byte (TTLB), is a good source of performance data.

Time To First Byte (TTFB) is the time from the request for the page until the Web Application Stress Tool receives the first byte of data, in milliseconds. TTLB is the total time from the request until the last byte of data has been received on the client, also in milliseconds. This number includes the TTFB time and any additional time needed to receive all of the data for the page. The requests are sorted, and the data is divided into percentiles. For more information about the p-squared algorithm and percentiles, see the Web Application Stress Tool Knowledge Base article at http://webtool.rte.microsoft.com/kb/hkb30.htm.

In the Report view, you can select **Export to CSV** from the **File** menu. Exporting the report values to a format that Microsoft® Excel can read allows you to create charts that show where most of the requests fall. For example, the Excel chart shown in Figure 5.8 illustrates this in a Time To Last Byte report.

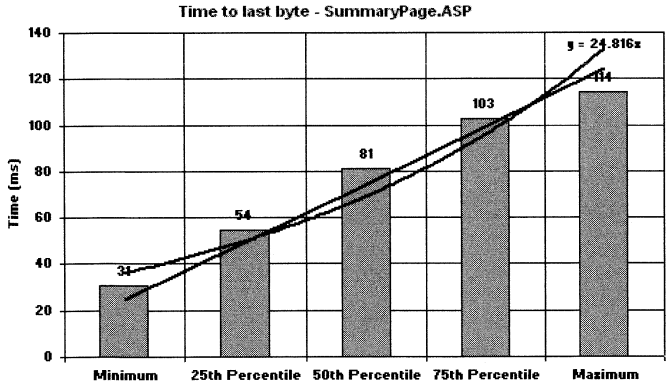


Figure 5.8 An Excel Chart of Sample Web Application Stress Tool Results

General Guidelines for Using the Web Application Stress Tool

- As a rule, use between 10 and 100 threads per client machine. If you find that you need to run a stress test that requires more than 100 threads per client machine, consider adding more client machines.
- Monitor processor utilization on the clients; anything below 80 percent should work well. A client machine may not be capable of sustaining the stress loads when its processor utilization is greater than 80 percent. If this occurs, the test becomes invalid.
- When you are adjusting the threads and sockets for a Web Application Stress Tool test, use just one socket (stress multiplier) unless you are performing a special type of test. For more information about this setting, see the Web Application Stress Tool online documentation help topic "Stress Level vs. Stress Multiplier."
- Limit the number of users to less than 1,000 unless there is a specific reason that you require more unique users. Although the number of users allowable is limited only by the amount of memory on the client machine(s), you may find that it takes too long for a test to initialize when you have a large number of users.
- Avoid creating scripts with more than 1,000, script items. The number of scripts is limited only by the amount of memory on the client machine, but you may find that it takes too long for a test to initialize when your scripts contain large numbers of script items.

Suggestions

In some situations, the Web server processor is the bottleneck. Increase the stress to the point where the number of requests per second starts to decrease, then back the stress off slightly. This is the maximum performance that the Web site can realistically achieve. (You increase the stress by increasing the Stress level setting—the number of threads—or by increasing the number of Web Application Stress Tool client machines.)

On the computer that is monitoring the Web server, use PerfMon to watch the Computername\ System\ % Total Processor Time, Computername\ Web Service\ Connection Attempts/sec, and Computername\ Active Server Pages\ Requests Queued counters of the Web server. If the processors are running at 80 to 85 percent of capacity, then they are most likely to create a bottleneck. On the other hand, if the number of Requests Queued fluctuates considerably during the stress test and the processor utilization remains relatively low, this is an indication that the script is calling a server COM component that is receiving more calls than it can handle. In this case, the server COM component is the source of the bottleneck.

It is a good idea to have an expected peak load that meets your business needs and then create a test that uses enough threads to reach that capacity. This will help you to determine the maximum request rate for the Web application and to confirm that it is in line with the expected peak load.

The amount of personalization in a Web application also plays a significant role in how it performs. The Web Application Stress Tool contains several features that make testing a personalized Web site easy. For example you can create users, and allow the Web Application Stress Tool to save a cookie with each of them. You can also use the Querystring editor to help create and store several sets of name-value pairs that are passed with each request.

Common Web Testing Problems

- Running tests against a platform that doesn't match production Web servers
- Using a script that doesn't accurately simulate production request behavior
- Insufficient processor power or scripts that are too complex for the hardware
- Thread safety issues caused by unstable server COM components
- In ASP pages, script errors and Global.asa issues

Useful Counters for Stress Testing

The following three tables list counters that provide a good indication of the health of a Web application during a stress test. Start with these counters, and add more to drill down to the cause of a bottleneck or resource leak. (The "Computername" parameter is omitted from the counters listed in these tables in order to save space.)

Table 5.16 Useful Counters for Monitoring Web Server PerformanceDuring Stress Testing

Object	Counter	Purpose/Description
Web Service	Maximum Connections	The maximum number of simultaneous connections to the Web service. Use this number to determine the total number of concurrent connections a server experienced during a test run. If this number grows throughout a test, this is a good indication of a blocking backend dependency.
Web Service	Bytes Total/sec	Shows the sum of bytes sent and received by the HTTP service. If this number is low, it means that IIS 5.0 is not transferring data at a reliable rate. Both ASP pages and HTTP content transfer bytes.

Continued

Table 5.16 Useful Counters for Monitoring Web Server PerformanceDuring Stress Testing *(continued)*

Object	Counter	Purpose/Description
Web Service	Current Connections	Shows the current number of connections to the service. This is the sum of both nonanonymous (authenticated) and anonymous (unauthenticated) users. If this number is at or near the maximum connections allowable, the Web service is at full capacity. Check the Web Site property page for this limit.
Web Service	Current NonAnonymous Users	The number of authenticated users currently connected to the HTTP Server. Use this number to determine the number of authenticated connections the Web server is seeing. If this is not greater than 0 during an authenticated stress run, confirm that you are using valid user accounts and that the appropriate permissions are being applied to the Web content.
Web Service	Not Found Errors	Shows the number of pages that are displaying an HTTP 404 result code to client. Anything greater than 0 is an indication of an invalid test run, because the test script is requesting pages that do not exist.
Active Server Pages	Errors/sec	The number of errors per second, including connection errors, compile errors, and run-time errors. If this number is greater than 0, something is wrong with the test scripts, server configuration, or scripts in ASP pages.
Active Server Pages	Requests/sec	The number of ASP page requests executed per second. Use this number to provide an indication of how heavy the stress on the Web server is. This number does not include HTTP requests and will fluctuate considerably based on the complexity of the ASP pages and the capacity of the Web server.
Active Server Pages	Request Not Found	Unlike the Web Service\ Not Found Errors, this shows only the ASP pages that were not found. These pages display as an HTTP 404 result code to the client. Anything greater than 0 is an indication of an invalid test run, because the test script is requesting pages that do not exist.
Active Server Pages	Requests Rejected	The total number of requests not executed because the queue was full or there were insufficient resources to meet the number of hits that the Web server is seeing. If this number is greater than 0, the stress test is too heavy, or the ASP pages are too complex.

Continued

Table 5.16 Useful Counters for Monitoring Web Server PerformanceDuring Stress Testing *(continued)*

Object	Counter	Purpose/Description
Active Server Pages	Memory Allocated	The total amount of memory currently allocated by ASP. Compare this number to Memory\ Available Bytes and Memory\ Committed Bytes to determine what percentage ASP is using. A ratio of greater than 50 percent during the test would indicate a memory leak in a Server-Side Object.
Active Server Pages	Requests Queued	This should remain close to 0 but it will go up and down when testing a heavily scripted ASP page. The maximum number for this counter varies by the number of processors on the machine and the metabase setting for **AspRequestQueueMax**. If the limit is reached, your browser will display "HTTP/1.0 Server Too Busy." If this queue grows rapidly as more stress is applied, this indicates that the ASP pages are too complex for the load.
Active Server Pages	Errors During Script Run Time	This is a count of errors that occurred in the ASP pages you are requesting. This should always be 0. If it sporadically shows a value greater than 0, look at the IIS 5.0 log to see which page is causing the problem and check for the error that occurred. If you have the registry set to capture these, the event log will also display the error.
Memory	Page Faults/sec	Displays the number of times a virtual page was not found in memory. If this number is consistently above 0, it indicates that too much memory has been allocated to an application, and not enough to Microsoft® Windows® 2000 Professional, or to the server you are running.
Server	Total Bytes/sec	Shows network activity. This gives you an idea of how close the Web server's network adapter(s) are to being fully utilized. This is particularly useful on "multihomed" Web servers (which have a network adapter for each LAN they connect to).
Process	Private Bytes\ Inetinfo	The current number of bytes the HTTP/ASP service has allocated that cannot be shared with other processes. If this number is consistently large and growing, there is probably a leak in a Server-Side Object. Compare with Process\ Private Bytes: _Total.

Table 5.17 Useful Counters for Monitoring SQL Server Performance During Stress Testing

Object	Counter	Purpose/Description
SQLServer	Cache Hit Ratio	Shows the hit rate that data is found in the cache. If data is not in the cache, the server will have to be read from the disk. This shows memory allocation and therefore indicates whether the server has sufficient memory for the task. A number consistently less than 85 percent indicates a memory problem.
SQLServer	User Connections	Shows the number of active SQL users. Each connection uses 37 KB of memory. Compare this number to the Active Server Pages:Requests/sec counter to get an idea of how much the scripts are really working the SQL server. A large difference may indicate that the test script is not a valid stress of SQL server.
SQLServer	Net—Network Reads/sec	Shows the number of data packets read from the network. An extremely high value for an extended time indicates that either the network card has a problem, or more likely, the application is not using enough stored procedures and is written improperly.
SQLServer	I/O—Lazy Writes/sec	The number of flushed pages per second by the lazy writer. A number consistently above 0 indicates that the lazy writer is constantly working to flush buffers to disk. This means that the application either has a memory leak or the SQL Server requires more memory for normal operation.
SQLServer - Locks	Total Blocking Locks	Shows a lock that forces another process to wait until the current process is complete. An occasional block is normal. If this number is consistently greater than 0 it indicates transaction problems. Some of the basic causes are inefficient query design, poor table design, or slow throughput due to inadequate hardware.
PhysicalDisk	Disk Queue Length	Shows the number of outstanding requests on the disk. Sustained queue lengths on one disk that are greater than 3 indicate a disk or memory problem. It may also indicate that a SQL Server is not set up correctly.

Table 5.18 Useful Counters for Monitoring Performance of Both SQL and Web Servers During Stress Testing

Object	Counter	Purpose/Description
Processor	% Total Processor Time	Shows the amount of time spent processing threads by all CPUs. A number consistently above 85 percent on one or more processors indicates that the test is too intense for the hardware. Make sure you add the 0 through x instances of this counter for multiprocessor machines.
PhysicalDisk	% Disk Time	**Requires "diskperf -y"** at the command prompt. Shows the percentage of elapsed time that the disk is busy with read/write activity. A number consistently above 80 percent may indicate a memory leak. Make sure you add the 0 through x instances of this counter for multidisk machines.
Memory	Available Bytes	Shows the total bytes of real memory available to the computer, less the number of bytes being used by running applications. Add this number to Memory\ Committed Bytes to get the total amount of memory on the machine.
Memory	Pool Nonpaged Bytes	Shows pages that are used by the operating system that do not leave memory. This will increase with each process, but watch for gradual growth in this counter over a test run. This would indicate an application's repeated inadvertent opening of a file or some other object. Performance will suffer if this approaches within 4 MB of Memory\ Available Bytes.
Memory	Pages/sec	Shows how many pages are being moved to and from the disk to satisfy virtual memory requirements. If the server does not have enough memory to handle its workload, this number will be consistently high.
Memory	Committed Bytes	Shows the size of virtual memory that has been committed to the running applications. Add this number to Memory\ Available Bytes in order to compute the total amount of memory on the machine. Committed bytes should increase as the test is ramping up, but it should remain fairly constant after that.
System	Total Interrupts/sec	Shows the frequency with which this computer is handling hardware interrupts. This gives you a good idea of how busy the whole system is.
Object	Threads	Threads are the basic executable entity that can execute instructions in a processor. If this number continues to rise over time, open the Computername\ Process\ Thread Count counter to discover which instance is creating all of the threads.
Process	Private Bytes: _Total	Shows the current number of bytes that all instances have allocated that cannot be shared with other processes. Make sure you select the _Total instance from the list, in addition to any other instances you suspect may be consuming too much memory.

Suggested Values

Table 5.19 lists some of the counters that are generally useful for monitoring your server, with suggestions about appropriate or dangerous values. Most of these values are given in relative terms, because actual numbers vary from site to site. For example, the Computername\ Active Server Pages\ Transactions/ sec counter is not relevant to a Web site with only static content, and Computername\ Disk\ Avg Disk Bytes/Transfer is different for different kinds of controllers and drives. (Again, "Computername" is omitted, in order to conserve space.)

Table 5.19 Useful Counters for Monitoring Web Server Performance

Object:Counter	Preferred or Ideal Value
Memory\ Pages/sec	0-20 (bad if over 80; probably indicates insufficient RAM).
Memory\ Available Bytes	At least 4 MB.
Memory\ Committed Bytes	Not more than about 75% of physical memory size.
Memory\ Pool Nonpaged Bytes	Steady (slow rise may indicate a memory leak).
System\ Context Switches/sec	As low as possible.
Processor\ % Processor Time	Less than 75%.
Processor\ Interrupts/sec	Depends on processor, and on network hardware and drivers. Up to 3,500 for 90 megahertz (MHz) Pentium; more than 7,000 for 200 MHz Pentium. Lower is better. If the value is too high, try moving some hardware devices to a different server.
Processor\ System Processor Queue Length	Less than 2 (but see the text for exceptions).
Disk (Logical or Physical)\ % Disk Time	As low as possible.
Disk (Logical or Physical)\ Queue Length	Less than 2.
Disk (Logical or Physical)\ Avg. Disk Bytes/Transfer	As high as possible.
Internet Information Service Global\ Cache Hits %	As high as possible.
Web Service\ Bytes Total/sec	As high as possible.
Active Server Pages\ Request Wait Time	As low as possible.
Active Server Pages\ Requests Queued	Zero.
Active Server Pages\ Transactions/sec	As high as possible.

In addition to these, you will find the Computername\ System\ Threads, Computername\ System\ Processes, Computername\ System\ Context Switches/sec and other System Object counters useful for monitoring your server's use of System resources.

Examining the Results

If you find problems, you can use these and other counters to analyze them. For example, if you see evidence of a memory leak (a slow rise in committed bytes or pool nonpaged bytes), you should monitor processes. Similarly, if you see evidence of a disk bottleneck, you can examine reads and writes separately as you begin to track down the problem.

> **Note** When interpreting the counters, remember that they show the most recently observed value, not a long-term average, unless they specify otherwise. You can use the log mode in PerfMon to determine the average values.

If you support a SQL database, be sure that your SQL Server Objects report events to Windows 2000 Server if the objects suffer errors. If, for example, your server runs 10 threads, and all of those threads become involved in SQL Server Objects that hang, ASP processing comes to a complete halt until you stop and restart the server. You can examine the event log and the Connection Attempts/ sec counters to detect this condition.

Internet Information Services Global\ Cache Hits % should be high because otherwise you are thrashing your server's disk. On the other hand, when you first start your Web server this number is going to be low until the cache has actually been populated with files. If you see a low number a few minutes after you bring up the server, you can ignore it. If the number is still low after you've had thousands of hits, something is probably wrong unless (for example) you are maintaining a large database on your server, and you have users performing a wide variety of unrelated queries. If very few pieces of information are requested more than once, the number of cache hits is going to be small; there isn't much you can do about it unless you can afford enough RAM to put the entire database in memory. This, at least, will provide a considerable decrease in access time on the database server.

Baseline Logging

You can use the counters mentioned in Table 5.19 to generate a baseline log that shows the characteristics of your server under ordinary conditions. Such a log should cover at least one week of normal operation, so that it contains enough information to give you a good sense of your server's behavior. Shorter logs that you create from time to time can help you decide when to upgrade.

Whenever you upgrade, you should generate another baseline log. You can compare this to previous logs, in order to discover the actual effects your upgrade is having on performance and capacity.

You can also log the counters of the Job Object to find out how much of your server's capacity is in use for Web service. In PerfMon, these counters provide an overview; if you host multiple Web sites and want a detailed breakdown, the IIS 5.0 log is where you need to look.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server, as well as useful resources for performance tuning.

Web Links

http://webtool.rte.microsoft.com/default.htm
The Web Application Stress Tool is available here.

http://microsoft.com/msj/0398/dcom.htm
This site contains information about the International Standards Organization's (ISO) Open Systems Interconnection (OSI) (the so-called "seven-layer model").

Books

Web Performance Tuning by Patrick Killelea; Linda Mui, Editor, 1998, Cambridge: O'Reilly & Associates.

Tools

Playback
This tool records traffic accessing an IIS 5.0 Web site. It then replays that captured traffic on the same server, or on a different server that mirrors the content of the server on which the recording took place. Playback is included on the Resource Kit companion CD.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 6

Developing Web Applications

The adoption of Internet standards coupled with the immense popularity of the Web have changed the architecture of distributed computing. The multilayered nature of the Web is an ideal environment for the development of component-based applications. They can be developed and customized quickly, with advanced system services such as database access and transaction processing. System resources can be managed and administered remotely. Moreover, new applications are available immediately, without requiring anything more than a browser on the user's system.

This chapter examines the opportunities that the Web provides for distributed application development, and demonstrates how to use Internet Information Services (IIS) 5.0 to develop the *n*-tier (also called multitier) Web applications of the future. In the process, the chapter will introduce client-based and server-based technologies that Microsoft has developed to implement this new breed of Web applications. The chapter assumes the reader is familiar with software development concepts.

In This Chapter

Building on Client/Server

Client-Side Technologies

The Middle Tier

Design Patterns for Web Applications

Debugging Applications and Components

Additional Resources

Building on Client/Server

Market analysts have noticed a trend toward developing multitier applications that are distributed over Internet-standard networks, and predict rapid growth in these distributed systems in the coming years. Some predict that, by 2005, the familiar architecture of client/server applications will be replaced by "super-suites" of interconnected components, operating in frameworks of widely-available distributed systems. In other words, applications will be assembled from reusable building blocks, by using a variety of cooperating subsystems.

Before delving into the implementation details of building Web applications, it might be helpful to take a brief look at the architecture of the Web from a historical perspective, beginning with the traditional client/server architecture.

Client/Server Revisited

Cooperating and communicating applications have typically been categorized as either *client* or *server* applications. While the client application requests services using Microsoft® Distributed Component Object Model (DCOM) or remote procedure calls (RPCs), the server application responds to client requests. Traditional client/server interactions, shown in Figure 6.1, are often data-centric and combine most (if not all) of the processing (or business) logic and user interface within the client application. The server's task is simply to process requests for data storage and retrieval.

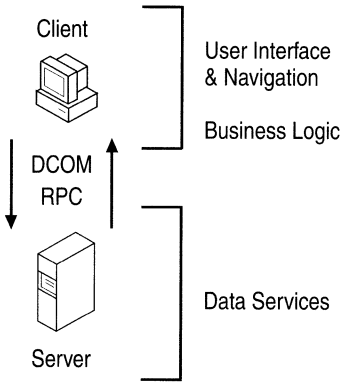


Figure 6.1 Functional Diagram of a Client/Server (Two-Tier) Application

Client/server (two-tier) applications have usually performed many of the functions of stand-alone systems; that is, they present a user interface, gather and process user input, perform the requested processing, and report the status of the request. Because servers only provide access to the data, the client uses its local resources to process it. Out of necessity, the client application can tell where the data resides and how it is laid out in the database. Once the server transmits the data, the client is responsible for formatting and displaying it to the user.

The primary advantage of two-tier applications over monolithic, single-tier applications is that they give multiple users access to the same data simultaneously, thereby creating a kind of interprocess communication. Updates from one computer are instantly available to all computers that have access to the server.

However, the server must trust clients to modify data appropriately—unless data integrity rules are used, there is no protection against errors in client logic. Furthermore, client/server connections are hard to manage—the server is forced to open one connection per client. Finally, because much of the business logic is spread throughout a suite of client applications, changes in business processes usually lead to expensive and time-consuming alterations to source code.

Although two-tier design still continues to drive many small-scale business applications, an increasing need for faster and more reliable data access, coupled with decreasing development time lines, has persuaded system developers to seek out a new distributed application design.

Multi-Tier Design

The new system design logically divides computing tasks across the application. Viewed from a purely functional standpoint, most applications perform the following three main tasks: gathering user input, storing the input as data, and manipulating the data as dictated by established operational procedures. These tasks can be grouped into three or more tiers, which is why the new system design provides for three-tier, or multitier applications. The application tiers, shown in Figure 6.2, are:

- **Client Tier** The user interface or presentation layer. Through this topmost layer, the user can input data, view the results of requests, and interact with the underlying system. On the Web, the browser performs these user interface functions. In non-Web-based applications, the client tier is a stand-alone, compiled front-end application.
- **Middle Tier** Components that encapsulate an organization's business logic. These processing rules closely mimic everyday business tasks, and can be single-task-oriented, or part of a more elaborate series of tasks in a business workflow. In a Web application, the middle tier might consist of Microsoft® Component Object Model (COM) components registered as part of a transactional application or instantiated by a script in Active Server Pages (ASP).
- **Third Tier** A database management system (DBMS) such as a Microsoft® SQL Server™ database, an unstructured data store such as Microsoft® Exchange, or a transaction-processing mechanism such as Transaction Services or Message Queuing. A single application can enlist the services of one or more of these data providers.

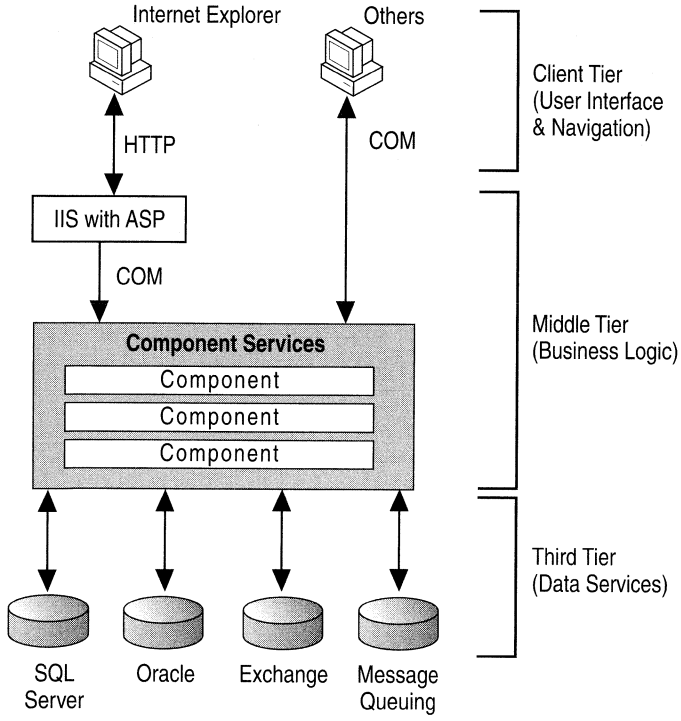


Figure 6.2 Three-Tier Architecture on the Web

Application tiers don't always correspond to physical locations on the network. For example, the middle and third tiers may coexist on the same server running both IIS 5.0 and SQL Server, or they could be separate. The middle tier alone may tie together several computers, and sometimes the server becomes a client itself.

Separating the application into layers isolates each major area of functionality. The presentation is independent of the business logic, which is separate from the data. Designing applications in this way has its tradeoffs; it requires a little more analysis and design at the start, but greatly reduces maintenance costs and increases functional flexibility in the end.

The explosive growth of the Internet is a strong motivation for organizations to adopt *n*-tier architectures in their products. However, organizations still face challenges. How can they take advantage of new technologies while preserving existing investments in people, applications, and data? How can they build modern, scalable computing solutions that are dynamic and easy to change? How can they lower the overall cost of computing while making complex computing environments work? One solution is Microsoft® Windows® Distributed interNet Applications (DNA).

Windows DNA

Windows DNA architecture is Microsoft's framework for building a new generation of *n*-tier computing solutions. Windows DNA provides a framework for delivering solutions that meet the requirements of corporate computing, the Internet and intranets, and global electronic commerce, while reducing overall development costs.

The heart of Windows DNA is COM. Windows DNA architecture makes use of a common set of services, including Hypertext Markup Language (HTML) and Dynamic HTML (DHTML), Microsoft® ActiveX® controls, COM components, client-side and server-side scripting, transactions, security and directory services, database and data access, systems management and HTML, and component authoring environments. These services are exposed in a unified way through COM, which enables applications to interoperate and share components easily.

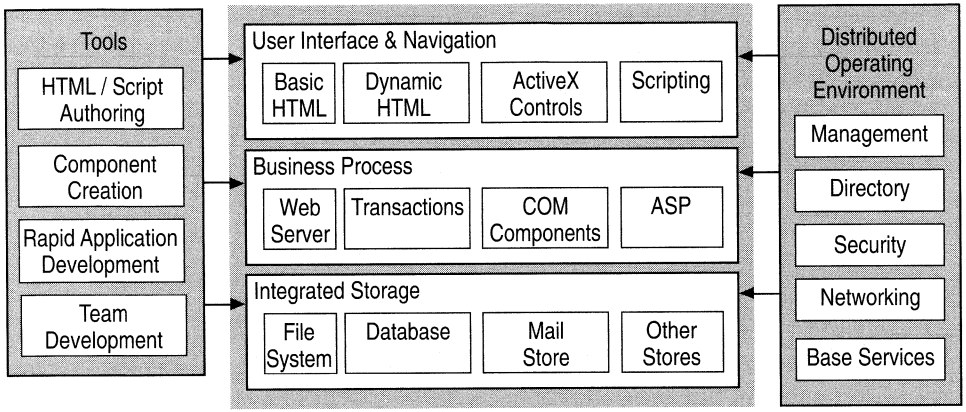


Figure 6.3 The Windows DNA Family of Technologies

Windows DNA builds on the client-side services of the Microsoft® Windows® operating system and Microsoft® Internet Explorer, on the distributed infrastructure of Microsoft® Windows® 2000 Server and the Microsoft® BackOffice® family, and on the company's integrated tools, such as the Microsoft® Visual Studio® development system. Because Windows DNA architecture uses open protocols and published interfaces, organizations can integrate third-party products and solutions. In addition, because Windows DNA architecture embraces an open approach to Web computing, it builds on the many important efforts at developing standards approved by bodies such as the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).

For more information about Windows DNA, see http://www.microsoft.com/dna/.

The Future of Applications on the Internet

Customers are beginning to demand global access to the information they need, both public and personal. Users increasingly want to use a single client application for their information access needs, and they rely on the versatility of the network and servers to provide content and services. Users will come to depend on these applications and want them to be universally available; they might even want to *replace* local applications on their desktop systems.

Consequently, there is likely to be an explosion of HTML-based server applications to feed the ubiquitous availability of the powerful Internet client. Applications will be factored into user interface–only client components (with little software required beyond the standard Internet browser), and a middle tier of server components that have no user interface and that provide services to the local desktop or across the Internet.

The following sections describe the roles that the client and middle-tier play in distributed Internet and intranet applications. The third tier is discussed in "Data Access and Transactions" in this book.

Client-Side Technologies

This section presents a survey of the technologies that make up the client tier of today's Web applications. Each technology is considered from the perspective of what it is, how it works as part of a Web application, and what the issues are regarding its use. This section does not discuss how to develop applications using client-side technologies.

Text and HTML

HTML is the basic formatting language of Web pages. Just like a printed page, text on a Web page can include a variety of font faces, colors, font weights and attributes, spacing, and columns. In addition, Web pages can include tables, frames, and HTML forms. Web applications make heavy use of tables and forms to display data, organize application elements, and collect user input.

HTML adheres to a set of standards that makes it usable over the entire Internet, as well as on intranets. For more information about HTML standards, either see the W3C home page at http://www.w3.org or consult your HTML reference materials. For information about the newest extensions to HTML, see "Dynamic HTML" later in this chapter.

Graphics and Multimedia

Graphics and multimedia, if used effectively, can greatly enhance the look and feel of an application. They can instruct, as well as draw the eye to important areas of the screen.

Multimedia is an especially powerful tool on an intranet. For example, by using streaming audio/video, like that available through Microsoft® NetShow™, you can broadcast special events as they happen, or use prerecorded video to train employees in complex technical operations.

Because of the speed limitations of modems used for most Internet connections, heavy use of graphics and multimedia over the Internet should be restricted. Low-resolution graphics, if designed correctly, not only download more quickly but may actually look better than high-resolution graphics on most computer monitors.

Hyperlinks

Hyperlinks connect the parts of your application together, act as the application's "menu," and can perform both client-side and server-side actions. For example, clicking a hyperlink can cause a page to load in another frame, or can run a client-side script to change the layout of the page.

Hyperlinks are normally embedded directly on the page as text or graphics (such as an image map) where the user can view and click them. They can also be activated when a form is submitted, or client-side script can dynamically create and trigger them.

There are lots of ways to present links to the user. You can simplify the layout of a large number of hyperlinks by grouping similar choices together, by using a similar style of presentation, or by hiding and displaying links as appropriate. For instance, you can choose to display links dynamically, based on the privilege level of your users. Only visitors with high-level access would be able to view links that perform advanced or administrative actions.

Client-Side Script

Client-side scripts run within the user's browser, using the processing power of the user's (client) computer. They can be written in any language supported by the browser; the most common is JavaScript, which is supported by most browsers. Some browsers, such as Internet Explorer, also support Microsoft® Visual Basic® Scripting Edition (VBScript). Client-side scripting enhances Web pages with a variety of custom capabilities. For example, you can use scripts to perform field edits and calculations, manipulate the client window, or validate form input. Scripts normally appear directly on the page they affect, but they can be used to manipulate the content of pages in another frame or browser window as well. For more information about browser support for client-side technologies, see Table 6.1.

ActiveX Controls

ActiveX controls can be used either to customize the user interface, or as "plug-in" applications (such as the Macromedia Shockwave animation control and the RealNetworks streaming audio/video player). ActiveX controls can perform a variety of tasks, from navigation to real-time interaction with stock quotes. They can be written in any language that supports COM Automation, including Microsoft® Visual Basic®, C++, Java, or even COmmon Business Oriented Language (COBOL).

ActiveX controls can be embedded into the HTML page by using the HTML <OBJECT> tag. If the control does not exist on the user's system, it can be downloaded using the URL specified in the CODEBASE attribute (see the following example). The <OBJECT> tag also supports component versioning. Once the control is downloaded and installed, the browser continues to use the cached control until an updated version is available on the server. The following example demonstrates the CODEBASE attribute:

```
<OBJECT ID="BoomButton" WIDTH=225 HEIGHT=35
   CLASSID="clsid:56F1BF40-B2D0-11d0-A6D6-00AA00A70FC2"
   CODEBASE="http://domain.microsoft.com/AControl.cab#Version=1,0,0,1">
</OBJECT>
```

A malicious ActiveX control could perform potentially destructive actions on the user's computer, such as erasing data from the hard drive. To help users determine whether a control is safe to use, Microsoft has developed security guidelines for vendors to follow when releasing a control. A control should identify its creator with a "signature" issued by a well-known security authority, such as VeriSign. Microsoft® Authenticode™, the company's code-signing technology, assures accountability and authenticity for software components distributed on the Internet. Only the original owner can modify a signed control, which prevents tampering by third parties. (For more information about Authenticode and code-signing, see http://www.microsoft.com/security/default.asp.)

As of this writing, only Microsoft® Internet Explorer 3.0 or later includes native support for ActiveX controls. Because of this, ActiveX controls are probably most useful for intranet sites or sites created especially for Internet Explorer users.

Case Study of a Web Application

Microsoft recently introduced a new means of filing employee expense reports. The old system required employees to prepare expense report forms, attach receipts, and submit them to their managers, who would review the forms and submit them to the accounting department. Mistakes were common, and forms often had to be resubmitted. Once the paperwork was finished, the reports were painstakingly entered into a database.

To eliminate some of the problems with the existing system, the accounting department introduced a Web application to control and streamline the employee reimbursement process. The new application allows the employee to report expenses using a Microsoft® Excel worksheet modeled after the paper version of the old form. The worksheet, after it has been downloaded to the user's browser, validates the data as it is submitted, catching most user errors up front. When the documents are ready, the electronic form can be routed to the employee's manager by e-mail. After the manager approves the form, a copy is returned to the accounts department and an approval notification is sent to the employee. The accounting department then performs all of its final work online, saving considerable time and effort.

The new expense-reporting system effectively:

- **Improved Control** Control was a recurring theme throughout the design of the application. The worksheet and associated Web site were carefully designed to ensure that both employees and managers understood their responsibilities. Using an electronic form meant that reports could be tracked on their way through the system, making expense auditing on the back-end faster and more verifiable.
- **Reduced Labor** Because there was less paperwork to be processed, the company was able to improve control while reducing the labor required to process and audit expense reports.
- **Decreased Resource** Part of the goal was to reduce paper waste. The online system requires fewer forms, and hence fewer resources.
- **Decreased Payment Cycle Time** The old process took 8 to 10 days from time of approval to employee reimbursement. Approval sometimes required weeks. The online solution enables a much faster turnaround time. In most cases, payment can be made within one or two days of approval.

Payoffs such as these are a common theme in most Web applications. A well-designed application can improve the way you work by being available wherever there's a browser.

Cascading Style Sheets

The Cascading Style Sheet (CSS) standard gives authors more control over fonts, sizes, two-dimensional overlapping, and exact glyph positioning (for rendering scripts and fonts of various languages, such as the diacritical marks used in Vietnamese or the calligraphic script of Urdu). CSS also separates formatting information from the Web page content, making it much easier to design and revise pages.

Style sheets control the appearance of HTML tags; they do not replace them. Style sheets give you the ability to attach style information to one or more HTML documents and to any of the tags within them, which greatly expands your control over the appearance and structure of each page. Formatting information can be applied to custom tags for a given browser, as well as to standard HTML tags.

CSS information can be specified by linking, embedding, or as an inline style modifier. An HTML document can use any combination of these three methods. However, the most common method is linking, because it establishes a basis for embedded and inline style modifications. An example of a link to a style sheet is shown here:

```
<LINK REL=STYLESHEET TYPE="text/css" HREF="./myCustom.css">
```

Note Because of changes to the CSS standards recently adopted by the W3C, the CSS support in Internet Explorer 3.0 is not fully compatible with that found in Microsoft® Internet Explorer 4.0, which supports the new standard. A detailed description of the latest CSS 2.0 standard is available at http://www.w3.org/Style/.

Dynamic HTML

Dynamic HTML (DHTML), which is supported by Microsoft® Internet Explorer 5, is an emerging standard that is more than just an extension of standard HTML. With DHTML, you can easily add advanced functionality that was previously difficult to achieve without client-side controls. For example, you can:

- Hide text and images in your document and keep this content hidden until a given time elapses or until the user wants to view it.
- Animate text and images in your document, independently moving each element from any point to another in the document, following a path that you choose, or that you let the user choose.
- Create a ticker that automatically refreshes its content with the latest news, stock quotes, or other data.
- Create a form; then instantly read, process, and respond to the data the user enters in the form.

Internet Explorer 5 does not require additional support from Java applets or embedded controls to achieve these effects. It automatically reformats and redisplays the DHTML page to reflect dynamic changes in content styles. It does not need to reload the document, load a new document, or depend on the server to generate new content. Instead, it uses the power of the user's computer to calculate and carry out changes.

DHTML documents make heavy use of styles and script to process user input and directly manipulate the HTML tags, attributes, and text in the document. Through the Internet Explorer 5 object model, you can control every property of every HTML tag to precisely control the layout, appearance and function of your page.

> **Note** Not all the features of DHTML are compatible with all browsers. Web pages intended for viewing by browsers other than Internet Explorer 5 should be tested for compatibility with other browsers.

For more information about DHTML, see http://msdn.microsoft.com.

Data Binding

Using DHTML, the results of database queries can be "bound" to HTML elements, such as the rows of a table. (You can also use data-binding ActiveX controls, such as the Advanced Data Connector (ADC), included in earlier versions of Internet Explorer.) Data Binding allows you to remotely view and modify the results of database queries within the browser. It is a function of Remote Data Services (RDS), which is part of the Microsoft® ActiveX® Data Objects (ADO) family of data access components.

> **Note** Data binding is supported if you are using Microsoft® Internet Explorer 4.0 or later.

For more information about RDS and ADO, see "Data Access and Transactions" in this book or see the IIS 5.0 online product documentation. For more information about data binding, see http://msdn.microsoft.com.

Browser Support

In intranet scenarios where a single browser type can safely be assumed, you can design your sites around browser-specific technologies with impunity. (Just in case someone is an errant browser user, you should alert your users with a "Best viewed with" graphic on the site's home page.)

On the Internet, however, you can't assume that everyone has an up-to-date browser. And, even among newer browsers, several different types are available; Microsoft, Netscape, and Sun Microsystems have all released browsers with varying degrees of support for ActiveX, Java, scripting, and HTML. The question of what functionality to perform on the client depends on the variety and capabilities of browsers you want to support.

Using the lowest-common-denominator approach, pages contain no more functionality than the least capable browsers can process successfully. Content is guaranteed to be viewable in its entirety on any browser. Unfortunately, users might notice, and be disappointed by, the limited functionality this approach requires.

Some sites provide text-only versions of their pages, or frames-free areas for less capable browsers. This duplication ensures that all users get the same information, but it requires that you develop, test, and maintain multiple versions of your site. Often the less functional version remains underdeveloped, as the focus of development tends toward "bells and whistles."

The best approach may be to develop pages using basic technology, such as HTML and JavaScript, and to add specific features once you have determined the browser type. This is often a good middle ground, because all pages can be developed using one set of design elements and content. The advanced features of the Web site, such as data binding, are made available only to browsers that support them.

The Browser Capabilities component included in the scripting environment of ASP provides a way to detect the browser type and to tailor the returned document in order to exploit browser-specific capabilities. For more information about this component, see the IIS 5.0 online product documentation.

Table 6.1 summarizes browser support for different client technologies.

Table 6.1 Support for Client Technologies by Browser

Technology	Widely Supported	Internet Explorer 3.0*x*	Internet Explorer 4.0 and 5
HTML	X	X	X
Graphics and multimedia	X	X	X
Hyperlinks	X	X	X
JavaScript	X	X	X
VBScript		X	X
ActiveX controls		X	X
Cascading Style Sheets		X	X
Dynamic HTML			X

Limitations of Client Technologies

Although it's possible to create applications that rely exclusively on client-side technologies, they are somewhat limited in their capabilities, just as client/server architecture is. There are several reasons why client/server architecture isn't suitable for full-scale enterprise applications on the Internet:

- A client application using client-side ActiveX controls or client-side scripting is not supported by all browsers. A line-of-business application for the Internet must work with as many browsers as possible, including those that do not support HTML tables, frames, Java applets, client-side scripting, or ActiveX controls.
- Coding business logic as client-side script fails to protect your programming investment (because the source code is available to all). Java applets and ActiveX controls are more secure, but whenever you combine business logic with the user interface, your application becomes harder to support and to debug. In addition, the resulting components are less likely to be reusable in other applications.
- Client-centric applications do not take full advantage of the three-tier programming model. Designs in which the client plays more than a supporting role typically take on tasks that are better suited for the server, such as resource management and data manipulation.

The Middle Tier

This section discusses what is perhaps the most important layer of Web programming, the middle tier. It is here that user input is combined with business logic to perform the work of your site.

The middle tier is not always just a single layer of logic. It can consist of many interrelated technologies, seamlessly combined to create the illusion of a single multipurpose layer. For example, the client's request could be preprocessed by an Internet Server Application Programming Interface (ISAPI) filter, then execute a script to run a custom-built component that manipulates a database with ADO. Technology integrates with technology, layer upon layer: a demonstration of true *n*-tier architecture.

Middle-tier technologies discussed in this section include CGI (Common Gateway Interface) applications, ISAPI extensions and filters, ASP, process isolation and crash recovery, and others.

CGI Applications

In the past, Web server application programming would usually require developing CGI programs or scripts.

CGI applications are most widely used on UNIX systems to create executable programs that run on the Web server. CGI programs are typically written in the C language, but can also be written in interpreted languages such as Perl. Remote users can start CGI applications on the server simply by requesting a URL containing the name of the CGI application. Arguments following the question mark in the URL are passed to the CGI application as environment strings. The output of a CGI application isn't much different from a desktop application; HTTP headers and HTML are generated using the basic output functions of the language (for example, `printf` in C).

CGI applications are easy to write, but scale very poorly on the Windows operating system. Because a separate process is spawned for each client request, hundreds of clients create hundreds of instances of the CGI program, each requiring its own memory space and system resources. This isn't such a bad thing on UNIX, which is designed to handle multiple processes with very little overhead. However, Microsoft® Windows® 2000, which is optimized for thread management inside a process, expends more system resources when creating and destroying application instances. As a result, ISAPI was developed specifically for IIS 5.0 as a high-performance Windows alternative to CGI.

ISAPI Extensions and Filters

An ISAPI *extension* is a run-time dynamic-link library (DLL) that is usually loaded in the same memory address space occupied by IIS 5.0. Since it is a DLL, only one instance of the ISAPI extension needs to be loaded at a time. Of course, the ISAPI extension must be thread-safe, so that multiple client requests can be received simultaneously.

Although ISAPI extensions are more complex than CGI applications, ISAPI uses a relatively simple Application Programming Interface (API). For each client request, the Web server invokes the **HttpExtensionProc** ISAPI call and passes a pointer to an ISAPI Extension Control Block (ECB), which contains information about the request. The ISAPI DLL can use server callback functions to access information such as server variables. The ISAPI ECB also provides the developer with access to some general-purpose support functions, such as URL redirection, session management, and response headers, which are not available to CGI applications.

Despite the obvious benefits over CGI, ISAPI extensions present some maintenance problems. For instance, if you want to make even a minor change to the HTML returned by an ISAPI extension, you have to recompile and link it. Also, an ISAPI DLL can cause the Web server to crash, if it hasn't been thoroughly tested and verified before being deployed and run in the Web server process.

You can select which ISAPI extensions are loaded in process with IIS 5.0 and which extensions should be loaded in a separate process. ISAPI extensions in a separate process can be stopped and restarted independently of the server process, and can be restarted automatically after a crash. Although out-of-process extensions are slower than in-process ones, being able to isolate and reload applications that are under development offers advantages in service reliability. For more information about out-of-process extensions, see "Data Access and Transactions" in this book.

ISAPI can also be used to create ISAPI *filters*. Filters are a fairly new concept in Web server extensibility—there is no CGI counterpart. ISAPI filters can intercept specific server events before the server itself handles them. The calling convention for filters is very similar to that of extensions. When a filter is loaded (usually as the Web service starts), it indicates what sort of event notifications it will handle. If these events occur, the filter has the option of processing the events, passing them on to other filters, or sending them to the server. In this way, you can use ISAPI filters to provide custom authentication techniques, or to automatically redirect requests based on HTTP headers sent by the client, such as **Accept-Language**.

However, filters can degrade performance, if they are not written carefully. With IIS 5.0, ISAPI filters can be loaded for the Web server as a whole or for specific Web sites. However, they cannot be run out of process.

Active Server Pages

The ASP scripting environment greatly simplifies server-side programming so that you can easily create dynamic content and powerful Web-based applications.

Scripts in ASP pages can perform the same sorts of tasks as CGI and ISAPI applications, but are much easier to write and modify. ASP creates a higher level of interactivity by managing application and session state on the server, thereby reducing the amount of information that needs to be transmitted back and forth between the server and the client. ASP makes it easy to work with information entered into HTML forms or in the URL as parameters. You can also control advanced HTTP features from client-side cookies and client security certificates. See the IIS 5.0 online product documentation for more information about advanced features.

At the heart of ASP is an ISAPI extension—Asp.dll— that compiles and caches .asp files in memory at run time using a script interpreter. The IIS 5.0 *script map* associates the .asp extension to Asp.dll. Because ASP must interpret and compile scripts before executing them, complex scripts can be about four times slower than plain HTML, and two to three times slower than ISAPI, *when they are first requested.* Afterward, the compiled version of the page is cached in server memory, making subsequent requests significantly faster and amortizing the initial cost of compilation over potentially thousands of page requests.

ASP is designed for usability and ease of development, giving you the opportunity to dramatically decrease time spent in development. However, it will never outperform static content, or custom-written, task-focused C++ ISAPI extensions. Only carefully designed ASP applications, combined with server-side components, can approach the speed and performance of ISAPI applications. For information about designing ASP pages, see the "Building ASP Pages" topic in the IIS 5.0 online product documentation.

ASP Server-Side Scripting

You can create highly interactive pages that are independent of the type of browser used to access those pages. Unlike client-side script, with ASP you can "hide" your scripting on the server, enabling you to protect your development ideas and intellectual property.

The ASP scripting environment is "language agnostic," meaning that it isn't limited to a particular language. VBScript, Microsoft® JScript®, or any language for which a third-party ActiveX scripting engine is available (such as PerlScript, REXX, or Python) can be used to create scripts in ASP pages.

ASP scripting instructions appear side-by-side with HTML. (In fact, you can create an ASP page simply by changing the file extension of a file in plain HTML to .asp.) To differentiate between HTML and script meant to run at the server, ASP uses special tags, called *server-side scripting delimiters*, to indicate server-side script: <% and %>. Script appearing inside these delimiters will be invoked on the server as the page is processed. A special form of these scripting delimiters, <%= expression %>, can be used as a shorthand for returning values from script.

The following line of server-side VBScript code returns the current date:

```
Today is <%= Date %>.
```

This instruction generates something like the following line (the exact text depends on the date):

```
Today is 7/4/99.
```

> **Note** You can also use the **Write** method of the ASP **Response** object to display text or the results of an expression on the page.

A slightly more complex example of ASP might use the conditional execution elements of the scripting language, intermixed with HTML, as follows:

```
<% If Hour(Now) < 12 Then %>
   <FONT COLOR=YELLOW>Good Morning!</FONT>
<% ElseIf Hour(Now) < 18 Then %>
   <FONT COLOR=LIME>Good Afternoon!</FONT>
<% Else %>
   <FONT COLOR=ORANGE>Good Evening!</FONT>
<% End If %>
```

Although each page has a primary scripting language, you can use more than one scripting language on a single ASP page. You can set the primary scripting language for an application in Internet Services Manager. Use *declaratives* (also known as *@-directives*) to define the primary scripting language for a page. For more information about declaring scripting languages, see the IIS 5.0 online product documentation.

ASP subroutines and functions can be in any script language, although if you define them inline with the rest of the script, you're limited to the primary scripting language. To change the scripting language of the subroutine, you need to use HTML <SCRIPT> tags to define them. You will also need to add the RUNAT=SERVER attribute, to indicate that this script is intended for the server rather than for the client.

The following sample page demonstrates how to combine a variety of scripting languages and subroutine declaration styles into a single ASP file:

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD>

<% Sub InlineSub %>
This text won't be displayed until this subroutine is called.<br>
<% End Sub %>

<SCRIPT LANGUAGE="VBScript" RUNAT=SERVER>
   'Immediate script (outside a function).
   Response.Write "This text is displayed last"
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript" RUNAT=SERVER>
function TestJavaScript(str) {
   Response.Write(str);
}
</SCRIPT>

<SCRIPT LANGUAGE="PerlScript" RUNAT=SERVER>
sub TestPerlScript {
   $Response->Write($_[0]);
}
</SCRIPT>
</HEAD>

<BODY BGCOLOR=#FFFFFF>
<%
   Response.Write "This is VBScript<br>"
   TestJavaScript "This is JavaScript<br>"
   TestPerlScript "This is PerlScript<br>"
   InlineSub
%>
</BODY>
</HTML>
```

Note You can use <SCRIPT> tags to enclose immediate server-side script, but don't expect it to be run until the entire page has been processed. Earlier versions of ASP used <SCRIPT> tags before the introduction of the <% %> delimiters, but their use for immediate script is now discouraged. You should reserve server-side <SCRIPT> tags for defining functions and subroutines only.

Execution Behavior of Scripts in ASP Pages

When you write ASP applications, you're operating in the world of IIS 5.0 and HTTP. Web developers who don't have a firm grasp of this architecture find themselves puzzled by strange errors in what seems to be straightforward code.

Consider the process shown in Figure 6.4.

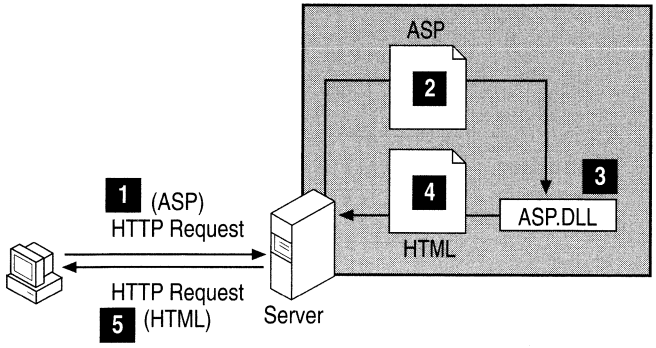


Figure 6.4 Requesting an .asp File from IIS 5.0

When a client browser requests an ASP page, a number of events occur in the following sequence:

1. The client requests an ASP page by sending an HTTP Request to the Web server.
2. Because the page has the .asp extension, the service (IIS 5.0) recognizes it as a script-mapped file, and sends the file to the appropriate ISAPI extension (in this case, to Asp.dll) for processing. (This step does not occur when the client requests an HTML file.)
3. The ASP ISAPI processes any server-side include directives first, before any server-side script is compiled. Next, the script is executed, and dynamic text, if any, is incorporated into the page that will be returned to the client. (This step only happens when the page is first requested. Previously compiled pages are retrieved from a server-side cache for faster performance.)
4. The server creates the resulting HTML page to be sent back to the client. The page output is sent incrementally as the page is generated, or all at once if the response is buffered.
5. Once the client receives the ASP page, it loads any client-side objects and Java applets, executes any immediate client-side script code and displays the Web page according to the HTML specification.

While this process looks simple, keep in mind that the client and server could be hundreds, or even thousands, of miles apart. Therefore, when a problem arises, you must determine *where* the error is occurring. Is it on the client or on the server? Equally important is understanding *when* each operation takes place. After ASP completes its processing in step 3 and the server sends the response in step 4, it moves on to other activities and other clients. The only way the client can recapture the server's attention is to request another page via the HTTP protocol. In other words, there is no real connection between the client and server. This is a very important concept.

Sometimes developers try to access server-side scripts or objects from the client, or conversely, client-side objects or scripts from the server. For example, consider client-side code that attempts to access one of ASP's built-in objects, such as the **Session** object. The attempt is destined for failure, because the code running on the client has no way of accessing an object located on the server. A typical error message might appear as follows:

```
VBS Script Error: Object Required: Session
```

Now consider an example in which a server-side script attempts to manipulate a client-side object. Suppose the developer wants to use server-side script to populate a client-side control called ListBox1, using the following instruction:

```
<% ListBox1.AddItem Value1 %>
```

The problem is that the HTML page, including the list box, does not yet exist when the server-side code is executed. Therefore, this instruction generates an error.

On the other hand, you can use server-side code for generating client-side code in order to populate a list box. For example, you could create a **Window_OnLoad** event, which is executed by the browser as soon as the window and its child controls are created. The following code uses server-side script to provide the **AddItem** method with values stored in the variables *Value1*, *Value2*, and *Value3*.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Window_OnLoad()
   ListBox1.AddItem "<%= Value1 %>"
   ListBox1.AddItem "<%= Value2 %>"
   ListBox1.AddItem "<%= Value3 %>"
End Sub
-->
</SCRIPT>
```

> **Note** If you use the HTML <SELECT> tag instead of an ActiveX control, the procedure is slightly more direct. Because a list box created with the <SELECT> tag is based on HTML code, you can use server-side scripting to generate the <OPTION> tags. Since the HTML is self-contained, you do not need to place any code inside a **Window_OnLoad** event.

Built-in Objects and Server-Side Components

If you have ever written client-side script, you have probably found yourself using built-in browser objects such as **document**, **form**, and **window**. These objects are provided as part of the browser's object model, and make interaction with the browser much more manageable. Likewise, ASP defines its own object model.

The **Response** object mentioned earlier is one of ASP's built-in objects, of which there are currently six: **Server**, **Application**, **Session**, **Request**, **Response**, and **ObjectContext**. These objects greatly simplify the interaction between the server and the client. A description of each appears in the sidebar, "The Built-in ASP Objects."

In addition to built-in ASP objects, you can create and manipulate a variety of custom server-side components. By combining active scripting with server-side COM components, also known as *server-side objects*, you extend the functionality of ASP with powerful, easy-to-use packages. You can instantiate server-side components by using the **CreateObject** method of the **Server** object and passing it the *ProgID* of the component you wish to create. Once the component is instantiated, you can access any of its properties or methods. For example, the following script instantiates a server-side object with **Server.CreateObject** and stores a reference to it in the variable *objAdRotator*:

```
<% Set objAdRotator = Server.CreateObject("MSWC.AdRotator") %>
```

The Built-in ASP Objects

ASP provides built-in objects that make it easier for you to gather information sent with a browser request, to respond to the browser, and to store information about a particular user.

Server Object

You use the **Server** object to access methods and properties on the server. The most frequently used method is the one that creates an instance of a COM component (**Server.CreateObject**). Other methods apply URL or HTML encoding to strings, map virtual paths to physical paths, and set the time-out period for a script.

Application Object

You use the **Application** object to store global application settings and to share information among all users of a given ASP application.

Session Object

You use the **Session** object to store information needed for a particular user session. Variables stored in the **Session** object are not discarded when the user jumps between pages in the application; instead, these variables persist for the entire time the user is accessing pages in an application. You can also use **Session** methods to explicitly end a session and to set the time-out period for an idle session.

Request Object

You use the **Request** object to gain access to any information that is passed with an HTTP request. This includes name/value pairs passed from an HTML form using either the POST method or the GET method, cookies, and client certificates. The **Request** object also gives you access to binary data sent to the server, such as file uploads.

Response Object

You use the **Response** object to control the information you send back to a user. This includes sending information to the browser, redirecting the browser to another URL, or setting cookie values.

ObjectContext Object

You use the **ObjectContext** object to either commit or abort a transaction initiated by a script in an ASP page. For more information, see "Data Access and Transactions" in this book. You can also use this object to access the other built-in ASP objects from within a component.

To get you started, the default ASP installation includes several task-oriented components, including the MyInfo, Ad Rotator, and Browser Capabilities components. When you install the Microsoft® Data Access Components (MDAC), included with IIS 5.0, you can also use ADO to access information stored in a SQL Server, Oracle, Microsoft® Access, or other database.

Why Components?

You could spend a lot of time writing scripts in ASP pages that emulate component functionality, but there are several reasons not to do so:

- Script is much slower than a compiled object and will be less likely to scale to large numbers of users.
- Script doesn't separate presentation from functionality. Undifferentiated script scatters business logic throughout the application, making it hard to find bugs and increasing the cost of maintenance.
- Components are inherently reusable; scripts are not. Components may also be used by other applications, such as those built in Visual Basic or C++.

Therefore, your development motto should be: The less script, the better. If you are serious about performance and application scalability, you should use components to perform the bulk of your business logic.

So, what makes a good server-side object? Generally, anything that expands the functionality of the server and scripting language is a candidate for a server-side object. Some server-side objects generate HTML that cannot be easily generated by ASP itself. Others perform server functions, like accessing the registry, sending mail, or administering a resource.

A compact memory footprint, computational speed, and multiuser re-entrancy are high priorities for server-side objects. Component stability is key, too. Memory leaks affect other applications on the server, and badly behaved components can cause IIS 5.0 to crash.

Because ASP component objects exist on the server, they should never be written to rely on user-interface elements, like dialog boxes or pop-ups. Configure components to send errors to the Event Log of Windows 2000 Server, or return detailed error information in the **Err** object, so problems can be reported to the user with script.

Reuse, Buy, or Build

When it comes time to procure components for your application, reuse the objects you already have if possible. If none of your prebuilt objects will do, consider buying third-party components from a reputable vendor.

If you can't locate a prebuilt component, you will have to build one yourself. Server-side components can be built using any development tool that supports the creation of COM Automation servers, such as Visual Basic, Microsoft® Visual J++®, or Microsoft® Visual C++®. As with scripts, choose the component language that suits your needs. Microsoft® Visual Basic® 6.0 creates "Apartment" threaded components that can be used on a page-by-page basis. If computational speed and multiuser re-entrancy are important to your application, you should use Visual C or Visual C++ with Active Template Library (ATL) 2.0 to develop your component. Visual C and Visual C++ can create "Both" threaded components that are suitable for use in the application and session scope. (For more information about threading considerations, see "Selecting Object Scope" later in this chapter.)

For more information about creating a server-side component, see the "ASP Tutorial" topics in the "Active Server Pages Guide" section of the IIS 5.0 online product documentation.

Building Windows Script Components

As you develop Web applications, especially when using ASP, you will find that your scripts tend to grow beyond your original intentions. This is a natural evolution of the middle tier that often occurs in the planning and development of an *n*-tier Web application. You may also decide to develop COM components in order to handle middle-tier logic for your applications.

ASP supports Microsoft® Windows® Script Components, a technology that provides a way to develop COM components by using scripting languages, such as VBScript, Microsoft® JScript® 2.0, JavaScript 1.1, PERLScript, and other languages compatible with the ECMA 262 language specification. Windows Script Components, which you can use as you would other COM components, provide many benefits in that they:

- Allow you to encapsulate scripted tasks as reusable COM components.
- Enable you to easily develop prototype COM components to be converted using a suitable programming language, after you are satisfied with your component's design and functionality.
- Provide access to a wide range of system services, just as other COM components do.
- Are small and efficient.
- Are easy to create, maintain, and deploy.
- Are supported by Component Services, a run-time environment for COM components.

Windows Script Component technology consists of:

- The script component run time (Scrobj.dll).
- Interface handlers, which extend the run time. These are compiled components that implement specific COM interfaces. When you install the script component run time, you receive the Automation interface handler, which allows you to call your script component from an .asp file.
- Your script component file (an .sct file), in which you specify which interface handler you want to use, and which methods can be called from an .asp file in order to accomplish the intended functionality.

ASP Applications

So far, this chapter has presented several examples of scripts in ASP pages that create HTML on the fly. Stand-alone pages, however, are only the beginning of what you can do with ASP. This section discusses how to use the ASP **Session** and **Application** objects to create a coherent application out of an independent series of ASP files. It also walks you through the different elements of an ASP application, and discusses how to configure it using Internet Services Manager.

ASP Session Management

Since HTTP is a stateless protocol, the Web server retains no memory of past actions and treats each browser request as a one-time event. This statelessness makes it difficult for Web developers to create the level of application interactivity that most users expect. Although persistent application state can be maintained in a database or stored in files, such solutions are often difficult to implement, and involve a hefty memory overhead and performance penalty. ASP surmounts these problems by providing its own session management.

Using the ASP **Session** object, one of the ASP built-in objects, developers can store any data they wish, including references to objects they have instantiated. The **Session** object is analogous to an associative array, into which values may be stored and retrieved by using a string (or keyword) as the index.

For example, the following instruction assigns "John Doe" to the `MyName` entry in the array:

```
<% Session("MyName") = "John Doe" %>
```

The value "John Doe" is retrieved from the **Session** object by using the `MyName` keyword, as shown here:

```
My name is: <%= Session("MyName") %>
```

The **Session** object exists in memory on the server and maintains its state for the duration of the user's Web session. You can disable this object on a page-by-page basis using the declarative `<%@ EnableSessionState=False %>`. Disabling the **Session** object does not affect values that have previously been stored in this object. However, it may speed up the processing of pages that don't use the **Session** object. Furthermore, multiple ASP pages in an HTML frameset may be serialized (run in sequential order) if they use session state. Disabling the **Session** object in child frames that don't require it will allow the frame requests to execute concurrently.

The **Application** object is used in a similar fashion to store values that persist for the lifetime of the ASP application; it is also used to share values across user sessions. The values stored in the **Session** and **Application** objects persist between page requests, and can be retrieved at any point using the keyword that they were assigned.

ASP Sessions Are Cookie-Based

ASP uses HTTP cookies to identify user sessions with unique session keys. Once an ASP session begins, ASP responds to a user's request with a Set-Cookie HTTP header. From that point on, each browser request is identified by the session ID cookie.

Web Clusters and ASP Session State

ASP session information is stored in memory on the Web server, which creates a challenge for sites using ASP in a Web cluster environment. Web clusters balance the load of user requests among a number of Web servers. In order to use ASP **Session** management, the *same* Web server must handle all requests from a user for the life of the session, a prerequisite that most load-balancing schemes cannot guarantee.

Several options are available if you want to use ASP in a Web cluster:

- Write your own session management logic to replace that provided by ASP. Keep your session state in a centralized place, such as a database.
- Use a third-party solution. For example, Cisco Systems' LocalDirector hardware load balancer can ensure that the same client gets the same server for multiple connections.
- Load balance new requests but, once a session begins, make sure that all subsequent requests return to the same server during the life of the session. This technique is called ASP session-aware load balancing.

ASP Session-Aware Load Balancing

In this scenario, all new requests continue to use the existing load-balancing mechanisms, such as round-robin Domain Name System (DNS), for distributing requests to a site's published URL. Then, during the **Session_OnStart** event, the script in an ASP page uses the **Response** object to redirect the browser to the application's start page by using the local computer's own Internet Protocol (IP) address or unique name, as follows:

```
Response.Redirect("http://w10.microsoft.com/Webapp/firstpage.asp")
```

Finally, to ensure the browser will confine its requests to the same Web farm server, all application links must be *relative URLs*. Relative URLs only specify path information relative to the current location, for example:

```
<A HREF="MyAsp.Asp">...</A>

<FORM METHOD=POST ACTION="./SubDir/FormAction.asp">...</FORM>
```

When one of these links is selected, the browser will construct the full URL path using the current address (specified in the redirect) and the relative URL, thus enabling it to locate the computer hosting the user session.

This technique may not be appropriate for all Web applications. Also, if users save URLs on their browsers using favorites or bookmarks, they will return to a specific computer, which can defeat the purpose of proper load balancing.

Here's an example of an ASP session ID cookie:

```
Set-Cookie: ASPSESSIONIDGGGGGJZB=EFENHLNDIIEHJGJOAGICNPEK; path=/
```

This is different from earlier versions of ASP, which would set the cookie path to that of the ASP application's virtual directory. In IIS 5.0, only one cookie is sufficient for *all* applications on the server. Once the cookie is received, every browser request includes the same HTTP cookie header:

```
Cookie: ASPSESSIONIDGGGGGJZB=EFENHLNDIIEHJGJOAGICNPEK
```

ASP uses this value to retrieve the correct **Session** object for the client connection. All requests to the application directory include the session ID cookie, even those for static HTML content in subdirectories of the ASP application. In this example, the cookie does not specify an expiration time, so it is only valid as long as there is an open client session. The cookie expires when the user closes the browser.

> **Note** If a user chooses not to accept the ASP cookie (or if the browser fails to return it in subsequent requests), the application will not be able to map the browser request to an existing **Session** object and will create a new one.

It is possible for several browser instances to share the same cookie (which means that more than one browser window can be accessing different sections of your application at the same time), but make modifications to the same **Session** object. This is especially bad for applications that make heavy use of session state, and is another reason to use session state only when necessary.

ASP session IDs are mapped to the in-memory **Session** object on the server. This works well when only one server is managing the user session. When using ASP in a Web cluster, however, more than one server may be handling the user's request. For more information about how to manage session IDs in a Web cluster, see the sidebar, "Web Clusters and ASP Session State" earlier in the chapter.

ASP Session IDs Are Not Unique

Applications that require a unique user identifier should not use the ASP session ID, which is unique only for the life of the current application. If the application restarts, the server may conceivably reassign the same session ID to another user. In a multiple-server environment, like a Web cluster, the likelihood of duplicate IDs increases. Consequently, it is not advisable to use an ASP-issued session ID as a unique key for tables, or for any persistent user identity.

Instead of using the ASP session ID, you must use a separate mechanism designed to create unique numbers across multiple servers and sessions. Component Services includes the component TakeANumber, which can be used to generate unique sequential numbers for user identification. (For more information about this component, see the sidebar "Take A Number: A Component Services Example.")

Take A Number: A Component Services Example

Component Services includes the TakeANumber component designed to produce sequential numbers. Because the number is incremented as part of a transaction, you are guaranteed a unique identifier that works across sessions and across servers.

The TakeANumber component is installed with Component Services, but requires a little preparation to use. You first need to define a SQL Server table in order to store the current number. This table must be named "TakeANumber" and contain two columns named "NextNumber" (integer type) and "PropertyGroupName" (string type). The PropertyGroupName column identifies which counter you are using—more than a single counter can be stored in the table. Once your table is ready, you need to enter the first number of the series. The following SQL statement will accomplish this:

```
INSERT INTO TakeANumber VALUES (1234, 'MyProp')
```

Finally, create a File Data Source Name (DSN) so that the component can connect to the table you just created. You can create a File DSN with the **Data Sources (ODBC)** application (in **Administrative Tools**) in Control Panel. For step-by-step instructions, see "Data Access and Transactions" in this book.

Now you're ready to use the component. The following instructions retrieve the next number from the `MyProp` series:

```
<%@ LANGUAGE=VBScript EnableSessionState=False %>
<HTML>
  <HEAD><TITLE>Take A Number</TITLE></HEAD>
    <BODY BGCOLOR=#FFFFFF>
    <% Set tn = Server.CreateObject("MTS_TakeANumber.TakeANumber") %>
    Next Number: <%= tn.GetANumber ("TakeANumber.dsn", "MyProp") %>
  </BODY>
</HTML>
```

As long as each server in your site connects to the same TakeANumber database, you are guaranteed a unique identifier across servers.

ASP Session ID and Session Security

The cookie approach to session management could become a potential security problem. If an intruder were able to capture or guess the session ID cookie in use by an active session, he or she could submit valid HTTP requests that included this cookie. In this manner, an intruder could hijack, or steal, a user's active session. For example, if a user had supplied valid credit card information, and a script in an ASP page stored this information in the **Session** object, an intruder who managed to hijack the session could make purchases using the stolen session. For this reason, the following built-in security measures are taken when generating ASP session cookies:

- Session ID values are 32-bit long integers.
- Each time the Web server is restarted, a random session ID starting value is selected.
- For each new ASP session that is created, the session ID value is incremented.
- The 32-bit session ID is mixed with random data and encrypted to generate a 16-character cookie string. Later, when a cookie is received, the session ID is decrypted from the 16-character cookie string.
- The encryption key is randomly selected each time the Web server is restarted.

ASP session ID values are selected from a huge range and are encrypted, making it difficult to capture a valid cookie. In addition, guessing a valid cookie once does not make it easy to guess another valid cookie.

If the complexity of the ASP cookie generation algorithm does not meet the security requirements of your site, user authentication and client certificates can be used in conjunction with session management to provide even more security to your Web applications. For more information, see "Security" in this book.

Building an ASP Application

In its simplest form, an ASP application consists of all the HTML and script files stored within an application boundary. Before any sessions are created, the application initializes, instantiates application-scope components, and imports type-library declarations. From that point on, each connected user has a separate and distinct session, with its own values and component instances. The rest of this section explains how application and session management is accomplished.

Application Boundaries

An ASP-based application consists of all the files in its root virtual directory and in any subdirectories. An application defines a *namespace* (also called the *application root*), that begins at the root directory and includes all files, directories, and virtual directories contained within—except for those that are application roots themselves or ancestors of another application root. For example, if a virtual directory "Applications," and its subdirectory "Isolated Applications," are both application roots, then URLs that contain only "/Application" are part of one application, and URLs that contain "/Application/Isolated Application" are part of the other. Figure 6.5 illustrates how this looks in Internet Services Manager.

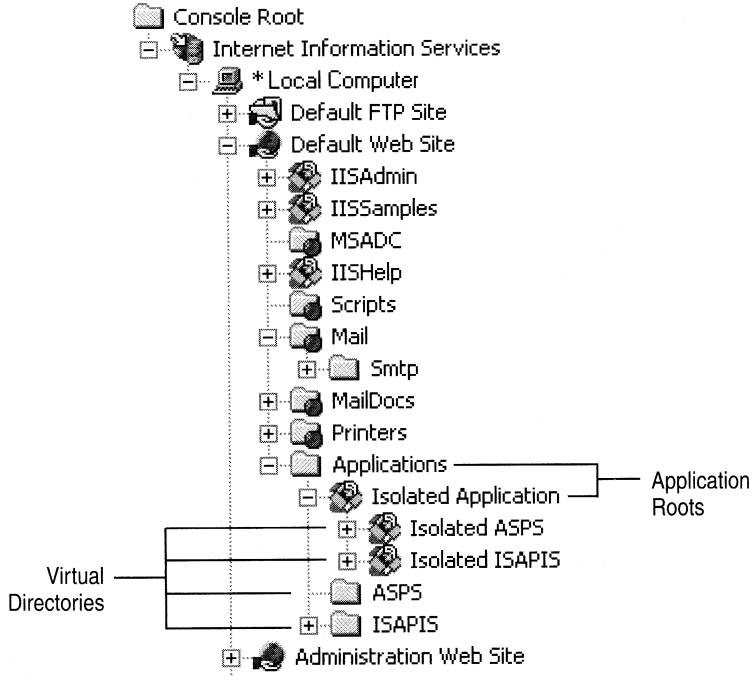


Figure 6.5 Application Roots and Virtual Directories

In Figure 6.5, the virtual directories—ASPs and ISAPIs—are contained within the Applications namespace. The Isolated ASPs and Isolated ISAPIs virtual directories are part of the other application.

Application developers can enforce a logical division between applications with separate application roots. Any *Application* and *Session* variables created in an application's namespace are segregated from the *Application* and *Session* variables of other applications on the server. There is currently no way to view values from other applications.

Global.asa

Global.asa is used to store information used globally by the application; it does not generate content that is displayed to users. Global.asa must be stored in the starting point (normally the root directory) of the application. An application can have only one Global.asa file.

Global.asa files contain only the following: **Application** events, **Session** events, object declarations, and type library declarations. If you include text that is not enclosed by <SCRIPT> tags, or define an object that does not have session or application scope, the server returns an error. The server ignores tagged script that the application or session events do not use, as well as any HTML in the file.

Application and Session Events

Every application has two events associated with it: **Application_OnStart** and **Application_OnEnd**. The script for these events is defined with server-side <SCRIPT> tags within Global.asa. Event script can be written in any language supported by the server. **Application_OnStart** is called once for each application, when the first client makes a request for a page within the application boundaries. The **Application_OnStart** event procedure is a good place to set global state variables and to create any objects that will be used by all users of the application.

After the **Application_OnStart** event, and for each subsequent new session, the **Session_OnStart** event occurs. An ASP application should use the **Session_OnStart** event to perform any required session initialization tasks. At this point, a **Session** object and a **Request** object exist. The **Session** object includes a unique session ID; the **Request** object includes fully parsed collections of values passed by the browser, as well as the server environment variables.

The **Session_OnStart** event procedure is a good place to redirect to the start page of your application. If you don't redirect, the application begins execution with the document requested in the URL, or it will begin with one of the default documents configured in Internet Services Manager for the application's virtual root. Redirection allows you to control where your application will go first.

Since the **Response.Write** method is not available during the processing of event procedures, you won't immediately be able to report errors if they occur. However, if you want to notify the user of any problems, you can save the error text in the **Session** or **Application** object (depending on which event handler caused the error) and report it on the first page that is loaded thereafter.

A session ends either when it times out or when the **Session.Abandon** method is called. When this happens, the **Session_OnEnd** event procedure is called, giving you the chance to destroy any object references, and perform any other session cleanup. Of the server built-in objects, only the **Application**, **Session** and **Server** objects are available to the **OnEnd** event handlers. Additionally, you can't use the **MapPath** or **CreateObject** methods of the **Server** object during the **Session_OnEnd** event.

Ending the ASP Session

Unless you have provided a means for explicitly logging off, there is no way to determine if the user is still actively connected to your application. HTTP is a stateless protocol, and doesn't keep track of user connections.

For this reason, ASP provides a mechanism to close a session when a specified time-out period expires. If a user begins a session but stops making requests to the Web application, ASP automatically triggers the **Session_OnEnd** event. The time-out period defaults to 20 minutes, but can be adjusted by setting the **Timeout** property of the **Session** object. You can also change the default value.

To change the default value

1. Right-click the application's virtual directory in Internet Services Manager, then click **Properties**.
2. Click the **Configuration** button, and select the **App Options** tab.
3. Type a value in the ASP Script time-out box.

For applications that cache a database connection or consume a lot of server resources, the session time-out period may represent a time that other users cannot access server resources. If your application falls into this category, you should consider letting the user end the session when finished. You can do this by simply providing a **Log Out** button. When the button is clicked, the application calls the **Session.Abandon** method, which immediately triggers the **Session_OnEnd** event.

This session-time-out characteristic of Web applications is equally troublesome to applications that rely on resources requiring user authentication. If the user ignores a running application for too long, the application will end the session and log off any connections it has established. If the user makes another request, the application may not function as expected.

You can avoid this time-out problem with a little planning. One popular method of detecting session time-outs is by storing the **Session** object's **SessionID** property as a *Session* variable. Then, each time the user tries to navigate to a page requiring a valid connection, you check the current **SessionID** against the ID stored in the **Session** object. If they do not match (or if the *Session* variable is empty), you have detected a session time-out, and you can take appropriate action.

Importing Type Library Constants with Global.asa

A COM component typically includes a list of named constants as part of its type library, along with other information about the component that enables it to be automated by other applications.

If your Web application uses a COM component that has declared enumerated data types in its type library, you can import them into your application space in Global.asa. Doing so makes it possible to refer to the data types declared in the type library by name from any script within the application boundary. The syntax for importing a type library is as follows:

```
<!--METADATA TYPE="TypeLib"
NAME="typelibraryname"
FILE="file"
UUID="typelibraryuuid"
VERSION="majorversionnumber.minorversionnumber"
-->
```

You're required to specify either the *File* or the *UUID* parameter, but your application will be more portable if you specify both. The *Name* parameter may come in handy if you have to disambiguate, or need to establish a single interpretation for enumerated constants that have the same name but are from different type libraries. The following example imports the constants declared in the ADO type library:

```
<!--METADATA TYPE="TypeLib" NAME="ADO"
FILE="C:\Program Files\Common Files\System\ado\msado15.dll"
UUID="00000200-0000-0010-8000-00AA006D2EA4" VERSION="2.0"
-->
```

Declaring Objects in Global.asa

You can declare objects in Global.asa with session or application scope by using the extended <OBJECT> tag. This tag is placed outside of any <SCRIPT> tags. In addition to setting *ID* and *ClassID* parameters, you must set SCOPE (either "Session" or "Application") and RUNAT=SERVER. This example creates an instance of the Page Counter component:

```
<OBJECT ID=GlobalPageCounter SCOPE=Application RUNAT=Server
   CLASSID="clsid:4B0BAE86-567A-11D0-9607-444553540000">
</OBJECT>
```

Having defined the **GlobalPageCounter** object in Global.asa, you can use it from any ASP file in your application without first retrieving it from the **Application** or **Session** object, or specifically calling **Server.CreateObject** first.

```
There have been <%= GlobalPageCounter.Hits("default") %> hits.
```

> **Note** The objects declared in Global.asa with <OBJECT> tags are not fully instantiated until the server processes a script that references that object. This saves resources by creating objects only as necessary.

Selecting Object Scope

Components can exist and operate within the scope of an application or session, or they can be created and destroyed on a page-by-page basis.

Objects stored in the **Application** object are available to all users of the application. The **Application** object is a suitable place to store objects that have no user affinity, such as a page counter. Normally, values and objects with application scope are created when the application begins, and are accessed in a read-only fashion by users of the application. When you make changes to **Application** object values, you should use the **Lock** and **Unlock** methods to prevent users from accessing data while it is changing.

In general, very few objects should be given application scope. They should support the "Both" threading model, as "Apartment" threaded objects force the **Application** object into a single thread of execution, and free-threaded components tend to be slower overall. COM components should not be given application scope, because they would be created when the application starts up, even though they might not yet be needed.

Because the **Session** object is designed to store information about the current user's session, components stored in this object can exist as long as the user's session is active. Since each value and object stored in the **Session** object increases the server's memory requirements for each user of the application, you should store values for only as long as they are necessary, and free them when they are no longer needed.

Page-scope objects are created each time the page is requested. They can use any threading model, but only the "Apartment" and "Both" models are recommended for scalability purposes.

Process Isolation and Crash Recovery

Besides adding the power of transactions to ordinary Web pages, the combination of IIS 5.0 and Component Services produces another powerful result: *process isolation.*

Applications and Processes

Extending a Web server has always involved tradeoffs between performance and safety. In early versions of IIS, all ISAPI applications (including ASP) shared the resources and memory of the server process. Although this design increased performance, unstable components could cause the server to crash—not an acceptable behavior for mission-critical applications like IIS. To make matters worse, in-process components couldn't be unloaded unless the server was restarted—which meant that reloading existing components required all sites that shared the same server to restart, whether they were directly affected by the upgrade or not.

Running isolated processes in IIS 5.0 combines the performance of the ISAPI/ASP model with the safety of the CGI model. Segregating applications into their own process space protects the entire Web server from unexpected application failures. This is the main idea behind process isolation.

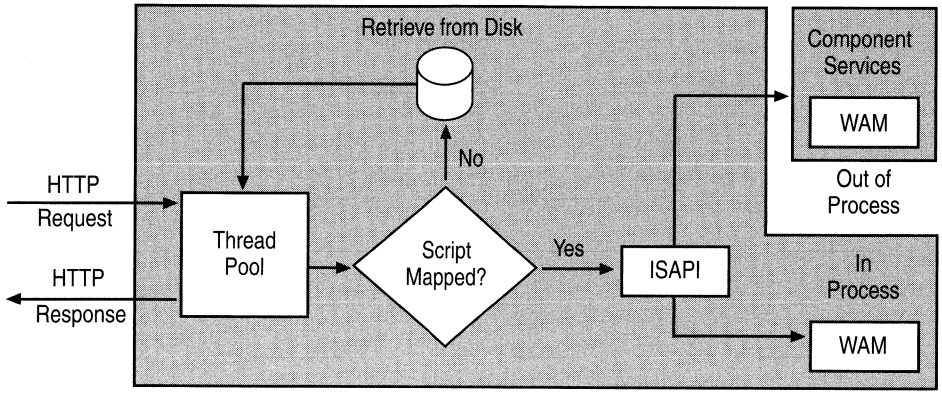


Figure 6.6 Process Flow for IIS 5.0 Applications Running In and Out of Process

Out-of-process applications run within their own processes, separate from the rest of IIS 5.0. When an ASP file is requested by the browser, the Web server first assigns a thread from the pool to handle the request. Then, IIS 5.0 determines how to handle the requested file according to its internal Multipurpose Internet Mail Extensions (MIME) and script mappings. For an ASP file, the ISAPI filter for ASP is invoked.

Next, the Web Application Manager (WAM) determines whether the file is part of an application marked to run in a separate memory space. If so, the request will be handled by an external Component Services process. Component Services, which is a run-time environment for COM components, hosts a WAM proxy object for each isolated process. Otherwise, the request is completed in process with IIS 5.0. WAM hosts all Web applications, whether they're in-process applications or not, and also hosts controls that load and interface with ISAPI DLLs. WAM itself is a COM component, and can be run in the IIS 5.0 process or isolated in a separate process. There are three main reasons for running an application as an isolated process: component development, fault isolation, and Web site safety.

Component Development

Rather than taking down the entire server to update a single component, process isolation makes it possible to stop and restart just a single application. You can use Internet Services Manager to add an updated component to an application.

To add an updated component

1. Right-click the application root directory, then click **Properties**.
2. Select the **Virtual Directory** tab; click **Unload**.

Once the old component has been replaced, IIS 5.0 will restart the application when it receives the first request.

Fault Isolation

Process isolation limits the effects of a crash to the single application that caused it. In addition to protecting your server from the crash, the application can be configured to restart automatically. In the case of a fatal error, the application's process is automatically terminated. Because the application is running in the Component Services system process, all transactions in progress are aborted. The Windows® Event Log stores a record of the event, and Component Services restarts the application. The only ones affected by the failure are clients with outstanding requests to that specific application.

Web Site Safety

The process isolation that Component Services provides makes it possible to host untested or unstable applications without risking the stability of the entire server. Now the Web server can gracefully tolerate failures with minimal disruption to clients of other applications. By separating unstable applications into their own memory space, you prevent a single application error from bringing down the rest of your site.

Configuring an Isolated Process

By default, an IIS 5.0 application runs in the IIS 5.0 process. If you decide to run your application as a separate process, you can set this up with Internet Services Manager.

To run an application as a separate process

1. Right-click the application root directory, then click **Properties**.
2. Select the **Virtual Directory** tab. (You must first create an application for your virtual root, if you haven't already done so. For more information, see "ASP Best Practices" in this book.)
3. Select the **Run in separate memory space** check box, and click **OK**.

Internet Services Manager automatically creates a Component Services package for your application. For example, if you were to configure the IIS 5.0 Help application to run in an isolated process, a new Component Services package would be created, named `IIS-{Default Web Site//Root/IISHELP/}`. Transactional components to be run within the context of this new process can be installed into this Component Services package by using Internet Services Manager.

Out-of-Process Components

Out-of-process components are COM components implemented as executables; these start as a separate process on the same computer as the client application. Out-of-process components, also called "local servers," are different from out-of-process applications (known, once again, as isolated processes). When you start an out-of-process component on the Web server, IIS 5.0 becomes the client application.

COM objects are implemented inside a server. Most of the time, the COM server is implemented as a DLL that executes in the same process as the client application. Sometimes the COM server is implemented as an executable that runs in a separate process from the client. For instance, when you create an instance of an Active Document server such as Microsoft® Word, you actually start a copy of the server application. When you instantiate such an object from ASP, you create a new process on the Web server. Because the Active Document server is an executable rather than a DLL, it cannot be loaded into the IIS 5.0 process.

When you use **Server.CreateObject** in an ASP page to start an out-of-process component, and IIS 5.0 has not been configured to allow out-of-process components, it will return the following error:

```
Server object error 'ASP 0196'
Cannot launch out of process component
/myvroot/launch_exe.asp, line 16
```

This is the result of an ASP safety mechanism that prevents executables (but not DLLs) from being started directly from ASP.

There are several reasons for this safeguard. Not all executables are safe to use on the server, and some may pose security risks. Also, because in-process component DLLs are faster, more secure, and can be hosted by Component Services, they are much better suited for server-side use.

Furthermore, out-of-process components often create individual server processes for each object instance, reducing their performance to that of CGI applications. They simply do not scale as well as component DLLs that run in process with IIS 5.0 or Component Services. If performance and scalability are priorities for your site, using out-of-process components is strongly discouraged. On the other hand, intranet sites that receive moderate to low traffic might be able to use an out-of-process component without adversely affecting the site's overall performance.

Using the Metabase

The IIS 5.0 metabase stores configuration settings for IIS 5.0. It performs some of the same functions as the system registry, but uses Microsoft® Active Directory Service Interfaces™ (ADSI) to administer this high-use storage facility.

If you want to enable the use of out-of-process components, you must set the IIS 5.0 metabase property **AspAllowOutOfProcComponents** to TRUE. This setting is accessible from either the **IIsWebService** or **IIsWebVirtualDir** Admin objects.

If you set the **AspAllowOutOfProcComponents** property to TRUE on the **IIsWebService** object, all in-process applications will be able to start executables from script. An in-process application is a virtual directory that has been marked as an application starting point, but which does not have the **Run in separate memory space** option selected in Internet Services Manager.

If you set the **AspAllowOutOfProcComponents** property to TRUE on the **IIsWebVirtualDir** object, and it contains an application that has been marked to **Run in separate memory space** as an isolated process, only the affected application may start executables from script. If the application is set to run in process, the setting will have no effect.

You must have adequate permissions to modify the IIS 5.0 metabase. If you try to modify it without proper permissions, you might encounter an error message.

The following ASP code demonstrates the steps required to set the *AspAllowOutOfProcComponents* parameter on the **IIsWebService** Admin object. You will need to restart the Web server service (by stopping and starting the IIS Admin Service in the Service Control Manager) after making this change.

```
<%
   'Get the IIsWebService Admin object.
   Set oWebService = GetObject("IIS://LocalHost/W3svc")

   'Enable AspAllowOutOfProcComponents.
   oWebService.Put "AspAllowOutOfProcComponents", True

   'Save the changed value to the metabase.
   oWebService.SetInfo
%>
```

Security Considerations

Out-of-process applications and components, including ISAPI extensions, are not able to access IIS 5.0 metabase properties using the built-in administration objects of IIS 5.0. This restriction is designed to prevent changes to the metabase from unauthorized sources. If you want to allow out-of-process applications to access the metabase, change the identity of the out-of-process Component Services package from the interactive user to a specific user account, and give that account access to the metabase. This is also somewhat risky, but the risk is limited to a single application package.

A Note About Application Testing

Performance testing, especially using multiuser scenarios, is a critical part of Web application design and development, and needs to be considered as a part of the overall planning of the application. Performance testing is much more critical for server applications than for desktop applications, because managing simultaneous users places higher demands on the application.

Remember that using the **Session** object to store values increases the memory requirements of your application, and therefore decreases the number of concurrent users an application can support. When you build your application, the following three factors determine how much server memory your application will consume:

- **Concurrent Sessions** The number of sessions that exist at any given moment is cumulative over the lifetime of the **Session** object. So, if you have a **Session.Timeout** value of 20 minutes, your concurrent sessions will be equal to the number of connections you expect to service over a 20-minute period.
- **Variables and Objects Per Session** How many objects or variables are you storing? A few session settings are fine. Long lists of session-scope variables (especially if they are components) should be avoided. As the number of variables increases, the time it takes to retrieve them also increases.
- **Size of Each Variable or Object Stored** Are you storing lengthy strings or large component objects? Store them for as long as necessary, and then free them—replace long strings with empty string and objects with **Nothing** (or **Null**).

Design Patterns for Web Applications

Web application developers may find that the model used by applications on the Web conflicts with the conceptual model they have developed for other platforms. For instance, unless you use client-side ActiveX controls or Java applets, your user interface is limited to an HTML description of the form inputs or a list of links to other areas of your site. Despite the availability of DHTML and ASP, most Web-based applications will never consist of more than an interconnected series of dynamically generated static states. Click a button and something happens. Click a link and a new page appears.

This final section of the chapter explores effective Web-application design, and covers a variety of techniques that you can use to enhance your applications.

Factoring Your Application

A Web application is a hierarchy of interdependent pages, each one representing a distinct stage of the application. Web applications map an ordered sequence of input events to a corresponding sequence of output events, using a finite number of elements.

Most Web applications make use of (or should make use of) the following: content, hyperlinks, forms, components, server-side actions, redirection, and loops.

- **Content** Content—information presented in the form of text, graphics, or even music or video files—is the most common element in most Web applications. Content can be presented on static HTML pages or be dynamically generated by an ASP page. Content elements usually have one well-defined entry point with links to other pages.
- **Hyperlinks** The primary purpose of hyperlinks is to facilitate movement to other pages or parts of the same page. Site maps, toolbars, HREFs, anchors, form buttons, and navigational controls are all examples of hyperlinks. These elements often appear in their own browser frame and control the navigation throughout the site. They can also appear as a separate index page or table of contents, which is replaced once a link is selected. Hyperlinks represent a user choice, very much like a menu option does in a stand-alone application.
- **Forms** Forms are used to collect information from the user. Depending on how a form is designed, its outcome might change once it is submitted, as a function of user input. Forms can be chained together consecutively to create "Form wizards." Client-side or server-side actions usually process these forms.
- **Components** A component is any code-level unit providing a relatively independent piece of logic that can be used either separately or in combination with other components throughout the application. The level of granularity and scope of responsibility distinguishes components from other application elements. Components can generate content or provide server-side logic.

- **Action** Action in this case means how an application responds to user action. When the user submits a form, for example, the application processes the form. Action pages perform business logic, such as data entry, calculations, or administrative functions. Action (processing) can occur both on the server and on the client, and can be used to generate content such as confirmation or error messages.
- **Redirection** A redirection page or script can perform logic to branch the flow of the application. Although it doesn't actually perform a redirect, a page containing a frameset can also be considered a redirection page, since it causes other pages to load. Redirection is usually more adaptable and flexible if implemented as a separate ASP page.
- **Loops** Loop pages are ASP pages that refer to themselves in order to present different content, depending on user input. For example, a page might select content based on which browser is being used.

You can create a huge variety of complex applications using the elements just described in this list.

Using Forms for Input

The standard method of interacting with Web pages is the HTML form. Forms can contain any number of inputs, including text entry, command buttons, "radio" selection controls, and check boxes. Forms can be as simple as a single button, or they can contain a complex layout of client-side controls. A Web page might have several distinct form structures, each with its own processing logic to be performed when the form is submitted.

Suppose you want your users to log on to your Web application by providing a user name and password. To accomplish this task, you create a simple HTML page with two text fields and a **Submit** button in an HTML form.

The HTML for the form might look like the following example:

```
<FORM ACTION="./Logon.asp" METHOD="GET">
  Your name:     <INPUT TYPE="TEXT"     NAME="User">
  Your password: <INPUT TYPE="PASSWORD" NAME="Pwd">
  <INPUT TYPE="SUBMIT" VALUE="Log On">
</FORM>
```

When this form is submitted, the values that the user enters are collected and sent to the server as a request. These values are passed as name/value pairs to the page referenced in the ACTION attribute of the <FORM> tag. They are appended to the requested URL after a question mark (?) and are separated by ampersands (&). If the user had entered "John Doe" as the user name, and "Amnesia" as the password, the following URL would be requested when the **Submit** button was clicked.

```
http://myServer/test/Logon.asp?User=John+Doe&Pwd=Amnesia
```

Any information appended to the URL like this is said to be *URL encoded*. URL encoding replaces reserved characters, like spaces and ampersands, with URL-neutral characters. The space in "John Doe" is replaced by a plus character (+). Pluses, equal signs, commas, percent symbols, and question marks also need to be encoded. These and other special characters can be represented in the format *%hh*, where *hh* is the hexadecimal value of the ASCII code for that character.

ASP provides a server-side method, **Server.UrlEncode**, to perform encoding (and decoding) for your parameterized URLs. Whenever you create hyperlinks that contain name/value pairs, you should always encode them to avoid invalid URL syntax. Unfortunately, you cannot use this ASP method on the client, since it is a method of a server-side object. You could write a client-side script function to do this, but it's easier to let the form processing logic of the browser do it for you.

The Difference between GET and POST

When the user enters information in a form and clicks **Submit**, there are two ways the information can be sent from the browser to the server: in the URL, or within the body of the HTTP request.

The GET method, which was used in the example earlier, appends name/value pairs to the URL. Unfortunately, the length of a URL is limited, so this method only works if there are only a few parameters. The URL could be truncated if the form uses a large number of parameters, or if the parameters contain large amounts of data. Also, parameters passed on the URL are visible in the address field of the browser—not the best place for a password to be displayed.

The alternative to the GET method is the POST method. This method packages the name/value pairs inside the body of the HTTP request, which makes for a cleaner URL and imposes no size limitations on the form's output. It is also more secure.

ASP makes it simple to retrieve name/value pairs. If the form's output is passed after the question mark (?) on the URL, as occurs when using the GET request method, the parameters can be retrieved using the **Request.QueryString** collection. Likewise, if the form is sent using the POST method, the form's output is parsed into the **Request.Form** collection. These collections let you address the form and URL parameters by name. For example, the value of the form variable *User* can be passed into a VBScript variable with one line of script:

```
<% UserName = Request.Form("User") %>
```

You don't need to specify the collection (**Form** or **QueryString**) in which you expect to find the *User* parameter. The following is an equally valid method of searching for the *User* parameter:

```
<% UserName = Request("User") %>
```

In the absence of a specific collection, the **Request** object will search *all* of its collections for a matching parameter. This is meant to be a programming convenience. However, the ASP **Request** object also contains collections for **ServerVariables** and **ClientCertificates**, which contain sensitive server and user authentication information. To avoid the possibility of "spoofed" values, which are values entered by the user in the URL, it is highly recommended that you explicitly use the collection name when searching for parameters from these collections.

The following script combines a form and an action (the script that processes the form) into a single page. By posting the form data back to the same ASP page that displays the form, server-side script can process the output of the form. This is perfectly valid, and for simple script is often more convenient than posting to a second ASP page.

```
<%@ LANGUAGE="VBScript" %>
<!-- FILE: logon.asp -->
<HTML>
  <HEAD>
    <TITLE>Authentication Form</TITLE>
  </HEAD>

  <BODY BGCOLOR=#FFFFFF>
    <% If Request.Form("User") = "" Then %>
       <P>Please enter your Name:
       <FORM ACTION="./logon.asp" METHOD="POST">
       Your name:     <INPUT TYPE="TEXT"     NAME="User">
       Your password: <INPUT TYPE="PASSWORD" NAME="Pwd">
       <INPUT TYPE="SUBMIT" VALUE="Log On">
       </FORM>
    <% Else   'User verification and logon code goes here %>
       Welcome <%= Request.Form("User") %>!
    <% End If %>

  </BODY>
</HTML>
```

Note If you use a separate ASP file to handle the processing of a form, the **Request.Form** collection will be emptied when you redirect to the new page. In order to retain the form values, you must copy them to the **Session** object from which they can be accessed on subsequent pages.

Although the sample authentication form shown here works, there's a good reason why you would not want to use it in practice. Logon information is sensitive and should be subject to rigorous protection from prying eyes. Although you can use the POST method to contain the user's password within the body of the HTTP response, it is still possible to intercept and read it.

For mission-critical applications, IIS 5.0 provides both secure authentication with integrated Windows authentication and Client Certificates, as well as data encryption with Secure Sockets Layer (SSL). For more information about authentication and encryption, see "Security" in this book.

Client-Side Form Validation

Forms require some sort of input. If the user hasn't entered any information or has entered a bad combination of information, it makes little sense to send the form back to the server. Forms that are submitted without preliminary data validation increase the server's load (and the user's frustration) unnecessarily. Validate information as much as possible when the form is submitted.

In fact, don't stop with the client tier. Just in case the client doesn't support client-side scripting, validate again as data is passed to the middle tier. Most importantly, use the data integrity and validation rules of your database to protect yourself against inadvertent mistakes in your own business logic. This defensive approach to data validation can save you frustration in the long run; you'll know that data is protected at all levels of your application.

To validate form input with client-side script, you need to declare an event-handler for the submit event of the form. If your form uses a SUBMIT input type, create an event handler using the name of the form followed by an underscore and the command: "**_OnSubmit**." To submit the form, return True from the validation routine. Return **False** to abort the submission and to return to the form.

If you prefer to use the BUTTON input type on your form, you'll need to define an event handler using the name of the button followed by an underscore and the command: "**_OnClick**." Instead of returning True to submit the form, however, you will need to call the **Submit** method of the **Form** object explicitly. This method may not be available on some browsers.

The following client-side script verifies that the form's user name and e-mail address fields (both required by the form) have been filled in. It then submits the form by returning True from the **OnSubmit** event-handler:

```
<SCRIPT LANGUAGE="VBScript">
<!--
Function FeedbackForm_OnSubmit()
   'Disallow submit until the form fields have been validated.
   FeedbackForm_OnSubmit = False

   'Get a reference to the form.
   Set theForm = Document.FeedbackForm

   'First, check that UserName has been filled in.
   If Trim(theForm.UserName.Value) = "" Then
      MsgBox "Enter your name.", vbCritical, "Need Input"
      theForm.UserName.Focus
   Else
      'Next, check for the e-mail name.
      If Trim(theForm.UserEmail.Value) = "" Then
         MsgBox "Enter your e-mail address.", vbCritical, "Need Input"
         theForm.UserEmail.Focus
      Else
         'Continue with submission.
         FeedbackForm_OnSubmit = True
      End If
   End If
End Function
--></SCRIPT>

<form name=FeedbackForm action=TeeFeedback.asp method=POST>

<B>Tell us how to get in touch with you:</B>
<PRE>
      Name     <INPUT type=TEXT name="UserName"> (Required)
      E-mail   <INPUT type=TEXT name="UserEmail"> (Required)
      Tel      <INPUT type=TEXT name="UserTel">
</PRE>

<INPUT type=SUBMIT value="Submit Comments"> 
<INPUT type=RESET  value="Clear Form">
</FORM>
```

Hidden Form Fields

When you chain forms together to create a "Form wizard," the information entered on each form needs to be stored until the last form has been filled in. There are three ways to pass values between ASP files:

- Accumulate information in the **Session** object.
- Append information to the end of the URL and use **QueryString** to pass it.
- Use hidden HTML form variables.

Sometimes the requirements of your application don't permit you to store form data in the **Session** object, even temporarily. This might be the case for a large-scale site with thousands of concurrent users, where memory is at a premium.

Passing values on the URL works for small amounts of information, but will be insufficient when the quantity of data is large.

So, although the amount of information passed between the client tier and the middle tier increases, hidden form fields make it possible to include previously entered or application-specific information as part of the current form submission. A hidden form field isn't displayed to the user, but is sent as a name/value pair when the form is submitted.

> **Note** You should avoid using hidden form fields to send back information that you are using for security or authentication purposes. Since they are available as text in the form body, these values can easily be "spoofed" by anyone who can view the HTML source. Even an unsophisticated intruder could develop a small routine to try many possible values, in an attempt to crash (or otherwise break) whatever server code is using the hidden value.

Redirection

Sometimes, the page being requested by the browser isn't the one you'd like to send. For example, suppose a user requests Oldpage.htm, which has been replaced by NewPage.asp. Standard HTML syntax provides a means whereby you can redirect (or divert) a request to another location. The syntax looks like this:

```
<HEAD><META HTTP-EQUIV="REFRESH" CONTENT="0;URL=NewPage.asp"></HEAD>
```

You can also use ASP to redirect a request to another page based on the logic of your application. The syntax is simple:

```
Response.Redirect "./NewPage.asp"
```

The **Redirect** method of the **Response** object operates by sending the "302 Object Moved" response header, plus the new location of the file, to the client. When it receives this response, the user's browser automatically requests the new page.

Because redirection depends on HTTP headers, which come at the beginning of the document, you can't redirect once text has been sent to the client. If you redirect in a server-side script after data has been sent to the client, the following error occurs:

```
Header Error
The HTTP headers are already written to the client browser. Any HTTP header
modifications must be made before writing page content.
```

If you don't know at the beginning of the page whether you need to redirect, you can use the buffering capabilities of the **Response** object. If **Response.Buffer** is True, HTML output is collected in a buffer and sent all at once to the client. If at some point you need to redirect to another page, ASP automatically discards any existing output in the buffer when you call **Response.Redirect**. You may also use **Response.Clear** at any time to clear the buffer and start again.

The following example demonstrates this concept:

```
<% 'Begin buffering the HTML.
   Response.Buffer = True %>
<HTML>
<BODY>
HTML text before potential redirect.

<%
   On Error Resume Next

   'Script generates an error here.

   If Err.Number > 0 Then
      Response.Clear
      Response.Redirect "./error.asp"
   End If
%>
```

Once you call **Response.Redirect**, the script ends and the redirection headers are sent immediately. Any code following the redirect will not be executed, although it is required for syntactical correctness.

You can use **Response.End** to end a response from the server immediately and to send the current output to the browser. Since the response ends at that point, any HTML that follows is not sent. The following script detects that the user has connected anonymously (the *LOGON_USER* server variable is empty) and forces a logon by returning a "401 Access Denied" message.

```
<%
   strLogon = Request.ServerVariables("LOGON_USER")
   If IsEmpty(strLogon) Or strLogon = "" Then
      'Up to this point, no HTML has actually been sent.
      Response.Status = "401 Access Denied"
      Response.End
   End If
%>
<HTML>
You are logged on as: <%= strLogon %>
</HTML>
```

A well-behaved browser will try this page a second time, with logon credentials. If you don't use **Response.End**, the script in the ASP page will execute twice, possibly leading to unexpected side effects (such as adding duplicate records to a database).

Client-Side Redirection

Redirection can also occur in client-side script. A client-side redirect shifts focus from the current page to another. It can occur as the page is loaded, or can be deferred until the user performs an action, such as clicking a button.

The browser's object model includes a **Location** object, which represents the URL of the content currently displayed in the browser window. The following example uses this object to reload a frame-dependent page inside its parent frame. The script (which appears at the top of each frame page) detects that the page has been loaded as the top frame, and changes the browser window location to the parent frame. (Note that this script can be placed in an include file, to avoid duplicating it on all your pages.)

```
<SCRIPT LANGUAGE=JavaScript>
<!--
if(top == self) {
   var currURL = unescape(window.location.pathname);
   var newURL  = "parentFrame.asp?" + currURL;
   var appVer  = navigator.appVersion;
   var NScp = (navigator.appName == 'Netscape') &&
              ((appVer.indexOf('3') != -1) ||
               (appVer.indexOf('4') != -1));
   var MSIE = (appVer.indexOf('MSIE 4') != -1);
   if (NScp || MSIE)
      location.replace(newURL);
   else
      location.href = newURL;
}
//-->
</SCRIPT>
```

This method requires that the parent frame accept the child frame location as a *URL* parameter, and that it deal with this parameter appropriately. The following script manages the last part of this redirection:

```
<%@ LANGUAGE=VBScript EnableSessionState=False %>
<%
   frmSrc = Request.QueryString  'Get entire URL parameter.
   If frmSrc = "" Then frmSrc = "homepage.htm"
%>

<FRAMESET rows="60,*" frameborder=0 framespacing=0>
    <FRAME name="nav_fr"  src="navbar.htm"  scrolling=auto noresize>
    <FRAME name="page_fr" src="<%=frmSrc%>" scrolling=auto>
</FRAMESET>
```

Redirecting During Session_OnStart

Redirection during the **Session_OnStart** event is possible, and sometimes required by your application (see the sidebar "Web Clusters and ASP Session State" earlier in this chapter). Here is an example:

```
Sub Session_OnStart
   Response.Redirect "MyStartPage.asp"
End Sub
```

Not all of the **Response** object's methods are available during the **Session_OnStart** event. Most notably, you cannot use the **Write** method to display messages, since the client would never see them. However, you can redirect to an error message, if your application cannot successfully initialize a new user session.

Debugging Applications and Components

Debugging can be the most frustrating part of the Web application developer's job. Unlike desktop applications, which are self-contained, Web applications can be spread over several systems and frequently combine different programming languages and technologies. Although some application errors can be avoided with careful planning, others are unexpected side effects of complex component interactions. Tracking down these problems often requires the use of a variety of debugging and development tools.

This section contains debugging tips for the ASP environment. It also contains information about setting up a debugging environment for ISAPI extensions and server-side components.

For information about common error messages and error logging, see the IIS 5.0 online product documentation.

Script Debugging in Active Server Pages

Because ASP pages often contain a mixture of HTML, server-side script, and components, problems can be difficult to track down. There are basically three ways to deal with errors: avoidance, debugging, and script management.

Avoiding Common Mistakes

Studies show that developers tend to make the same types of mistakes no matter what programming language they are using. Rather than spending a lot of time tracking down bugs, it is always easier to avoid creating them in the first place. Here are some common scripting mistakes that lead to bugs, as well as ways to avoid them.

Misspelling Variables

Most scripting run-time errors can be attributed to misspelled variables. VBScript automatically declares unrecognized variables, which can lead to subtle errors in your code. To avoid this problem, use the **Option Explicit** statement as the first line of script on every page. The **Option Explicit** statement requires variables to be explicitly declared with a **Dim** statement.

JScript has no counterpart to the **Option Explicit** statement. Unrecognized variables are automatically declared when they are defined. Some ASP methods, such as **Response.Write**, produce script errors when used with an undeclared variable. If a variable is declared, but not defined, it will return as either "Undefined" or "NaN." (For more information, see the JScript documentation.)

Using an Object or Variable Out of Scope

A common problem is not understanding which objects are available in a given context. For example, you might attempt to set properties for a **Session** object in a client script, or to use the Internet Explorer object model in a script running on a different browser.

Make sure that the objects you reference are available in the current scope. Be aware that objects (such as ASP built-in objects) are not an inherent part of a language such as VBScript; instead, they are part of the environment in which a script is running.

Not Using the Web Server to View ASP Pages

To view ASP pages in the browser, you need to request them from the Web server by using HTTP syntax. If you try to view them directly from the file system, the browser will either try to download the file, or will display the script without executing it. Here are some other reasons for ASP pages not displaying properly: not having script (or execute) permissions set for the directory in which the page is stored, and not using the .asp extension when naming files.

Using the Scripting Language Inefficiently

Sometimes when developers are unfamiliar with a scripting language, they write code that fails to take advantage of the language's capabilities. Consequently, their code runs more slowly than correctly implemented script. Likewise, if they don't understand the language conventions, they might inadvertently make syntax errors, such as using the wrong type of quotation marks to enclose string literals. This is an easy mistake to make when switching between languages such as Visual Basic and SQL.

Always seek to familiarize yourself with the native functionality, operators, and conventions of the scripting language you are using.

Mixing Data Types

Because all ASP scripting variables are variants (meaning they can hold values of any type), you can easily assign inappropriate values to variables. For example, you could assign an object to a variable you intend to use for strings.

To help you remember what type of data a variable is supposed to contain, use a naming convention that will help you remember variable types. For a table of recommended variable names for VBScript, see "ASP Best Practices" in this book.

Misusing the Equal Sign

Visual Basic users expect the single "=" (equal sign) to evaluate the equality of the two operands. In JScript, however, the single "=" will assign the right-hand value to the left-hand operand. (JScript uses a double "=" to express equality.) Mistakenly using a single "=" in an expression will not only overwrite the left-hand value, it will cause the expression to evaluate to TRUE. If this happens in an **If** statement, the inner logic is executed; in a **While** statement, it creates an infinite loop.

Using Procedures Incorrectly

Not understanding when to use a function or procedure, or calling the incorrect function, is a common problem in any language. You can avoid using the incorrect arguments for functions, or passing arguments in the wrong order, by double-checking the function definition. Additionally, make sure that the function you are calling performs the task you want it to, check syntax for functions whenever using them, and avoid relying on default argument values.

Not Handling Errors

You should always check for unexpected user input, such as a string when prompted for a number, or a number value outside the bounds of what the program can use. Likewise, you must anticipate errors from within the program, and understand the meaning of values returned by the functions you use.

To handle internal program errors in VBScript, you must use the **On Error Resume Next** statement. Without this instruction, the script will abort as soon as the error is detected, which prevents you from quietly handling errors with script. Here are a few precautions to keep in mind when using the **On Error Resume Next** statement:

- When debugging, it is often necessary to disable error handling, since it will not be apparent where errors are occurring. Re-enable it when you are finished debugging.
- Because you are resuming program execution at the next statement whenever an error occurs, you should avoid using script that may produce an error as the logic portion of an **If** statement, or the test of a **While** loop. If these statements cause errors, the inner logic of the **If** statement will not be executed, and the **While** loop will never complete.

Making Off-by-One Errors with Collections

Not all collections in VBScript begin with element 1. Some collections—the ADO **Fields** collection, for example—start at element 0. When looping through a complete collection, it is often safest to use the VBScript methods **LBound** and **UBound** to specify the starting and ending array indexes, respectively. Note that JScript also defines these methods on its **VBArray** object.

Forgetting Ending Braces, Delimiters, and Statements

The longer the script becomes, the more likely that end braces, the ending statement of a code block, such as **End If**, or an ending code delimiter (such as **%>**) can be overlooked. The result is often confusing, and ambiguous script errors can occur. To avoid such problems, make a practice of typing the closing portion of a statement as soon as you type the opening portion.

Debugging ASP

Until the introduction of Microsoft® Script Debugger, debugging scripts in ASP pages often required lots of debugging output and an intimate knowledge of how the scripting language processed the code. Script Debugger makes this process much easier.

Microsoft Script Debugger

You can use Script Debugger to test scripts written in VBScript and JScript, as well as applications written in Java. You can also debug scripts in other languages that support host-independent debugging, such as REXX or PerlScript.

Using Script Debugger, you can:

- View the source code of the script you are debugging.
- Control execution line by line through the script.
- View and alter variable and property values.
- Set breakpoints and view the call stack.
- Switch between threads of execution.

You can use Script Debugger to debug both client-side and server-side scripts. Debugging must be enabled for each ASP application that you want to debug.

To enable debugging

1. In Internet Services Manager, right-click an application virtual directory and click **Properties**.
2. On the **Virtual Directory** tab, click **Configuration**.
3. On the **App Debugging** tab, select **Enable ASP server-side script debugging** and click **OK**.

To begin editing a document in Script Debugger, first open the **Running Documents** window (from the **View** menu) and double-click to open a document. Before you can open the script, it must be loaded locally in the browser, or in the ASP environment on the local Web server, if one is installed. Once the script is loaded, you can set breakpoints, and step through the application.

You can view and change the values of individual variables in the **Command** window when a script is running. To view a variable value, type a question mark (?) followed by the name of the variable, and press **Enter**. You can also evaluate simple numeric expressions using the "?" command form. To edit the value, type the name of the variable followed by an equal sign (=) and a new numeric or string value.

Since Script Debugger only works with scripts running on the local Web server, you cannot debug scripts running on remote Web servers. Because the script debugger interrupts script execution when it encounters an error, you won't want to enable script debugging on systems that are hosting applications critical to your business. For these systems, it is better to trace execution by using logs and output.

For more information about Script Debugger, see the IIS 5.0 online product documentation.

Debug Tracing

ASP debug tracing is limited to the following three methods:

- Liberal use of debug **Response** or **Write** statements throughout the code.
- **Response.AppendToLog** to write script values and events to the IIS 5.0 Server log.
- A custom logging component that writes to the Windows® Event Log in Windows 2000 Server, to a database, or to an application-specific text file.

Implement debug tracing wisely. When you have to insert debug messages into your script, you should emphasize events that might enable you to reconstruct the flow through the code leading up to an error. Capture function calls and values of parameters. Log important values only; if you write too many messages, you can slow down the application considerably.

Try to make it easy to disable and enable debug tracing. For small projects, it's okay to comment out the debug trace statements that you want to disable. For larger projects, however, you should encapsulate the debug trace code in a script or component function that can be enabled and disabled independently of the code.

Tracking Events in Global.asa

Debugging **Session** and **Application** events can be tricky, because you can't use the **Response.Write** method to display messages to the client browser. One useful technique is to use the **Scripting.FileSystemObject** to generate a simple log file with text messages indicating the success or failure of these events. The following script writes notifications for **Application** start and end events, as well as for every **Session** start and end event.

```
<!--METADATA TYPE="TypeLib"
NAME="Scripting" FILE="C:\Winnt\System32\scrrun.dll"
UUID="420B2830-E718-11CF-893D-00A0C9054228" VERSION="1.0"
-->

<OBJECT ID="AppFileSystemObject" SCOPE=APPLICATION RUNAT=SERVER
   PROGID="Scripting.FileSystemObject">
</OBJECT>

<SCRIPT LANGUAGE=VBScript RUNAT=SERVER>
Sub OutputDebugFile(ByRef strText)
   Dim ts
   Application.Lock
   Set ts = AppFileSystemObject.OpenTextFile(Application("DebugFile"),_
            ForAppending, True, TristateUseDefault)
   ts.WriteLine Now & ": " & strText
   ts.Close
   Application.Unlock
End Sub

Sub Application_OnStart
   'DebugFile must be defined before calling OutputDebugFile.
   Application("DebugFile") = "c:\webs\appevnts.log"
   OutputDebugFile("Application Started")
End Sub
```

```
Sub Application_OnEnd
   OutputDebugFile("Application Ended")
End Sub

Sub Session_OnStart
   OutputDebugFile("Session Started: " & Session.SessionID)
   Response.Redirect "./end.asp"   'End Session (this creates a loop!).
End Sub

Sub Session_OnEnd
   OutputDebugFile("Session Ended: " & Session.SessionID)
End Sub
</SCRIPT>
```

Script Management

This section discusses some techniques to help you manage longer scripts, and to minimize subtle errors in large projects.

Establishing a Library of Helper Routines

It's often a good idea to collect commonly used functions into one file and to include that file in all pages requiring those functions.

The include file can contain plaintext, declare functions and subroutines with <SCRIPT> tags, or can define variables and constants. To include the file, use a server-side include directive at the top of your file. Here is an example:

```
<!-- #include virtual="/MyRoot/include/funclib.inc" -->
```

Usually, include files are named with the .inc extension. Although this differentiates the file, it may pose a security threat if you have directory browsing enabled. The .inc extension is not normally script-mapped by IIS 5.0, and most browsers don't recognize it. Consequently, anyone who knows where to find the files can download and open them. You can prevent this from happening by either associating the .inc extension with Asp.dll in the script map (use the **App Mappings** tab of the **Application Configuration** dialog box), or by storing the include files in their own subdirectory and disallowing directory browsing as well as read access.

The entire text of the include file is incorporated into the .asp source file at the point where the include statement appears. Although there is technically no limit to the number of files you can include, each one adds to the size of the compiled .asp file—so it's better to include only files that you know you will need.

Since all include files are processed before the script executes, you cannot dynamically decide which file to include. For the same reason, you cannot use the include directive in Global.asa.

Using Dictionaries to Partition the Session Namespace

When used indiscriminately, the **Session** object is no more than a global data area. For large Web applications, the **Session** collection can become quite cluttered. This increases the likelihood that some part of the application might inadvertently make changes that have unexpected repercussions elsewhere. To avoid this situation, development teams must either use a naming convention that will decrease the chance of duplicate **Session** key names, or must use other methods of storing session values.

One such method uses the **Dictionary** object to further partition the global session namespace. Like the **Session** object, the **Dictionary** object can store any number of values and keywords in an associative array. Disparate sections of the application can create individual **Dictionary** objects as needed to contain local values, and can also store a single reference to their namespace in the **Session** object. Not only can groups of values be managed as a single entity, the **Dictionary** object makes it easy to free resources all at once when they are no longer required by the application. For more information about the **Dictionary** object, see the IIS 5.0 online product documentation.

Debugging ISAPI and Server Components

This section explains how to connect to running applications, as well as how to use the Script Debugger in order to step through the source code of your components, ISAPI filters, and extensions.

Disabling Debug Exception Handling

ASP experts should recognize the following error message:

```
Error 'ASP 0115'
Unexpected error
A trappable error occurred in an external object. The script cannot
continue running.
```

This error is caused by an access violation in a component process. The reason ASP is able to display a message is because the crash is detected by the built-in *exception handling* of the ASP process.

When you are debugging an application component, it is often easier to detect where faults occur when exception handling is disabled. You can disable exception handling by using the **Process Options** tab of the **Application Configuration** dialog box in Internet Services Manager. The **Process Options** tab appears either when editing global properties, which affect all in-process applications, or when editing the properties of an out-of-process application marked to run within its own memory space.

The extra overhead in system resources required to run an application in its own process is worth the added security of knowing a crash will not affect other applications. When you clear the **Enable debug exception catching** check box on the **Process Options** tab, ASP will not capture serious application errors. Instead, they will appear as faults in both the isolated application process and in-process components.

Debugging IIS 5.0

Debugging IIS 5.0 may be necessary should any of the following events occur:

- You receive ASP 0115 error messages from pages that call server-side components.
- IIS 5.0 stops serving ASP pages.
- The Web service stops and clients cannot connect to the server.
- Inetinfo.exe causes a Dr. Watson to appear.
- Inetinfo.exe stops running for no apparent reason.

There are two ways to establish an environment for debugging server components and ISAPI extensions:

- Attach a debugging tool to the IIS 5.0 process.
- Use the World Wide Web Publishing Service to start a debugging tool.

Attach a Debugging Tool to the IIS 5.0 Service

If you are using a debugging tool capable of attaching to a Windows 2000 Server process, you can use it to debug your component or extension.

To debug an ISAPI extension with Visual Studio

1. Start the iisadmin process. This can be done from the command line with the command **net start iisadmin**. You can also use the **Services** application (in **Administrative Tools**) in Control Panel to start the IIS Admin Service; this will start the iisadmin process.
2. Run Visual Studio and select the **Attach to Process** command from the **Start Debug** submenu of the **Build** menu.
3. Click the **Show System Process** check box.
4. Select the Inetinfo process from the list and click **OK**.
5. Start the w3svc service. This can be done from the command line with the command **net start w3svc**. You can also use the **Services** application in Control Panel to start the World Wide Web Publishing Service.

Use Publishing Service to Start a Debugging Tool

If the debugging tool you are using is not capable of attaching to a Windows 2000 Server process, you will need to establish an appropriate debugging environment.

To start a debugging tool

1. Right-click on **My Computer** and click **Manage**.
2. Open the **System Tools** menu and click **Services**.
3. Right-click **IIS Admin Service** and click the **Logon** tab.
4. Select the **Allow Service to Interact with Desktop** check box and click **OK**.
5. Repeat steps 2 and 3 for all processes that run under the IIS Admin process, for example, World Wide Web Publishing Service and FTP Publishing Service.
6. Use the Registry Editor to add a subkey named "Inetinfo.Exe" to the following key:
   HKEY_LOCAL_MACHINE
   \Software
   \Microsoft
   \WindowsNT
   \CurrentVersion
   \Image File Execution Options key
7. Add the following entry to this new key:
   Debugger = DebuggerExeName, where DebuggerExeName is the full path to the debugger you are using.

   Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft® Management Console (MMC) whenever possible.

When the World Wide Web Publishing Service is started, your debugger will run as well. You can now set appropriate breakpoints in your ISAPI extension.

You won't be able to set breakpoints in a component's source code until the component has been loaded into memory. First, you will need to start Internet Explorer and view the ASP page containing the object. As soon as the page is loaded, you should be able to set breakpoints in your component. Click **Refresh** to view the page again, and trigger the breakpoints you selected. If the component cannot be loaded even once (for instance, if the fault occurs in component startup code), you will need to load the component DLL prior to starting the debugging session.

Inability to Create Components

Sometimes the **Server.CreateObject** method fails and produces an "ASP 0177: Server.CreateObject Failed" error. This can happen even if the component works fine in Visual Basic on the same computer or while using ASP on other computers.

One likely cause for this behavior is that the authenticated user does not have permission to invoke the COM object. In the simplest scenario, the authenticated user doesn't have access to the component DLL or executable. In many cases, however, the component depends on other DLLs that the authenticated user does not have permission to access. Usually, a majority of users access a site anonymously, using the IUSR_*computername* account.

To verify that this is a permissions problem, try invoking the component from another tool such as Visual Basic. This approach verifies that the component is registered properly on the server. If the component cannot be created from Visual Basic, you are probably not dealing with a permissions problem. (The component might need to be registered.)

If you believe you are dealing with a permissions problem, check permissions on the component and on any dependent files, such as other DLLs. If you still are unable to track down the problem, you may need to systematically search permissions.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server, as well as features for developing Web applications.

Web Links

http://www.15seconds.com/
: A free resource for developers working with Microsoft Internet solutions. This site includes the *15 Seconds* newsletter, Stephen Genusa's Frequently Asked Questions, List Servers, and the Consultant Program. There are also book reviews, how-to articles, and job opportunities that deal with ASP and Microsoft Internet solutions.

http://www.chilisoft.com
: ChiliSoft's Chili!Soft ASP brings the power of ASP to servers other than IIS. Chili!Soft ASP can host ASP pages and components on a variety of Web servers without any changes to code. Includes support for Windows-based Netscape Web servers.

http://www.activeserverpages.com/genusa
The premier "unauthorized" support site for ASP. Provides an excellent collection of ASP resources.

http://www.microsoft.com/intranet/
This comprehensive Web site called TechNet has everything you need to plan and build an intranet site. Explore white papers, FAQs, and case studies, or download free intranet solutions written by top Microsoft® Solution Providers.

http://msdn.microsoft.com
Open the **Libraries** menu and point to **Web Workshop Home**. From there, point to **Server Technologies**. This is the Active Server Pages workshop area of Microsoft MSDN; a must-see resource.

Books

ActiveX Web Programming by Adam Blum, 1996, New York: John Wiley & Sons, Inc.
Although written while ASP was still in development, this book is an excellent introduction to CGI, ISAPI, ActiveX controls, and client-side scripting.

Working with Active Server Pages by Michael Corning, 1997, Indianapolis: Que Corporation.
Covers design, development, and implementation of ASP pages. Includes examples of database-driven customer scenarios using ASP and ADO.

Professional Active Server Pages 2.0 by Alex Federov, 1998, Chicago: Wrox Press Ltd.
A highly recommended and comprehensive tutorial of ASP and ADO. Includes practical techniques for creating *n*-tier Web applications.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 7

Data Access and Transactions

To be useful, data must be accurate and accessible, and it must conform to the needs of users. Users should be able to respond to, and act upon, the information presented to them. The tool that makes this flexibility possible is the database, an essential element in today's Web applications.

The previous chapter introduced the *n*-tier application model, and described how to use a combination of client-side scripts and components with Active Server Pages (ASP) to create dynamic Web-based applications. This chapter builds on those concepts, beginning with a conceptual overview of Web database technologies, followed by technical discussions of how to harness the power of a data-driven approach for content publishing and information management on the Web.

In This Chapter

Web Database Technologies

Client-Side Data Access

Accessing Data with ASP and COM Components

Transaction Processing on the Web

Additional Resources

Web Database Technologies

The Internet is changing our expectations about the availability of information. What we expect from the Internet and from the Web is changing as well. Once you've visited a site that lets you browse a product catalogue and initiate a sales transaction online, nothing is more frustrating than visiting another site that talks about products, but doesn't let you purchase them immediately.

As online commerce and electronic publishing become increasingly common, sites that provide a high level of interactivity will replace those that simply present information. Interactivity and complexity call for information to be stored in a way that makes it easy to manipulate and modify. This is the central role of the database in today's Web applications.

Why a Web Database?

What makes the Web such a good mechanism for accessing a database? When you use a data management solution as part of your site, you reap the following benefits:

- **Ease of Deployment** It's no secret that the World Wide Web is a cheap and practical alternative to traditional client/server application deployment, and that it provides immediate cross-platform support on the client side. Implementing a dynamic Web database solution can be done relatively quickly and doesn't require a large team of developers.
- **Database Standards** The components that enable Web database access are built on proven standards. Web pages can access data from a variety of sources, such as Microsoft® Access, Microsoft® SQL Server™, or any database compliant with Microsoft® OLE DB or Open Database Connectivity (ODBC) Operator.
- **Data Security** By using IIS 5.0 and SQL Server, you leverage the security model of Microsoft® Windows® 2000 Server. By using Microsoft® Component Services, you automatically gain the data protection and operational integrity provided by a distributed transaction coordinator. You have a lot invested in your content—protect it! (For more information about transactions, see "Transaction Processing on the Web" later in this chapter.)
- **Dynamic Content** Dynamically generating HTML from a database is easier than making numerous manual changes to individual pages. By automating the creation of HTML from content stored in a database, you save time and make site management easier. In the end, you can focus on updating your content, not your HTML.

Data Publishing Considerations

Of course, not everything great about Web database access is free. Before you start combining databases and HTML, there are some important issues you should consider:

- **Static vs. Dynamic** How much of your site really needs data access? How often does your content change? Dynamic solutions, especially if they involve accessing a database, are slower than plain, static HTML pages. If you display data that doesn't change frequently, you can improve performance (for your server and your client) by converting dynamic pages to HTML.
- **Server Load** Be sure you have sufficient server resources to handle the increased demands of database access. Consider memory, CPU speed, Internet connection speed, disk subsystems and other critical hardware factors. If you are expecting heavy database traffic, you may need to separate your Web server and database management system (DBMS) onto two computers (or more). Also, use existing database and performance management tools to help you measure and balance your server load.
- **Tool Support** The tools used to develop Web-based applications sometimes aren't being updated as fast as the technology is changing. Research and choose your tools carefully before you implement a large-scale database project.
- **Database Scalability and Reliability** Determine how much the database is likely to grow. On average, how often will users access it? What kinds of tasks will they perform? What is your Web site's overall growth estimate—in both content and readership?
- **Client Presentation** How will users access the data on your site? Will they be able to add to it, or modify it? Will the users have their own copies of the data, or will they only have access to the information while online? Using Microsoft® Data Access Components (MDAC), information can either be manipulated on the server as part of a server-side query, or be bundled as a package and transmitted to the client. Choosing how the information will be presented to the user is perhaps the hardest decision you'll have to make—often a hybrid approach is best.

Industrial Strength Information

Database-centric publishing is not just a convenience—it's a form of commerce applied to the commodity of information. You should consider the content of your company's Web site as you would your company's product. In addition to providing the latest information, a good Web administrator makes sure the information suits the needs of the customer.

Intranet and Extranets

Using the same networking technologies as the Internet, the corporate intranet publishes and disseminates information. The intranet is quickly becoming an effective way to capture and present information throughout the enterprise. With the same technology used on your Web site and intranet, combined with a tight security model, you can also extend the intranet to your trading partners, contractors, and suppliers via an extranet. The extranet has become a primary means of interaction between companies that contribute to each other's everyday business.

Extending the reach of information is a key feature of intranets and extranets. Users need to have access to information that is stored in more than one location, across multiple databases, or different file formats. Many technologies exist today to give you full access to information stored in mail systems, relational databases, file systems, and mainframes.

If information truly is your commodity, you should constantly be searching for ways to improve its content and distribution. Not only can you replace paper forms with online data entry, teams and working groups can improve their communication and productivity by sharing common resources, data, knowledge, and new ideas.

Database-Centric Publishing

Instead of authoring directly in HTML, many successful Web sites initially store their content in a database and combine it with HTML layout tags only when deploying the content for publication. Combining raw information with a layout template, or HTML boilerplate, imbues the content with the same look and feel as other pages on the site. The end result is a site that has a uniform appearance from page to page, even though the information has been created by dozens of people.

For instance, Microsoft® Sidewalk®, the company's family of city-centric entertainment guide Web sites, uses a custom-built application based on HTML forms and SQL Server to collect content from its contributors. Once the information is entered, it is managed separately from the visual representation of the site. Sidewalk.com employs a large number of freelance correspondents to keep the content fresh, and a smaller team of Web administrators to manage the technological aspects of the site.

Content Search and Personalization

Data in a database can be sorted, filtered, and queried for relevance. Why should it not be the same on your Web site or intranet? It is surprising how much raw data a site can contain. Not every piece of information is of interest to everyone. Users don't want to spend time searching through data, and will be less likely to visit sites with poor search tools.

To attract visitors, many sites provide a list of current headlines, or compilations of topics that appeal to people with common interests. Others allow the user to select which items will be displayed on the site's home page. Without a data-driven solution, it would be impossible to develop a site that could be tailored to display the latest items of interest each time a visitor returned.

Real-Time Information Systems

There's nothing wrong with collecting information about customers, as long as that information is not misused, or sold without permission. Building a database of customer habits is useful not only as a mechanism for feedback to management, but also as a means of providing individually tailored service to repeat customers. Likewise, field sales information, design and production schedules, product inventory, online purchases, sales transactions, marketing research, and competitive information all have an impact on your ability to provide service to your customers. Making that information available in real time (or as a continuous up-to-date flow of information) dramatically affects your ability to make rapid and incremental improvements to your business processes.

Data Warehousing and Online Analytical Processing

As organizations collect increasing volumes of data, the need to analyze and derive conclusions from raw data becomes more acute. A data warehouse is often used as the basis for a decision-support system (also referred to as a business intelligence system). It is designed to overcome some of the problems encountered when an organization attempts to perform strategic analysis using the same database that is used to perform online transaction processing (OLTP).

Typically, OLTP systems are designed specifically to manage transaction processing and minimize disk storage requirements using a series of related and normalized tables. However, when users need to analyze their data, a myriad of problems often prohibits the data from being used:

- Application databases can be segmented across multiple servers, making it difficult for users to find the data in the first place.
- Users may not understand the complex relationships among the tables, and therefore cannot generate ad hoc queries.
- Security restrictions may prevent users from accessing the detailed data they need. Database administrators prohibit ad hoc querying of OLTP systems. This prevents analytical users from running queries that could slow down the performance of mission-critical production databases.

Data warehousing offers a viable solution to these problems. Basically, data warehousing is an approach to storing data in which heterogeneous data sources from across the enterprise (typically from multiple OLTP databases) are migrated to a common homogenous data store. Sometimes organizations maintain smaller, more topic-oriented data stores called data marts. Whereas data warehouses or data marts are the stores for data, online analytical processing (OLAP) is the technology that enables client applications to efficiently process the data. Data warehouses (combined with OLAP) provide the following benefits to analytical users:

- Differences among data structures across multiple heterogeneous databases can be resolved. Data transformation rules can be applied to validate and consolidate data, when data is moved from the OLTP database into the data warehouse.
- Data is organized to facilitate analytical queries rather than transaction processing. Frequently-queried data is pre-aggregated and the results are stored as "cubes," which are table-like structures that enable very fast response time to ad hoc queries.
- Security and performance issues can be resolved without requiring changes in the production systems. OLAP is a powerful tool for creating new views of data, based upon a rich array of ad hoc calculation functions.

Microsoft Data Access Components

The previous section discussed the benefits of providing database access solutions on the Web. This section discusses how you can develop these solutions using Microsoft data access technologies.

Universal Data Access is Microsoft's initiative for providing high-performance access to all types of information (including relational and nonrelational data) across an organization, from the desktop to the enterprise, using practically any tool or language.

In a parallel effort with the Microsoft® Windows® Distributed interNet Applications (Windows DNA) architecture, the Universal Data Access initiative is a platform, application, and tools strategy that defines and delivers standards and technologies essential for application development.

MDAC is comprised of the data access technologies that enable Universal Data Access: ODBC, OLE DB, and Microsoft® ActiveX® Data Objects (ADO). Figure 7.1 shows how these data access components interact. For the latest information and current releases of MDAC see http://www.microsoft.com/data.

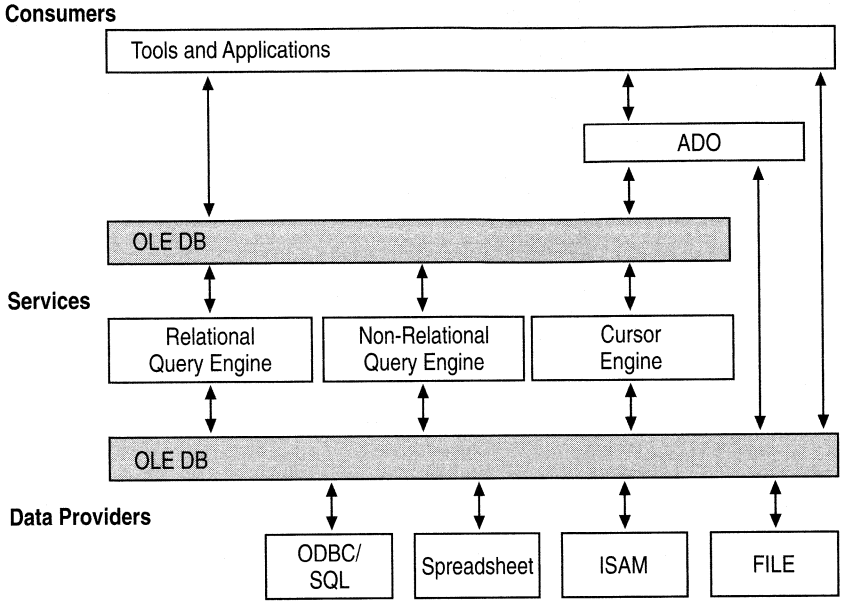


Figure 7.1 Microsoft Data Access Components

The next few sections describe each of the MDAC technologies. Most of your Web applications will use ADO, which is described in detail later.

ODBC and OLE DB

The *ODBC* standard is a widely recognized method of accessing data in a variety of relational databases. It is fast, lightweight, and provides a common method that is not optimized for any specific data source.

Like ODBC, *OLE DB* is an open specification. It was designed to build on the success of ODBC by providing another standard for accessing data. Whereas ODBC was created to access relational databases, OLE DB interfaces are designed to communicate with any data source including relational and nonrelational data, such as Microsoft® Excel spreadsheets, as well as e-mail, and text files. There is no restriction on the type of data you can access with OLE DB—relational databases, indexed sequential access method (ISAM), text, or hierarchical data sources.

OLE DB is set of programming interfaces designed for driver vendors who want to expose a data source, and for C++ developers wanting to develop custom data components. Microsoft® Visual Basic®, which does not support Automation objects, cannot use OLE DB directly.

Applications that use OLE DB fall into two categories: *consumers* and *providers*. A consumer application uses (or consumes) data through the OLE DB interfaces or components. A provider is any component or data source that allows consumers to access data in a uniform way through the OLE DB interfaces. In a sense, an OLE DB provider is similar to an ODBC driver that provides a uniform mechanism for accessing relational data.

Several OLE DB providers are available, including the Microsoft® ODBC provider, which exposes any ODBC-compliant database through OLE DB. Developers can implement an OLE DB provider for whatever data access they require, if one does not already exist.

ADO and RDS

ADO and *Remote Data Service (RDS)* use OLE DB providers to communicate with local and remote data sources respectively. Any application that uses ADO objects gets its data indirectly from OLE DB. If there is an OLE DB provider for it, the data is accessible through ADO.

You can use ADO to write both server-side and client-side applications that can access and manipulate data. ADO, designed to provide a universal high-level data access method, is a collection of Automation objects that can retrieve, update, and create records from any OLE DB provider.

ADO exposes a set of functions that all data sources are expected to implement. Using these core functions, ADO can access the unique features of specific data sources through OLE DB. Additionally, unlike earlier data access methods, you no longer need to navigate through a hierarchy to create objects. You can create most ADO objects independently and reuse them in different contexts. If used correctly, the result is fewer ADO object calls and a smaller working set.

There's a downside to all this flexibility, however. Because ADO is an OLE DB consumer, the peculiarities of the OLE DB provider that you are using directly influence the behavior of ADO. Just because you can write an application in ADO doesn't mean that the provider will support it. Often, ADO errors are a direct result of performing operations not supported by the OLE DB provider, or the underlying data source. As you develop database access components and applications, keep in mind that there are sometimes multiple ways to perform any given action.

RDS is a feature of ADO that facilitates client-side programming by optimizing the transfer of data between the client and the ADO components in the middle tier of a Web application. RDS uses ADO as a programming interface between the code and the data exposed by the underlying OLE DB provider. The client-side components of RDS are Microsoft® ActiveX® controls that use either Microsoft® Component Object Model (COM) components or Hypertext Transfer Protocol (HTTP) to communicate with the server components. Microsoft® Internet Explorer includes the RDS client-side components.

> **Note** For Microsoft® Internet Explorer 3.x users, MDAC 2.0 provides service components that are compatible with RDS server-side and client-side components. Client-side components are included with Microsoft® Internet Explorer 5. Since later versions of MDAC are not 100 percent compatible with earlier versions of the browser, you might need to upgrade your clients before using the more advanced features of RDS and ADO.

A detailed look at RDS and ADO is included in the section "Client-Side Data Access" later in this chapter.

Other Data Access Methods

In today's fast-paced technological arena, it is not surprising to find that last year's good idea is this year's outdated software. For the sake of those that cannot throw away older technology just to adopt the latest craze, the following section enumerates some older data access technologies that are still supported by IIS 5.0. However, unless you have a good business reason to use them, you should rely on ADO instead. ADO is designed to balance flexibility with programmatic simplicity. In most cases, it is the only data access method you will need.

ADC

The *Advanced Data Connector (ADC)* can be considered the parent of RDS. In fact, the RDS technology used to access remote data is inherited directly from ADC. The early design of ADC was less flexible than the ADO programming model, so it was integrated with ADO to provide a uniform means of accessing remote data. ADC itself is now considered obsolete; RDS (and ADO, which is used by RDS in the middle tier) has replaced ADC programming.

Instead of ADC, use RDS when you need to provide a common programming model for accessing either local or remote data. RDS objects are installed with Microsoft® Internet Explorer 4.0 and Internet Explorer 5 on your client, or you can download them at run time from the .cab files shipped with MDAC components.

Jet Database Engine and DAO

The Jet database engine is a workstation-based storage system. Jet databases can be accessed using *Data Access Objects (DAO)*. You can also access Jet databases with the ODBC drivers provided with Access, but only limited functionality is exposed using these drivers. The Jet database engine has its own query and result-set processors and is capable of executing queries against homogeneous or heterogeneous data sources. Developers who are familiar with DAO can use ODBCDirect to bypass the Jet database engine when connecting to back-end data sources.

DAO requires that you change programming models depending on whether your data is stored in a Jet database or some other data store. ADO provides a common programming model for Jet databases or any other OLE DB data source.

RDO

The *Remote Data Objects (RDO)* were specifically designed to access remote ODBC relational data sources, and to add a thin object layer to the ODBC application programming interface (API). RDO performance is, in most cases, close to that of the ODBC API.

RDO was specifically designed to deal with remote, intelligent data sources (such as SQL Server or Oracle, as opposed to ISAM databases), so it does not support some of the DAO table-based interfaces or Dynamic Data Exchange (DDE). RDO can execute ordinary table-based queries, but it is especially adept at building and executing queries against stored procedures. It also handles all types of result sets including those generated by multiple result set procedures, those returning output arguments, and those requiring complex input parameters. RDO 2.0 provides a high level of control over remote data sources, so it is not necessary to expose the underlying ODBC handles in order to manipulate the data sources, except in the most unusual cases. It also can create client cursors to manage "disconnected" result sets.

Once again, newer technologies have surpassed older ones. For example, ADO provides equivalent functionality and performance to RDO, with an easier-to-use object model; ADO can also access a much larger variety of data stores.

The Cost of Data Access

Technology, like MDAC, comes with a price.

Suppose you're creating a site that publishes bus schedules for more than a hundred routes. The "static" solution might be an index page—perhaps with an HTML form—that allows the user to select and view route-specific pages. The "dynamic" solution might use a query page to look up each bus schedule as it is requested, and return it on a customized, dynamic page.

Both approaches offer the same solution, but the "static" one offers better performance, for two reasons:

- Delivering a static page demands far less processing than creating the same page on the fly.
- The static page solution creates each page once; the dynamic page solution might create a new page for each request (depending on how the server's cache is utilized) for information that generally doesn't change much.

Consider how your data will typically be used. Often a static approach is best for static data (such as bus schedules), and a dynamic one is generally best for dynamic data (such as stock quotes). However, the best solution is to provide a controlled mix of static and dynamic pages, as your users require them, that your site can support. For example, if people need infrequent access to large amounts of data, the best solution may be the dynamic approach: a query page. But if they can read a relatively short list of articles while online, it's better to generate the page once and display a static listing.

Once you've determined that the dynamic approach is necessary, choosing a data access method is based on what your application requires in the following areas:

- Performance
- Ease of development
- Ease of maintenance
- Ease of deployment

The emphasis you place on each factor should be motivated by both the current and future needs of your application. Given the robust growth patterns of the Web in recent years, application performance and scalability should be your foremost goals. It is important to note, however, that throughput numbers should in no way dictate your choice of technology. For example, although the cost of developing an ADO application can be significantly less than with ODBC, the number of transactions ADO can process per second (TPS) is significantly lower than what ODBC can achieve. If both ADO and ODBC exceed your target TPS, your decision on which technology to use should be based on factors other than performance.

Table 7.1 compares the TPS results of similar tests performed with each data access technology. For these tests, Component Services (implemented in Microsoft® Visual C++®) and connection pooling were used. The SQL Server database was scaled to 800 TPS, with 384 megabytes (MB) RAM, and 4 percent procedure cache.

Table 7.1 TPS Per Number of Threads by MDAC Technology

	1	2	5	10	20	50
ODBC	66.37	146.28	350.46	626.76	900.24	859.91
OLE DB	67.30	141.92	326.19	590.57	794.91	715.78
OLE DB 2.0	61.73	126.93	297.29	506.75	575.35	526.61
ADO 2.0	51.24	108.12	240.91	377.30	361.26	310.34

The results of using relational data are striking (the comparison is not valid if the data is not relational). The ODBC component produced the highest throughput (as much as 277 percent higher than ADO using 50 threads). All data access technologies increased in throughput up to approximately 20 threads, after which performance dropped off.

These test results make the cost of data access very clear. If you use data access frivolously, you will be plagued with excessive delays and bottlenecks as pages are generated over and over again, and as transactions are processed for thousands of concurrent requests. By using the right data access methodology, and judiciously choosing when to generate dynamic pages, you can enhance and complement the static elements of your site.

Client-Side Data Access

The components that make up MDAC are designed for distributed applications that take advantage of the processing power on client, middle-tier, and database server computers. These components are part of a simple yet rich programming model for manipulating data and building applications that are easy to configure and maintain.

ADO and RDS are suitable for applications that need a high degree of database accessibility. When you create an RDS application for an intranet, you normally would use IIS 5.0 and Internet Explorer to communicate between the client and business services tiers. However, if the client- and middle-tier components are on computers within a local area network (LAN), you can also use COM to marshal the interfaces and method arguments across the network. RDS can also be used by applications written in Visual Basic to manage remote information.

ADO, with RDS, enables the Web-application developer to:

- Bind ADO **Recordset** objects to intrinsic Dynamic HTML (DHTML) controls (and other data-aware controls) hosted in the browser, by using the DHTML "databinding" model.
- Create and manage remote and disconnected ADO **Recordset** objects. ADO normally maintains a persistent connection to the database, but RDS works with locally cached, or disconnected, data.
- Asynchronously request information from the server, and respond to events that are triggered when operations are complete. Asynchronous fetching is a feature specific to Microsoft® Cursor Service for OLE DB. It returns the first rows from a query and then continues fetching in the background, while the user manipulates the rows that have already been sent.
- Invoke Automation objects on the Web server over either HTTP or COM. You can use RDS for applications that browse records, or that connect to and modify data on the middle tier.
- Work with hierarchical and multidimensional recordsets. With the newly available Microsoft® Data Shaping Service for OLE DB (Microsoft® Datashape), RDS can finally overcome its previous limitation of only one recordset per query.

The RDS client-side and server-side components are described in detail in the following sections.

Client-Tier Elements

As described in the previous chapter, the client services tier provides the visual interface for presenting and gathering data. In a Web-based RDS application, the Web page represents the RDS front end. The RDS client tier contains the following components:

- A Web page or Visual Basic application containing an **RDS.DataControl** object and one or more data-aware controls.
- An **RDS.DataSpace** object and client-side proxy for middle-tier business objects. The default business object is **RDSServer.DataFactory**. You can develop custom business objects as well.
- A data cache and the client-side cursor engine, Cursor Service for OLE DB.

Data-Aware Controls

You can bind data-aware controls (for example, a DHTML text box, FlexGrid control, and so on) to data from remote servers, allowing you to view, edit, and update data from a Web page. RDS is only one of several data binding controls that can execute query results and bind them to other DHTML elements in Internet Explorer.

In RDS, the data-binding mechanism is the **RDS.DataControl** object. This is a nonvisual control that can bind a single ADO **Recordset** object, which represents the results of a query, to one or more data-aware controls on a Web page. It is possible to have more than one **RDS.DataControl** object on the page. The following HTML establishes a connection to the MySite server, and selects all the authors from the Pubs database:

```
<!-- RDS.DataControl -->
<OBJECT ID=RDS classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33">
  <PARAM NAME="SQL"      VALUE="Select * from Authors">
  <PARAM NAME="SERVER"   VALUE="http://MySite">
  <PARAM NAME="CONNECT"  VALUE="dsn=Pubs;UID=sa;PWD=;">
</OBJECT>
```

When you use the **RDS.DataControl** to obtain a disconnected **Recordset**, RDS calls the **RDSServer.DataFactory** object on the middle tier for you. You don't need to call **RDSServer.DataFactory** or **RDS.DataSpace** explicitly.

When the RDS proxy invokes the **RDSServer.DataFactory** object on the server, the server-side object creates an ADO **Recordset** and executes the query against the specified data source. The results are then transmitted to the client where the **RDS.DataControl** reconstructs the **Recordset** for the client application. Once the data has been retrieved, the client disconnects from the data source. This helps to eliminate the contention for database connections that sometimes occurs as multiple clients simultaneously access the same data source.

Figure 7.2 shows a data-bound grid control in Internet Explorer. When the page loads, the grid control is automatically filled with records from the Adventure Works database. The following code listing shows the Microsoft® Visual Basic® Scripting Edition (VBScript) used to create this page. Note that very little of the script actually executes on the server; most of it operates using objects created on the client.

Example: The Sheridan Grid Control

The example uses the Sheridan data-bound grid control, which is available from the Sheridan Web site, http://www.shersoft.com, or as part of the MDAC samples. If you have installed the MDAC samples, the .cab file for the grid control can be found at http://<yourserver>msadc/Samples/ssdatb32.cab.

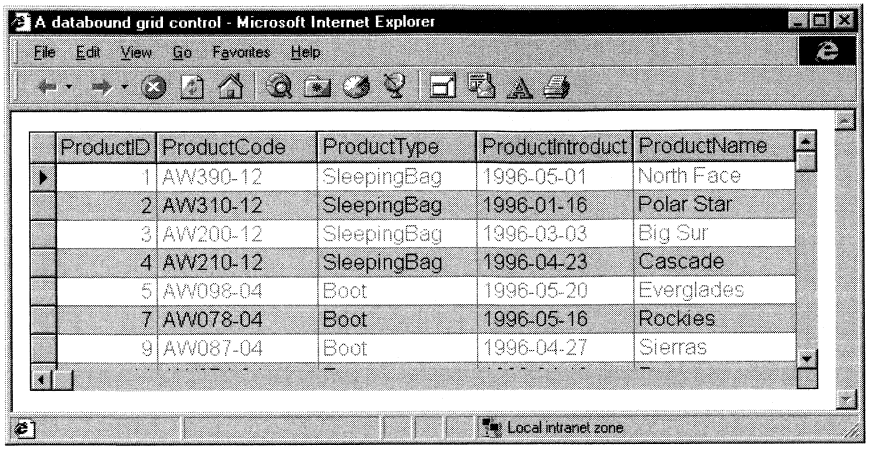


Figure 7.2 A Data-Bound Grid Control in Internet Explorer

```
<%@ LANGUAGE=VBScript EnableSessionState=False %>
<!-- #include file="adcvbs.inc" -->
<HTML>
  <HEAD>
    <TITLE>A databound grid control</TITLE>
  </HEAD>
  <BODY BGCOLOR=#FFFFFF TEXT="#000000">

    <!-- RDS.DataControl -->
    <OBJECT ID="DATA" WIDTH=1 HEIGHT=1
    CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33">
    </OBJECT>
    <!-- Sheridan data-bound grid control (note DATASRC) -->
    <OBJECT ID="GRID" WIDTH=600 HEIGHT=200 DATASRC="#DATA"
    CLASSID="clsid:00028C00-0000-0000-0000-000000000046">
    </OBJECT>
    --><SCRIPT Language=VBScript>
    Sub Window_OnLoad
    If DATA.ReadyState <> <%=adcReadyStateComplete%> Then
    MsgBox "Query results still arriving, Please wait"
    Else
    DATA.Server  = "http://<%Request.ServerVariables("SERVER_NAME")%>"
    DATA.Connect = "DSN=AdvWorks"
    DATA.SQL     = "Select * from Products"
    DATA.Refresh
    GRID.Rebind
    End if
    END SUB
    </SCRIPT>
  </BODY>
</HTML>
```

Each **RDS.DataControl** object can be connected to a different data source and contain the results of a separate query. When you use the **RDS.DataControl** object to send a query to a database, the RDS server-side components return an **ADODB.Recordset** object to the client. On the client, the RDS proxy creates a client-side cursor and an **ADOR.Recordset** object to manipulate the returned records. You don't necessarily have to write any ADO-specific code to make this happen—RDS handles this for you when you use the **RDS.DataControl** object.

Figure 7.3 shows how the client-side and server-side components of a Web-based ADO application work together to process a user's query and display information from a database.

1. The user enters the query text, chooses a preformatted request, or navigates to a page containing an embedded query.
2. When an event fires on the Web page, such as the **Window_OnLoad** routine or **OnClick** event of a Search button, ADO creates a **RDSServer.DataFactory** proxy (or business object proxy) on the client.
3. The proxy translates the user request into an HTTP request, by formatting the parameters of the business object method as URL parameters. It then sends the request to the Web server specified in the **RDS.DataControl Server** property. IIS 5.0 forwards the HTTP request to an Internet Server Application Programming Interface (ISAPI) extension.
4. Advanced Data ISAPI (ADISAPI) interprets the URL parameters, creates an instance of the requested business object, and makes the method call. (By default, it calls the **Query** method of the server-side **RDS.DataFactory** object.)
5. The **RDS.DataFactory** object executes the user's query via OLE DB. It sets the **CursorLocation** property of the **Recordset** so that Cursor Service is used as its buffering component.
6. OLE DB passes the complete results of the query to the Cursor Service, which populates its buffers with the data, including all of the metadata for tables that are part of the result set.
7. The Client Cursor Engine passes a reference to the result set back to the **RDS.DataFactory** object.
8. The **RDS.DataFactory** object creates an instance of an ADO **Recordset** and initializes it with the result set. The ADO **Recordset** is then passed back as the return value of the original **Query** call from step 4.
9. The **RDS.DataFactory** object passes the **Recordset** back to ADISAPI, which packages the return value of the call into Multipurpose Internet Mail Extensions (MIME) format.
10. ADISAPI sends the **Recordset** over HTTP as multipart MIME packets to the business object proxy on the client side.

11. The client-side proxy unpacks the results of the method call, and recreates the **Recordset** in the client-side Data Cache.
12. Finally, the embedded **RDS.DataControl** object binds the data in the client-side Data Cache to the visual controls.

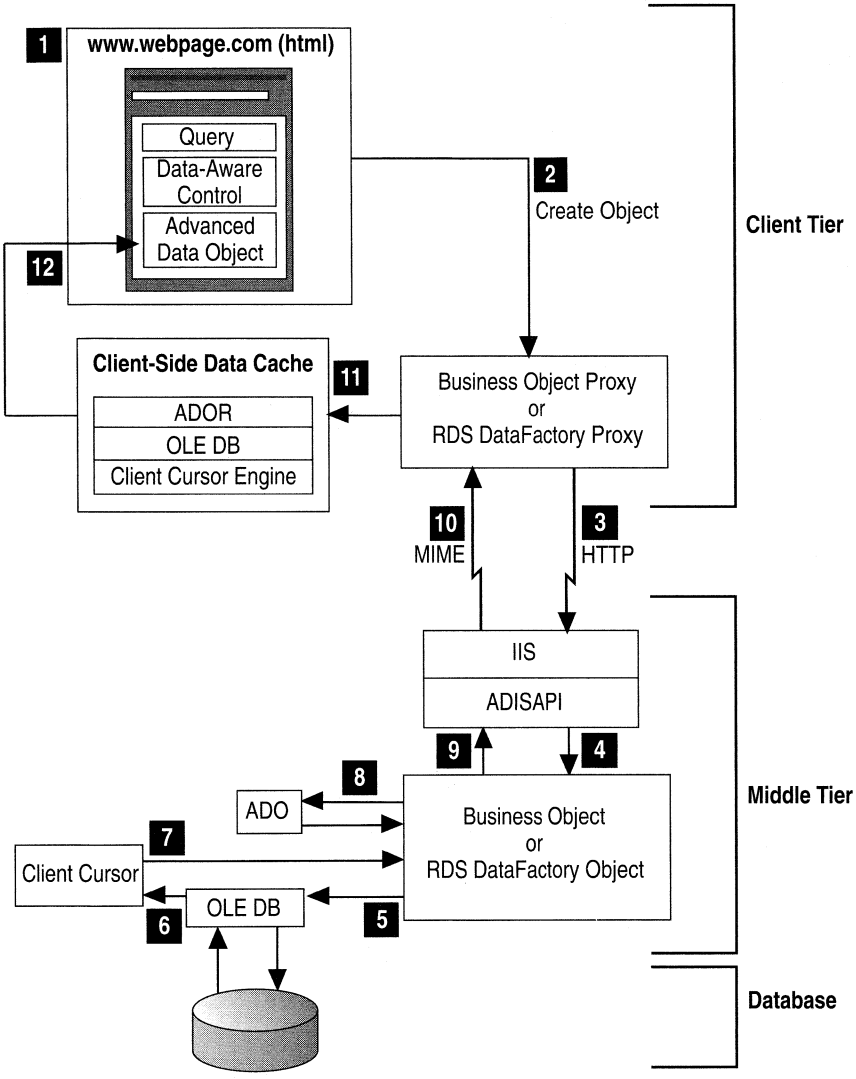


Figure 7.3 Process Diagram of an ADO Application Using RDS for Data Partitioning

Data Cache

One of the most important features of the RDS is its in-memory data caching on both the client and middle tiers. The Microsoft® Cursor Service caches the set of records returned by the query on the client computer. The only client actions that require another trip to the server are updates to the data or requests for new data. The client-side data cache does the following:

- Reduces the number of requests for data between client-side application components and the database server. The performance improvement is especially noticeable for data access across the Internet.
- Makes data immediately available to client-side application logic, without the application having to wait for data to travel across the network.

Because the data is cached on the client workstation, a user can quickly sort, filter, and scroll through the data without another round-trip to the server.

Client Cursor Engine

Remote Data Service calls Cursor Service (invisible to the user) to perform tasks for you automatically. It caches, in memory (or temporarily on disk, for large sets of data), the set of query results retrieved from a data source, as well as client updates to those results. It also contains layout information about the data such as table layouts, row counts, primary and secondary keys, column names, and timestamps, as well as the actual table data itself. To manage the cache, Cursor Service can:

- Create and delete temporary tables.
- Populate tables.
- Manage updates to the data values.
- Provide schema information (such as base tables, key columns, read-write columns, and computed columns).
- Provide the mechanism to send updates as a batch to the server, with minimum network traffic, by sending only the modified records.

For more information about the differences between the client-side and server-side cursor engines, see "Recordsets and Cursors" later in this chapter.

Business Object Proxies and the RDS.DataSpace Object

RDS uses proxies for business objects that enable client-side components to communicate with business objects located on the middle tier. Proxies facilitate the packaging, unpackaging, and transport (marshaling) of the application's data across process or computer boundaries.

Like the **RDS.DataControl** object, the **RDS.DataSpace** object is a nonvisual ActiveX control that creates client-side proxies for custom business objects located on the middle tier. Client-side proxies facilitate packing, transporting, and unpackaging of disconnected recordsets (and other standard data types) across machine boundaries. For example, the following client-side script instantiates a Customer component using an **RDS.DataSpace** object:

```
<!-- RDS.DataSpace object -->
<OBJECT ID=RDSDataSpace WIDTH=1 HEIGHT=1
CLSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E36">
</OBJECT>
<SCRIPT LANGUAGE=VBScript>
Set objProxyCust = RDSDataSpace.CreateObject("MyCls.Customer", & _
"http://<%=Request.ServerVariables("SERVER_NAME")%>")
Set rs = objProxyCust.Orders(Request.Form("UserID"))
</SCRIPT>
```

Note In order to create server-side object proxies with the **RDS.DataSpace** control, the ProgID of the business component must be registered in the key in the Microsoft® Windows® 2000 Server registry.

HKEY_LOCAL_MACHINE
    \SYSTEM
        \CurrentControlSet
            \Services
                \W3SVC
                    \Parameters
                        \ADCLaunch

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft® Management Console (MMC) whenever possible.

Middle-Tier Elements of Client-Side Data Access

The middle tier is the "bridge" between client computers and database servers. The components on the middle tier respond to requests from the user (or business services) in order to execute a business task.

In an RDS application, business objects on the middle tier handle the data request from the client sent through a transport protocol such as HTTP. Data, in the form of **Recordset** objects, is made available as an update to client-side controls through the OLE DB provider. The middle tier consists of the following components:

- IIS 5.0 and ADISAPI
- **RDSServer.DataFactory** or business objects

IIS 5.0 and ADISAPI

The underlying RDS code uses an ISAPI extension that helps to create server-side stubs, in order to communicate with client-side business object proxies. The ADISAPI breaks up the MIME packets that pass through IIS 5.0 and invokes methods on the server-side Data Factory objects.

The ADISAPI component communicates with business objects for you, and provides parsing, automation control, and **Recordset** object marshaling; it also provides tabular data packaging, streaming, and unpackaging. This ISAPI extension performs the necessary work to instantiate business objects, invoke their services through automation interfaces, and process the return parameters for transport back to the calling client through the tabular data streaming protocol.

> **Note** Requests to the ADISAPI component are made through the **msadc** virtual directory of IIS 5.0. This directory is created in the default Web site when IIS 5.0 is installed. If the default Web site has been renamed or replaced (for example, to improve administrative control of installed Web applications), the developer must recreate the **msadc** virtual directory on the new default Web site. The other option would be to specify the port address of the site that includes it in all RDS client-side requests.

The RDS Data Factory and Custom Business Objects

There are two ways to pass a **Recordset** object back from your server to the client with RDS. You can:

- Use the **RDSServer.DataFactory** object.
- Create a custom business object that exposes data access methods.

RDS includes a default business object, the **RDSServer.DataFactory** object, that provides read and write access to data sources, but contains no business rules or application logic. The **RDSServer.DataFactory** is a server-side business object—specifically an ActiveX dynamic-link library (DLL)—that uses ADO to dispatch SQL statements to a database management system (DBMS) through the OLE DB provider, and packages the results for transfer across the Internet or an intranet.

Designing Custom Business Objects

The main application components on the middle tier are business objects that contain information such as application logic, business rules, and specialized data-access code, in order to retrieve information from underlying databases. Business objects can be created with Visual Basic, C++, or any other language that supports COM.

The life span of a business object is as long as the execution of the method call invoked by the client-side **RDS.DataSpace** object. Business objects are created and destroyed with each method call, so no interim state is maintained. Therefore, custom business objects designed for use with RDS need to be stateless (don't use properties or static data). If state needs to be saved between invocations to the object, consider using the Shared Property Manager. A feature of Component Services, the Shared Property Manager simplifies programming of components that need to save state information or share state with other components. For more about working with business objects, see "Transactional Components" later in this chapter.

When designing business objects, keep the following in mind:

- RDS can only marshal back one **Recordset** at a time. Beware of procedures that return two or more result sets, since they will require special handling by your business object. You can use the new MSDataShape provider to return hierarchical **Recordset** objects, but the data shape query language doesn't support stored procedures at this time.
- RDS cannot marshal user-defined data types. Return only basic data types (Integer, String, and so on), **Recordset** objects, or Variants. Variant arrays are also supported.
- Pass and return ADOR recordsets, not ADODB.
- Use *BYVAL* parameters whenever possible. *BYREF* (by reference) parameters require two network trips, so using the *BYVAL* keyword to pass parameters by value saves time. *BYREF* is the default behavior of Visual Basic, so you must type **BYVAL** wherever you need it.
- Business objects created with the **RDS.DataSpace** object cannot access ASP objects, including Session or Application state.

You can use either RDS or ADO, or a combination of both technologies, in the same application. For example, your application can create a custom business object that uses ADO to manage server-side data, and that uses RDS to transfer remote data to the client tier, where the user can interact with it.

The next section discusses the issues involved in creating a server-side ADO application, using ASP and COM components.

Accessing Data with ASP and COM Components

When you create a data-enhanced application, you need to decide whether data manipulation functionality or far-reaching browser support is more important to you. Unlike RDS, a server-side ADO solution is browser-independent. Because ADO can be accessed from server-side ASP and COM components, it doesn't require any ActiveX controls on the client side—a serious consideration for mixed-browser environments like the Internet.

Preparing the Database

Before you can connect to a database using ADO, you must take a few preliminary steps to properly configure your data source.

Connection Strings

ADO uses OLE DB connection strings to define the specifics of the data source. The connection string consists of a series of keyword/value pairs separated by semicolons, which specify information used for establishing a connection to the data source. Such a string might look like this:

```
"Provider=sqloledb; Driver={SQL Server}; Server=MySite;" & _
"Initial Catalog=pubs; User Id=sa; Password=;"
```

When using the OLE DB provider so that ODBC can connect to an ODBC driver through ADO, elements of the OLE DB connection string are passed down to the driver for the data source.

Creating an ODBC-Compliant Data Source

When designing a Web database application, it is often easier to initially prototype with a desktop database like Access, and then scale up to an enterprise-level database like SQL Server once the design work has been finished. This is an easy alteration to make if you are using script, but it would require you to recompile your business object DLLs. With a connection string, you can make such a change without altering code in your application.

When you use an ODBC Data Source Name (DSN) entry to store the connection string values externally, you simplify the information needed in the connection string. This makes changes to the data source completely transparent to the code itself.

To create a new ODBC DSN

1. Click the **Start** menu button, point to **Programs**, then **Administrative Tools.**
2. Click the **Data Sources (ODBC)** icon, and select the **System DSN** tab.
3. Click **Add** to create a new DSN entry, and then select the appropriate OLE DB provider for your database from the list of installed providers. Click **Finish** to configure your provider.

   Note Make sure you create either a File or System DSN. ADO does not recognize user (or local) DSNs. Because they store settings in the system registry, System DSNs offer slightly faster performance than File DSNs, which store connection parameters in a file on your hard disk.

"DSN-less" Connections

You can open a connection to a database without creating or specifying a named data source. Connections made in this way are called "DSN-less," because they don't require the system administrator to create an ODBC DSN. Rather than relying on information stored in a file or in the system registry, DSN-less connections specify the driver name, and all driver-specific information in the connection string.

Whether or not to use a DSN depends on how much flexibility you want. Connection parameters are readily visible to system administrators. As a result, no matter which connection style you use, there are no extra security benefits. Probably the most common reason for using a DSN-less connection is to connect to a database on a system that is not under your direct control. This makes DSN-less connections good for testing and for applications under development.

The following DSN-less connection strings contain the minimum parameters required by the ODBC drivers of Access and SQL 6.5:

```
strConAccess = "Driver={Microsoft Access Driver (*.mdb)};DBQ=C:\db.mdb"
strConSQL    = "Driver={SQL Server};Server=(local);UID=sa;PWD="
```

Essentially, a DSN-less connection is "hard-coded" to use a certain driver, user identity, and network location, which makes it bothersome and difficult to update when the database parameters change. Because the variety of connection parameters can differ greatly from one ODBC data source drive to another, it is recommended that you use a DSN whenever possible.

Selecting an OLE DB Provider

A provider is a component that manages access to a data source. The Access and SQL Server providers are only two of the OLE DB providers available with ADO. In most cases, you don't have to specify the OLE DB provider; you can just use the default ODBC driver. However, not all OLE DB providers are alike; some connect to nonstandard data sources and some have specialized functionality. Microsoft continues to update and add to the following list in Table 7.2 of OLE DB providers installed with ADO 2.0:

Table 7.2 Common OLE DB Providers

Driver	Description
MSDASQL	OLE DB provider for ODBC. Connects to existing ODBC data sources, either with a System DSN or from connection string parameters. This is the default, if no provider is specified.
Microsoft.Jet.OLEDB.3.51	Access data provider. Connects to Jet databases.
SQLOLEDB	SQL Server OLE DB data provider.
MSDAORA	OLE DB provider for Oracle.
MSIDXS	Provider for Microsoft® Indexing Service.
ADSDSOObject	Microsoft® Active Directory™ Services provider.
SNAOLEDB	OLE DB provider for Microsoft® SNA Server (requires SNA Server Client).
MSDataShape	Hierarchical recordset service provider.
MSPersist	Persisted recordset (local storage) service provider.
MS Remote	RDS disconnected recordset service provider.
MSDAOSP	Simple OLE DB provider. This provider can be used as the basis of custom providers that you create.

Choosing a data provider is as simple as setting the **Provider** property of the **Connection** object. The following ASP example uses a sample provider ("sampprov") to create a **Recordset** object from a Comma Separated Values (CSV) file:

```
Set cn = Server.CreateObject("ADODB.Connection")
cn.Provider = "sampprov"
cn.Open "Data Source=C:\oledbsdk\samples\sampclnt"
Set rs = cn.Execute("customer.csv")
```

Note The "sampprov" provider is shipped with OLE DB SDK 1.5 and later releases.

Data Source Permissions

Setting up a DSN is the first step toward connecting to a database. You also need the correct permissions before you can access data from the Web. If you don't have these and you try to open an Access database stored on an NTFS file system, you might get the following error:

```
Microsoft OLE DB provider for ODBC Drivers error '80004005'
[Microsoft][ODBC Access 97 Driver] The Microsoft Jet database engine cannot
open the file '(unknown)'. It is already opened exclusively by another user,
or you need permission to view its data.
```

This problem occurs when the anonymous (IUSR_*computername*) account doesn't have the necessary permissions for the directory containing the database (.mdb) file. In order to make modifications to the data, the Jet database engine (on which Access is built) needs to create a lock file, a temporary working file in the same directory as the database. If the anonymous user can't write to this lock file, an error will occur. Make sure that you grant sufficient permission on the database and its directory for the anonymous user account. Also verify that exclusive access and record locking are turned off.

Security and SQL Server

For SQL Server, permissions issues multiply, especially if SQL Server and IIS 5.0 are running on different computers. By default, connections to a SQL Server use a service of the Windows 2000 operating system known as a "named pipe." In order for a SQL server client to gain access to a Windows 2000 named pipe, the client needs to be validated by the server. This is normally accomplished either by means of a Workgroups-style validation (identical user names and passwords are created on the client and the server), or by using the domain method (both the client and server are domain members).

The SQL Server connection uses the identity of the user associated with the Web connection. If the connection is anonymous, you need to create a guest account that corresponds to the IUSR_*computername* account. If this guest account already exists, then make sure it has rights to log onto the SQL Server computer.

Support for the anonymous user account can be configured by any of the following means:

- Add the IUSR_*computername* account to the local user account database on the server that hosts SQL Server.
- Make the account a member of the domain that SQL Server resides in.
- Enable the Windows 2000 Guest user account on the remote SQL Server computer.

If you have configured IIS 5.0 to use integrated Windows authentication in the Windows 2000 operating system (by either disabling Anonymous access, or forcing a logon by returning a "401 Access Denied" response), IIS 5.0 tries to connect to the SQL Server by using the user's security context. See Figure 7.4. If SQL Server resides on a separate computer from IIS 5.0, the Windows 2000 operating system detects the attempt to use a network named pipe handle that had been opened in a different user context. It then forces the pipe closed, according to its security rules.

The following OLE DB error is an indication of this problem:

```
Microsoft OLE DB provider for ODBC Drivers error '80004005'
[Microsoft][ODBC SQL Server Driver][DBNMPNTW]ConnectionWrite (GetOverLappedResult()).
```

If SQL Server is running on the *same* server as IIS 5.0, use a local named pipe connection instead of a network named pipe connection. In the SQL Server connection string or in the DSN configuration, change "SERVER=*computername*" to "SERVER=(local)." The name "(local)" (with parentheses) is a special keyword to the SQL Server ODBC driver, and indicates that a local connection should be used.

If SQL Server is running on a *different* server than IIS 5.0, you can use a nonauthenticated protocol between IIS 5.0 and SQL Server, such as Transmission Control Protocol/Internet Protocol (TCP/IP) sockets. To use these, you must configure both the SQL Server and the SQL Server client on the IIS 5.0 server.

To set up a TCP/IP connection on the server hosting SQL Server

1. Run SQL Server Setup.
2. In the **Options** dialog box, click **Change Network Support**, and click **Continue**.
3. Select the entry for TCP/IP Sockets (leave Named Pipes selected also) and click **OK**. Accept the default Named Pipe name and TCP/IP Socket number.
4. Exit SQL Server Setup. Stop and restart the SQL Server.

To set up a TCP/IP connection on the server hosting IIS 5.0

1. From Control Panel, point to **Administrative Tools**. Select **Data Sources (ODBC)**. Select a SQL Server data source, and click **Configure** to start the SQL Server DSN Configuration Wizard. Click **Next**, then click **Client Configuration**. Figure 7.4 shows the resulting dialog box.
2. Select the **Network Library** tab, and select **TCP/IP** as the default network protocol. Click **OK**. IIS 5.0 will now use TCP/IP sockets when connecting to the SQL Server specified in this DSN.

Figure 7.4 Configuring the SQL Server Client Connection

The Database Connection

There are several ways to establish a database connection. One way is to create an ADO **Connection** object explicitly, and use its **Open** method to connect to the database using a DSN, user name, and password. The following example shows you how this would look in an ASP page:

```
<%
'Open the database.
Set cn = Server.CreateObject("ADODB.Connection")
cn.Open strDSN, strUserName, strPassword
%>
```

You can also dynamically create a connection when you create a **Recordset** or **Command** object by using a connection string in place of a **Connection** object. When called, the object creates a new database connection. The following example uses a **Recordset** object in conjunction with a connection string in order to create a new database connection:

```
'Open the database and retrieve the records.
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open strSQL, strDSN
```

> **Note** When you bypass the **Connection** object in this way, you run the risk of inadvertently creating multiple database connections, which uses extra server resources and slows down the processing of your queries. This is particularly a problem when ODBC Connection Pooling is disabled. If you have already created a **Connection** object, it is always more efficient to reuse it when creating new ADO objects.

ODBC Connection Pooling

Database applications that frequently open and close connections can reduce database server performance. Fortunately, ODBC 3.5 implements *connection pooling*, which enables applications to share connections across user requests, and thereby improves performance by reducing the number of idle connections.

Before making a new connection to an ODBC data source, the ODBC driver searches the connection pool for any idle connections that may satisfy the connection request. For the connection to be reused, the connection string and user context of the pooled connection must exactly match that of the request. If a matching idle connection is found, it is returned to the calling application.

When an ODBC connection is released, the connection returns to the pool, rather than being closed. The **Connection Pool Timeout** (**CPTimeout**) setting of your ODBC driver determines the length of time that a connection remains in the connection pool. If the connection is not reused during this time, it is closed and removed from the pool. The default CPTimeout value is 60 seconds.

You can modify the CPTimeout value of each ODBC database driver by changing the following Windows 2000 Server registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE
    \ODBC
        \ODBCINST.INI
            \driver-name
                \CPTimeout = timeout (REG_SZ, units are in seconds)
```

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

For example, the following setting configures the ODBC driver for the SQL Server connection pool time-out to 180 seconds:

```
HKEY_LOCAL_MACHINE\SOFTWARE
    \ODBC
        \ODBCINST.INI
            \SQL Server
                \CPTimeout = "180"
```

A severed connection to the database could prevent applications that use a connection pool from connecting successfully. Client applications would continue to make new connection requests even though the connection is broken. It takes time for each request to determine that the server is unavailable, and new connection requests must wait for others to time out. Eventually, the server will be unable to accept any more requests.

The ODBC Connection Manager can be configured to retry dead connections on a preset interval. If the connection attempt fails, the connection is marked as "bad" and placed back in the pool. Once a bad server has been identified, subsequent connection requests for that server *immediately* return an error. Periodically, the Connection Manager attempts to re-establish the connection. If the attempt succeeds, the connection returns to a valid state and normal processing resumes.

You can configure the **Retry Wait** property for a specific ODBC driver by creating a registry key with the following settings:

```
HKEY_LOCAL_MACHINE\SOFTWARE
    \ODBC
        \ODBCINST.INI
            \driver-name
                \Retry Wait = timeout (REG_SZ, units are in seconds)
```

The following setting instructs the ODBC driver for SQL Server Connection Manager to retry a dead connection after a 60-second wait:

```
HKEY_LOCAL_MACHINE\SOFTWARE
    \ODBC
        \ODBCINST.INI
            \SQL Server
                \Retry Wait = "60"
```

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

Every process uses its own connection pool. If you are using an out-of-process component (or LocalServer application), you cannot take full advantage of ODBC connection pooling. Because each application process uses a separate pool, your application can only share connections with itself. In order to share connections with other components, you must write your business logic as a DLL. OLE DB providers automatically handle connection pooling.

Tips for Optimizing Database Connections

One of the main challenges of designing a sophisticated Web database application is maintaining a properly managed database connection. Opening and maintaining connections, even when no information is transmitted, can deplete a database server's resources and result in connectivity problems. Well-designed Web database applications manage connections wisely and compensate for delays due to network traffic.

Here are six tips to help you optimize your use of database connections.

Tip 1: Enhance Performance on SQL Server Systems

If you don't intend to connect to Access databases with ADO, you can enhance your application's performance by changing the threading model for the main ADO components from "Apartment" to "Both" in the registry.

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

By default, the ADO objects are assigned the Apartment threading model in the registry. This model guarantees that each object is allocated a dedicated thread for the life of the object, and all calls to the object execute on the same thread. Although the Apartment model provides significant improvements over the single-threading model (in which many objects share one thread), and works with providers that are not free-threaded (like Access), it also has its performance drawbacks. For instance, if you store ADO components, such as the **Connection** object, in the ASP **Session** object, IIS 5.0 will enforce a limit of one thread per user session.

To switch ADO to a Both threading model, open **Windows Explorer** and double-click **Makefre15.bat** in the ADO installation folder (C:\Program Files\Common Files\System\Ado, by default). To reverse the process (that is, to return the threading model to the Apartment model), double-click **Makeapt15.bat** in the ADO installation folder.

Tip 2: Set the Connection Time-Out

Limit the amount of time your application waits before abandoning a connection attempt and issuing an error message.

A database server experiencing a sudden increase in activity can become backlogged, which greatly increases the time required to establish a database connection. Excessive connection delays increase the time that users wait to find out that requests cannot be processed.

By changing the Connection object's **ConnectionTimeout** property, you can reduce the time it takes for a connection to time out. Not all providers support this property, but it can dramatically increase the perceived responsiveness of those that do. The default for the **ConnectionTimeout** property is 30 seconds.

The following script sets the **ConnectionTimeout** property to wait 20 seconds before canceling the connection attempt:

```
Set cn = Server.CreateObject("ADODB.Connection")
cn.ConnectionTimeout = 20
cn.Open "FILEDSN=MyDatabase.dsn"
```

Tip 3: Close Your Connections

Close your connections as soon as you are finished with them.

By proactively closing connections when they are no longer needed, you reduce demand on the database server and make resources available to other users. Connections are closed when they go out of scope, but you can use the **Close** method to close a connection at any time.

The following code creates a **Command** object, implicitly creates a new connection, and calls a stored procedure.

```
<%
   Set cmd = Server.CreateObject("ADODB.Command")
   cmd.ActiveConnection = "dsn=sqlsrvr"
   cmd.CommandText = "{call mysproc}"
   cmd.Execute
%>
```

The **Command** object opens a connection automatically and releases it when the **Command** object goes out of scope at the end of the page. To release it before that time, you can either set the **Command** object reference to **Nothing**, as in:

```
Set cmd = Nothing
```

or you could close the connection by closing the active connection:

```
cmd.ActiveConnection.Close
```

Remember that storing **Connection** objects in the ASP **Session** object is not a good idea, because active connections cause database resources to remain open and would defeat the purpose of the ODBC connection pool.

Tip 4: Share Active Connections

Don't use more database connections than you need. Share connections wherever possible.

Whenever you specify a connection string, rather than a connection variable, you are requesting a new connection to the server. Instead of a connection string, create a single connection variable, and use it with the **ActiveConnection** property of the **Command** and **Recordset** objects.

If your application needs to perform more than one database operation, this approach is a good choice, because it creates the **Connection** separately from other ADO objects. The object can be reused as necessary and closed when all operations have been performed. Simply specify your existing **Connection** object when opening **Recordset** and **Command** objects. For example, the following script stores a reference to an open database connection in a global variable, and uses it to perform two separate queries against the same database:

```
<%@ LANGUAGE=VBScript EnableSessionState=False %>
<HTML>
<HEAD>
<TITLE>Products and Customers</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF>
<H2>Products</H2>
<%
   Set cn = Server.CreateObject("ADODB.Connection")
   cn.ConnectionTimeout = 20
   cn.Open "DSN=AdvWorks"

   Set rs = Server.CreateObject("ADODB.Recordset")
   rs.CursorType = adOpenForwardOnly
   rs.LockType = adLockReadOnly
   rs.Open "Products", cn, , , adCmdTableDirect
   Do Until rs.EOF
   Response.Write rs("Product Name") & "<BR>"
   rs.MoveNext
   Loop
   rs.Close
%>
<H2>Customers</H2>
<%
   rs.Open "Customers", cn, , , adCmdTableDirect
   Do Until rs.EOF
   Response.Write rs("Company Name") & "<BR>"
   rs.MoveNext
   Loop

   rs.Close
   cn.Close
%>
</BODY>
</HTML>
```

Tip 5: Restrict Connections Across Pages

If your provider doesn't support automatic connection pooling, find a balance between the greatest number of connections on the one hand, and the hidden costs of creating and destroying connections on the other.

Storing **Connection** objects (or any of the database access components, for that matter) in the ASP **Application** object is not recommended. Unless your OLE DB provider supports the Both threading model, you will cause the Web server to serialize all user requests for the application—not a way to improve performance. Access does not support this, though SQL Server does; see tip 1. Even if your OLE DB provider supports free threading, you must be cautious when storing any components in the **Application** object.

If you must hold connections open, it is better to create them as individual users require, and store them in the user's **Session** object. Like the **Application** object, Apartment threaded providers such as Access lock the session to a single thread for all requests. Because sessions must time out before the server resources are finally released, applications that store connections in the **Session** object should also provide a means for terminating the session. This allows other clients to use the connection. For example, you could provide a Log Off button to explicitly end the user's session and release the active connection. (See "Developing Web Applications" in this book.)

The following example opens a connection at the start of the user's session. The connection is automatically closed when the session ends.

```
<OBJECT ID=cnSession RUNAT=Server SCOPE=Session
CLASSID="clsid:00000514-0000-0010-8000-00AA006D2EA4">
</OBJECT>
Sub Session_OnStart
'Open ADO connection to "UsersDB1" database.
cnSession.Open "UsersDB1", "userdblogin","userdbpassword"
End Sub
```

Remember, although you can utilize a single connection across more than one page, doing so holds the connection open and defeats the advantages of connection pooling.

Tip 6: Increase the Size of the Record Cache

Increase connection throughput by requesting multiple records at once.

By default, the **CacheSize** property of the **Recordset** object is set to 1 for all cursor types. By increasing **CacheSize**, you can increase the number of records the provider will retrieve at one time into local memory. For example, if **CacheSize** is 10, the provider will retrieve the first 10 records when first opening the **Recordset** object. As you move through the **Recordset** object, the provider uses the data from the local memory buffer. When you move past the last record in the cache, the provider retrieves the next 10 records from the data source.

```
rs.CacheSize = 10
rs.Open strSQL
```

Recordset caching doesn't work well for binary data, like images and text streams, because only the data from the last record in the cache will be available. When using binary data types, you must retrieve one row at a time. Also, records retrieved from the cache don't reflect changes other users may have made since the records were retrieved. To force an update of all the cached data, use a dynamic cursor and call the **Resync** method.

Recordsets and Cursors

A **Recordset**, in its simplest form, is a collection of rows returned from an ADO database query. In addition to the data, each **Recordset** object includes the database schema (or metadata) and a connection to the database. You can create and access a **Recordset** with just a few lines of script:

```
<%
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open "SELECT * FROM Authors", "DSN=Pubs;UID=sa;PWD="
'Loop through the recordset with MoveNext.
Do Until rs.EOF
'Access record fields here.
rs.MoveNext
Loop
rs.Close
%>
```

All **Recordset** objects include a cursor, which marks the current record. When a **Recordset** is first opened, the cursor is positioned to the first record that matched the query. As you move the cursor forward with **MoveNext**, it will eventually run out of records, causing the **Recordset**'s **EOF** (End Of File) property to change to **True**.

All of the cursor types will let you insert and update records in a **Recordset**, unless you have specified **adLockReadOnly** as the lock type. Different cursor types give you varying degrees of visibility for database actions performed by other users. Table 7.3 lists cursor types based on their level of functionality.

Table 7.3 Recordset Functionality by Cursor Type

Cursor Type	Insert, Update Records	View External Updates/Deletions	View External Inserts
Forward-only, read-only (default)	False	False	False
Static	True	False	False
Keyset	True	True	False
Dynamic	True	True	True

Table 7.4 names and describes the four types of cursors available for ADO **Recordset** objects. There is no single cursor type that you should always use. Your choice of cursor type should depend on the functionality you require. The following sections further describe the differences in these cursor types.

Table 7.4 ADO Cursor Types

Cursor Type	Description
adOpenForwardOnly	Forward-only cursor. You can only scroll forward through records. This is the default cursor type.
adOpenStatic	Static cursor. A static copy of a set of records that you can use to find data or generate reports. Additions, changes, or deletions by other users are not visible. The **Recordset** is fully navigable, forward and backward.
adOpenDynamic	Dynamic cursor. Additions, changes, and deletions by other users are visible, and all types of movement through the **Recordset** are allowed (except for bookmarks if the provider doesn't support them).
adOpenKeyset	Keyset cursor (keysets are values used to access specific records or rows). Like the dynamic cursor type, except that you can't see records that other users add. Deletions and other modifications made by other users are still visible.

Since most applications typically access the **Recordset** sequentially from the first record, the default cursor type is optimized for forward-only traversal. (In the case of SQL Server, it is read-only as well. If you use the **Execute** method of the **Connection** (and **Command**) object, you will always get a forward-only, read-only cursor. This is also the default behavior of **Recordset.Open** if no extra parameters are specified.

When you use the **Recordset.Open** method, however, you have the opportunity to configure your **Recordset** exactly as you'd like it (cursor type, record locking method, number of records to cache, and so on). The following example demonstrates how to create **Recordset** objects with varying degrees of cursor functionality:

```
Function SimpleCursor()
Set cn = Server.CreateObject("ADODB.Connection")
cn.Open "pubs", "sa", ""
'Creates a forward-only, read-only cursor.
Set SimpleCursor = cn.Execute("select * from authors")
End Function
Function MoreFunctionalCursor()
Set cn = Server.CreateObject("ADODB.connection")
cn.Open "pubs", "sa", ""
'Requests dynamic keyset cursor and optimistic
'Record locking (concurrency).
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open "select * from authors", cn, adOpenKeyset, adLockOptimistic
Set MoreFunctionalCursor = rs
End Function
```

Forward-Only Cursors

The first type of cursor is optimized for the most common type of database access—sequential, forward traversal of records. This is the default cursor type for **Recordset** objects created when you use the **Execute** method of the **Command** and **Connection** objects. Although you can still add, delete, and update records with a forward-only (but not read-only) cursor (depending on **LockType**), you cannot set a bookmark in the current row to return to the record later. If your cursor type supports bookmarks, you could save a bookmark to the current row if you need to come back later. Also, you cannot use the **Resync** method to view changes made externally to the **Recordset**.

Some providers (like SQL Server) implement the default forward-only, read-only cursor using a "firehose" mode, meaning that the connection is held open as long as there is data to retrieve. Once the cursor has been opened in this mode, the server does not wait for the client to request the next record and then provide it. Instead, the server sends the records as a continuous stream of data. Firehose cursoring is very fast, since overhead is only incurred as the data is read. Firehose cursors do not allow updates to the data.

Forward-Only Restrictions on Binary Large Objects

The Binary Large Object (BLOB) is a type of data field that contains large blocks of text, or binary data such as graphics, sound, or compiled code. BLOBs can be fetched with a forward-only or "firehose" cursor, but there are some access restrictions on the resulting **Recordset**.

- The **CacheSize** property of the **Recordset** must be set to 1. If you cache more than a single row, only the BLOB on the last row is available.
- All BLOB columns must be to the right of any scalar (non-BLOB) columns fetched in the query and you must access them in a left-to-right order.

These restrictions have nothing to do with the underlying table layout—only with the **Recordset** object. For instance, you could have a BLOB column in the middle of your table definition (with scalar columns to the left and right), but when you execute your SELECT statement, you must select the BLOB column after the scalar columns. The storage engine on SQL Server doesn't impose limitations on where a BLOB column lies, but it is significant to the ODBC driver.

Static vs. Dynamic Cursors

When you request a static cursor, you are requesting a snapshot of the data at the time the **Recordset** is created. Once the client has received all the rows, the cursor can scroll through all data without any further interaction with the server. The cursor position can be changed using both *relative positioning* (offsets from the current, top or bottom row) and *absolute positioning* (using the row number). The static cursor's only shortcoming is that changes made to the database while the **Recordset** is open aren't made available to a client using a static cursor.

A dynamic cursor, on the other hand, makes database changes available as they happen. Because the dynamic cursor requires the database provider to reorder data in the **Recordset** as it changes on the server, it is much more expensive than the static cursor in terms of the processing it requires. Also, because the order of the underlying records could change, only relative positioning is available to dynamic cursors. Lastly, dynamic cursors don't support bookmarks. If you need bookmark support, use a keyset or static cursor.

Keyset Cursors

A keyset cursor combines the functionality of the static and dynamic cursor types. It can view all database updates made by other users, and it can search using both absolute and relative positioning. A *keyset* is a set of values used to access specific rows or records in a database. The "key" is actually based on the database indexes used by the current table. When the database needs to update the cursor's value, it does so based on the row's key. Therefore, a "keyset recordset" must include enough columns to ensure unique records in a **Recordset** that's being updated.

Under normal circumstances, when you request records from a data source, you get all the data at once. However, when a keyset cursor is used, only the keys are returned, giving a performance boost. The data for the keys is retrieved when you scroll through the recordset. Except for the first query, which returns both keys and data, the next block of data is fetched only when it is needed. This way the cursor only has to manage small keys rather than large records. Therefore, keyset cursors are more suited for use with a **Recordset** in situations where not all the data will be needed at once.

Cursor Concurrency

The server-side **Recordset** supports four types of record locking (also called *concurrency*), as shown in Table 7.5.

Table 7.5 ADO Cursor Lock Types

LockType	Meaning
adLockReadOnly	The database doesn't lock records, since you are only accessing them in read-only state. This is the default concurrency.
adLockPessimistic	The database locks the records being changed as soon as editing begins. The records are unlocked when all changes are complete. No two users can access the same records at the same time.
adLockOptimistic	The database locks the records being changed only when the changes are committed. Two users can access the same record at the same time, and the database must be able to reconcile (or simply reject) conflicts.
adLockBatchOptimistic	This mode is required for batch updates using client cursors, and is similar to optimistic concurrency.

Note If you fetch more than a single record into the record cache, you should use optimistic concurrency. Doing so allows the server to forgo locks on the database until they are needed, thus freeing resources. However, if there is high contention for the resource, pessimistic concurrency may be preferred. It is easier to reject a request to access a database and have the server try again than it is to reconcile data that is rapidly becoming out-of-date in a record cache.

Cursor Location

When you use RDS to open and populate a disconnected **Recordset**, you're relying on the client-side cursor engine. Although you would normally use a server-side cursor in applications hosted by the middle tier, it is possible to create and use client-side cursors on the server as well.

Client-Side Cursors

In order to use a client-side cursor, you must change the **CursorLocation** property before you open a **Recordset**, as in the following:

```
rs.CursorLocation = adUseClient
```

The client-side cursor engine creates a disconnected RDS **Recordset**, such as you would find on the client tier. Of course, this also requires the database to return all the records that satisfy the query when the **Recordset** is opened. For this reason, no matter what value you select for **CursorType**, you will really only have a static snapshot of the data when using client-side cursors.

> **Note** Because updates to a disconnected **Recordset** aren't immediately communicated to the server, you won't be notified of **Recordset** concurrency errors immediately, as you are with server-side cursors.

Two additional **CursorLocation** values are available: **adUseClientBatch** (for use with **adLockBatchOptimistic** concurrency setting) and **adUseNone**. The **adUseNone** value instructs ADO to forgo its normal *rowset helper* support, which provides extra cursor functionality beyond that which the OLE DB provider may support. When you specify **adUseNone**, you are effectively telling ADO not to include this additional service provider.

Client-side cursors can be used to extend the functionality of the existing driver-supplied OLE DB cursor support. It is possible that the client-side cursor engine can support functionality that is unavailable from the DBMS driver itself, as is the case with the Oracle driver. The RDS client-side cursor may also support features that your server-side cursors do not. For example, you can only use absolute page positioning against a data source by using a client-side cursor.

Server-Side Cursors

Although client-side cursors have their advantages, most applications should be written using server-side cursors for the following reasons:

- **Memory Usage** When using server-side cursors, the client does not need to cache large amounts of data or maintain information about the cursor position; the server provides that functionality.
- **Performance** If you only intend to access a small fraction of the data in a **Recordset** (typical of many browsing applications), server-side cursors boost your performance. This is because only the required data (and not the entire result set) is loaded into memory.
- **Additional Cursor Types** Client cursors are either forward-only with read-only concurrency, or static with read-only and optimistic concurrency. Server-side cursors support the full range of concurrency values and a variety of different cursor types. For instance, keyset and dynamic cursor types are available only if you use server-side cursors.

Managing Records in a Recordset

No discussion of the ADO **Recordset** is complete without talking about how a developer can use it to accomplish the task at hand. In other words, how do you best use a **Recordset** object to add new records, delete and update existing ones, and work through the results of queries? Although this section describes a few techniques to help you through the more troublesome aspects of ADO **Recordset** management, it is not intended as a complete tutorial of ADO development. For more information, see the IIS 5.0 online product documentation.

Keeping Track of New Records

When you add a new record, several factors determine where in the **Recordset** it will be inserted. If you need to move the cursor after you've added a new record, keeping track of where the record was inserted can be a little tricky.

If your cursor type supports bookmarks, you could save a bookmark to the current row and come back to the record later. However, if you insert records into a table without an index, you will not be able to get back to the newly inserted row until after you use the **Requery** method to request a new **Recordset**.

Identity fields, which are updated automatically when you insert a record, can be useful for tracking entities in your database. However, you have no control over the value in this field, and from ADO you have no completely foolproof way of determining what the value will be before it is assigned. Since the value is assigned by the database system itself, you must use a dynamic or keyset cursor, and call the **Resync** method after you Update the new inserted row. The following example demonstrates this technique by adding a new record from data posted using an HTML form, and by displaying the new row's identity value:

```
<%
   'Open a static cursor on the Survey table, and add record.
   Set rs = Server.CreateObject("ADODB.Recordset")
   rs.Open "Survey", Application("ConnectionString"), _
   adOpenKeyset, adLockOptimistic, adCmdTableDirect
   rs.AddNew
   'Add form fields to Recordset, using field names as columns.
   For Each Item In Request.Form
   strItem = Trim(Request.Form(Item))
   If strItem <> "" Then
   rs(Item) = Server.HTMLEncode(strItem)
   End If
   Next
   'Set time of update, and update new record with form values.
   rs("Date") = Now()
   rs.Update

   'Force an update of the Recordset with the identity information.
   rs.Resync
   Response.Write "You are response number: " & rs("ID") & "<BR>"
   rs.Close
%>
```

Important Although this technique provides reasonable performance for smaller tables with few records, performance slows down considerably with larger tables. If you are using SQL Server, you should strongly consider using stored procedures to add new records. For more information, see "Stored Procedures" later in this chapter.

Avoiding Query Time-outs

It's not uncommon to find database tables of 10 million to 50 million records. On these systems, even stored procedures can sometimes take longer than a few minutes to run. A user running a report knows that the query may take several minutes and this is acceptable. However, a Web page doesn't normally wait that long before it will time out.

To solve this problem, you first need to set the **CommandTimeout** property of the **Connection** object. This property indicates how long to wait for a query to execute and applies to the **Connection.Execute** and **Recordset.Open** methods when the **Source** property is not a **Command** object. If the **Recordset.Open** call is using a genuine **Command** object, you'll need to set the **CommandTimeout** property of the **Command** object instead. This property establishes the length of time the application should wait for a **Command** object to execute.

You will also need to increase the values of the **ScriptTimeout** property of the ASP **Server** object. If this property isn't set, it doesn't matter how long the ADO query takes; the script will have stopped executing in the meantime. Both the **CommandTimeout** and **ScriptTimeout** properties accept values indicating the number of seconds to wait before canceling the operation.

Purging Deleted Records

After you delete a record, the **RecordCount** property still includes the deleted record.

If you refresh (or requery) the **Recordset**, it will not include the deleted record and **RecordCount** will be accurate. If you set **CacheSize** to 1 (so that you are caching only a single record at a time), **RecordCount** will accurately exclude deleted rows. You should avoid increasing **CacheSize** with a keyset cursor type in circumstances when you are expecting to be able to use **RecordCount** after you delete records.

References to Field Values

One of the more surprising consequences of using a loosely typed VBScript variable is that you sometimes don't know exactly what you are referencing. In the following ASP example, the session variable *strAuthor* apparently loses its value once the **Recordset** is closed.

```
Set Session("strAuthor") = rstPublishers.Fields("Author")
rstPublishers.Close
```

And then later in the script:

```
Response.Write Session("strAuthor") 'Where'd it go?
```

The problem is that *strAuthor* is a Variant data type containing a reference to a **Field** object, not a string, as it appears to be. When the **Recordset** is closed, the **Field** object is no longer valid, and the variable appears empty. You can avoid the problem by qualifying the code with the **Value** property:

```
Session("strAuthor") = rstPublishers.Fields("Author").Value.
```

Rather than perform a lookup from the **Recordset** collection each time you need a field (which is what happens when you use the column name), you can use a reference to a **Field** object to keep track of the current value of a field.

```
<%
   Set rs = Server.CreateObject("ADODB.Recordset")
   rs.Open "Products", "DSN=AdvWorks",_
   adOpenForwardOnly, adLockReadOnly, adCmdTableDirect
   'Select fields now, using single lookups.
   Set fldProdId   = rs("ProductID")
   Set fldProdName = rs("ProductName")
   Set fldProdDesc = rs("ProductDescription")

   'Loop through records using field references.
   Do Until rs.EOF
   Response.Write fldProdId & ": <b>" & fldProdName &_
   "</b>: " & fldProdDesc & "<BR>"
   rs.MoveNext
   Loop
   rs.Close
%>
```

With this technique, it is possible to perform a query even without specifying a SQL command.

VBScript Example: Filling a List Box

When using a server-side script to fill a list box for use on the client's HTML page, keep in mind that the list box doesn't exist yet when your server-side code is executed. For this reason, you need to generate client-side code using server-side scripting. You can use either the standard HTML SELECT tag (which creates a drop-down list box), or a custom ActiveX Listbox control. Using the SELECT tag, you simply use server-side HTML to fill in the VALUEs. The following example demonstrates how you could fill in a SELECT control from an ADO **Recordset**:

```
<SELECT>
  <% Do While NOT rs.EOF %>
  <OPTION VALUE="<%= rs("Name") %>"><%= rs("Name") %></OPTION>
  <% rs.MoveNext
  Loop %>
</SELECT>
```

When you are working with client-side ActiveX objects like the Listbox control, it's a good idea to use server-side scripting to programmatically pass values to **ListBox.AddItem** calls during the **Window_OnLoad** event. Here is an example (the groups of double quotation marks are used to keep the **Name** data together as a string):

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Window_OnLoad()
<% Do While NOT rs.EOF
Response.Write "ListBox.AddItem """" & rs("Name") & """" & vbCr
rs.MoveNext
Loop %>
End Sub
-  >
</SCRIPT>
```

PerlScript Example: Filling a Table

Filling a table is nearly identical to filling a SELECT tag. Here's an example that uses PerlScript to fill the rows of a table:

```
<%@ LANGUAGE=PerlScript %>
<% #--- Open the connection and query.
$Conn = $Server->CreateObject("ADODB.Connection");
$Conn->Open( "AdvWorks" );
$RS = $Conn->Execute( "SELECT * FROM Products" );
%>

<TABLE BORDER=1>
<TR>
<% #--- Create a row of column headings.
$count = $RS->Fields->Count;
for ( $i = 0; $i < $count; $i++ ) {
%><TH><%= $RS->Fields($i)->Name %></TH><%
   };
%>
</TR>
<% #--- Fill in the rows of the table with data.
while ( ! $RS->EOF ) {
%><TR><%
for ( $i = 0; $i < $count; $i++ ) {
%><TD VALIGN=TOP>
<%= $RS->Fields($i)->Value %></TD><%
      };
%></TR><%
$RS->MoveNext;
   };

#--- Close connection.
$RS->Close;
$Conn->Close;
%>
</TABLE>
```

Limiting the Number of Records

If you know you'll require only the first few records, it makes sense to limit the number of records retrieved from the database. You can do this with the **MaxRecords** property of the **Recordset** object. The default setting for this property is zero, which means that the provider normally returns all requested records. Setting it to some other value will limit the size of the rowset returned from the query. The effect is the same as if you had used the Microsoft® SQL Rowcount directive as part of your query.

The **MaxRecords** property is often used together with a SQL "ORDER BY" clause to produce a "Top Ten" list based on some attribute of the data. The next example returns a **Recordset** containing the 10 most expensive publications in the Titles table.

```
<%
   'NOTE: DSN-less connection.
   strCnn = "driver={SQL Server};server=(local);" &_
   "uid=sa;pwd=;database=pubs"
   Set rs = Server.CreateObject("ADODB.Recordset")
   rs.MaxRecords = 10
   rs.Open "SELECT Title, Price FROM Titles ORDER BY Price DESC",_
   strCnn, adOpenForwardOnly, adLockReadOnly, adCmdText
%>
```

Note You must set the **MaxRecords** property before the **Recordset** is opened. After setting **MaxPrecords**, the property is read-only.

Visual Basic Example: Paging through a Recordset

After browsing the top ten results, users may want to view the remaining records. In this case, you don't necessarily want to limit the records. You just want to stop after the first page is filled, and wait for the signal to continue. It's easy enough to stop after a certain number of records. But what's the best way to pick up where you left off?

The **Recordset** object exposes three properties to assist you in your task: **PageSize**, **PageCount**, and **AbsolutePage**. Once you have set the page size, your database provider can calculate how many pages of data it will return to you, and let you jump to locations at the beginning of each page. (If you set the **CacheSize** equal to **PageSize,** only the records you display will be cached in the **RecordSet**.) Note that you must use a client-side cursor to enable absolute positioning in ODBC data sources.

The following Visual Basic example uses the **AbsolutePage**, **PageCount**, and **PageSize** properties to display names and hire dates from the Employee table (of the Pubs database), listing five records at a time in a message box:

```
Public Sub AbsolutePageExample()
Dim rstEmployees As New ADODB.Recordset
Dim strCnn As String
Dim strMessage As String
Dim intPage As Integer
Dim intPageCount As Integer
Dim intRecord As Integer

strCnn = "driver={SQL Server};server=(local);" & _
"uid=sa;pwd=;database=pubs"
'Use client cursor to enable AbsolutePage property.
rstEmployees.CursorLocation = adUseClient
rstEmployees.CacheSize = 5
rstEmployees.Open "employee", strCnn
rstEmployees.MoveFirst

'Display names and hire dates, five records at a time.
rstEmployees.PageSize = 5
intPageCount = rstEmployees.PageCount
For intPage = 1 To intPageCount
rstEmployees.AbsolutePage = intPage
strMessage = ""
For intRecord = 1 To rstEmployees.PageSize
strMessage = strMessage & _
rstEmployees!fname & " " & _
rstEmployees!lname & " " & _
rstEmployees!hire_date & vbCr
rstEmployees.MoveNext
If rstEmployees.EOF Then Exit For
Next intRecord
MsgBox strMessage
Next intPage
rstEmployees.Close

End Sub
```

Recordset paging gets a little more complex when you are using an ASP page. Because you can return only a single page of data at a time, you'll have to decide what to do with the open **Recordset** object while waiting for the next client request. You have two options—throw away the results and keep just the current page number, or stow the **Recordset** object in the user's **Session** object for later retrieval. Which approach you choose will be based on the needs of your user and your application. You must then decide whether you want to save time by caching the results, or if you would rather release the memory for another application to use.

> **Note** An example of **Recordset** paging is included on the *Microsoft® Windows® 2000 Server Resource Kit* companion CD, as part of the Feedback application.

Retrieving Image Data

Images (as with long bodies of text) are stored in a database as BLOB fields. Because of the added overhead for the DBMS, retrieving images from a database is slower than referencing an image URL on disk, so whether or not you store images in a database depends on the requirements of your application. You can retrieve image data with ADO by using the **GetChunk** method of the **Field** object. This method requires that you specify the number of bytes or characters that you wish to retrieve.

It takes two files working together to retrieve multiple images on the same page. The main file, which contains HTML formatting and IMAGE tags, requires the use of a separate ASP file to perform the actual image query. The secondary ASP file, Image.asp, retrieves the requested image, and returns it as a binary object in the HTTP response using a MIME content type of "image/gif." The following is the source for Image.asp:

```
<%@ LANGUAGE=JScript EnableSessionState=False %>
<%
   var ID, rs, fld, cBytes
   ID = Request.QueryString("ID");
   if (ID + "" != "undefined") {
   rs = Server.CreateObject("ADODB.Recordset");
   rs.Filter = "ImageID=" + ID;  //--- Search criteria
   rs.Open ("Images", Application("ConnectionString"),
   adOpenForwardOnly, adLockReadOnly, adCmdTableDirect);
   if (!rs.EOF) {
            fld = rs("ImageData");     //Get field reference.
            cBytes = fld.ActualSize;   //Determine size.

   //Return raw binary image data as "image/jpeg" MIME type.
   Response.ContentType = "image/jpeg";
   Response.BinaryWrite(fld.GetChunk(cBytes));
         }
   else {
   Response.Write("Image '" + ID + "' not found.");
         }

   rs.Close();
      }
   else {
   Response.Write("No ID");
      }
%>
```

In the main file, images stored in the database can now be retrieved using the image ID as a URL parameter to Image.asp, like this:

```
<IMG SRC="./image.asp?ID=<%=ImageID%>">
```

When you use a firehose cursor, there isn't a good way to discover the size of a BLOB field before you read it. If you must use a firehose cursor, consider maintaining a separate column in the database table that stores the image's size in bytes. Otherwise, you could call the **GetChunk** method repeatedly until it returns nothing. Alternatively, if you think that the image will fit into available memory, simply use the **Value** property to retrieve the data all at once.

Stored Procedures

If you find yourself performing complex data manipulation, consider organizing database dependencies and rules into stored procedures.

Stored procedures are precompiled queries stored on the database server. They can simplify development and significantly speed up complex queries. When a stored procedure is called, it controls which operations are performed and which database fields are accessed. A single stored procedure can even execute multiple queries.

Stored procedures have explicitly defined parameters, each with a specific data type, direction, and size. Before calling a stored procedure, the **Parameter** collection of the **Command** object must be prepared to precisely match the number and type of parameters defined for the procedure on the server. Although you can request the complete **Parameter** collection by calling the **Refresh** method, building the collection parameter by parameter is preferred. Calling this method results in faster execution and avoids a network round-trip to the server. (Also, some providers do not support populating the **Parameter** collection with the **Refresh** method.) The code, however, ends up looking a bit more complex, as shown here:

```
<%
   Set cm = Server.CreateObject("ADODB.Command")
   cm.CommandText = "AddCustomer"
   cm.CommandType = adCmdStoredProc
   Set p = cm.Parameters
   p.Append cm.CreateParameter("@Name", adChar, adParamInput, 40)
   p.Append cm.CreateParameter("@Address", adChar, adParamInput, 80)
   p.Append cm.CreateParameter("@City", adChar, adParamInput, 20)
   p.Append cm.CreateParameter("@State", adChar, adParamInput, 2)
   p.Append cm.CreateParameter("@Zip", adChar, adParamInput, 11)
   cm("@Name") = Trim(Request.Form("Name"))
   cm("@Address") = Trim(Request.Form("Address"))
   cm("@City") = Trim(Request.Form("City"))
   cm("@State") = Trim(Request.Form("State"))
   cm("@Zip") = Trim(Request.Form("Zip"))
   cm.Execute
%>
```

Returning Values from Stored Procedures

The following lines of OSQL code (a utility that executes Transact-SQL commands) define two stored procedures—one that returns the value 10 directly, and one that returns a value by reference in its output parameter:

```
Create procedure sp_retvalparam as return 10
Go
Create procedure sp_inoutparam(@in char(200),@out char(200) out) as select @out = @in
Go
```

The following examples use two different but equally valid means of invoking these stored procedures. Here's the first method:

```
cmd.CommandText = "sp_retvalparam"
cmd.CommandType = adCmdStoredProc
'Quietly retrieve parameter info from server.
cmd(0).Direction = adParamReturnValue
cmd.Execute
Response.Write cmd(0)
```

The collection lookup, `cmd(0).Direction = adParamReturnValue`, causes an implicit **Refresh** to retrieve parameter descriptions automatically, which causes an extra trip to the server. You can avoid this extra trip by creating your own parameter collection with the **CreateParameter** method, as follows:

```
cmd.CommandText = "{ call sp_inoutparam(?,?) }"
cmd.CommandType = adCmdText
'Specify parameter info.
p.Append cmd.CreateParameter("in", adChar, adParamInput, 200, "foo")
p.Append cmd.CreateParameter("out", adChar, adParamOutput, 200)
cmd.Execute
Response.Write cmd(1)
```

Prepared Queries

If a SQL statement will be used multiple times, you can potentially improve the performance of your application with prepared queries. When a SQL statement is prepared, a temporary stored procedure is created and compiled. This procedure is executed when the prepared statement is called, which saves the overhead of parsing the command each time it is used.

The following example demonstrates the use of a prepared query:

```
<%
   Set cmd = Server.CreateObject("ADODB.Command")
   cmd.ActiveConnection = Application("ConnectionString")
   cmd.Prepared = True
   cmd.CommandText = "Select * From Catalogue Where Id=?"
   Set p = cmd.Parameters
   p.Append cmd.CreateParameter("prodId",adInteger,adParamInput)
   cmd("prodId") = Request("Id1")
   Set rs = cmd.Execute

   cmd("prodId") = Request("Id2")
   Set rs = cmd.Execute
%>
```

Usually, prepared queries are dropped when the connection is released. When connection pooling is enabled, however, you risk running out of space in your temporary database if your connections are recycled often enough. If you use prepared queries, you can configure the SQL Server driver to drop queries "as appropriate" when connection pooling is enabled. You can select this option in the **Data Sources (ODBC)** application (in **Administrative Tools**) in Control Panel.

Transaction Processing on the Web

People have been talking about client/server applications and middleware for years. But the truth is that, even with supposedly simple facilities like RPC (remote procedure call) and named pipes, writing server-based applications that readily and efficiently interact with desktop-based applications has remained just out of reach for most developers. Component Services makes it possible to easily write true *n*-tier applications.

Component Services extends the functionality of applications running on Windows 2000 Server and IIS 5.0. The integration of IIS 5.0 and Component Services makes it easy to manage transactions—complex, distributed database updates. In addition to managing transactions, Component Services provides other benefits that are perhaps even more important:

- **Simplified Programming** Component programming in the Component Services environment is as noninvasive as possible. In most cases, fewer than five lines of code are required to make an object transaction-aware. In fact, most COM objects can participate in Component Services transactions without modification.

- **Distributed Applications Framework** In the Component Services run-time environment, distributed applications are the norm, not the exception. Component Services seamlessly integrates the Component Services Executive, server processes, resource managers, and resource dispensers with the Microsoft® Distributed Transaction Coordinator (DTC) to create an environment that can run on the same system or be scattered across multiple systems.
- **Components** Application components can be built in any language that supports the creation of COM objects.
- **Security** Component Services automatically enforces user roles and Windows 2000 security for you.
- **Recovery and Restart** When failures occur, COM+ applications are automatically restarted, and data consistency is maintained.
- **Scalability** Context and thread management, automatic resource recycling, and just-in-time (JIT) activation help COM+ applications perform extremely well in high-usage scenarios.
- **Transactions** COM+ applications inherit the power and reliability of automatic distributed transactions. Any DTC-compliant OLE DB or ODBC data source, such as Message Queuing, can participate in those transactions.

Transactions Explained

A transaction, simply put, is an "all or nothing" sequence of database transformations. No modifications are committed to the database until all steps of the transaction have completed successfully. If any of the actions cannot be completed, the entire transaction is automatically "rolled back," or undone. Transactions are a technique applied to guarantee the "correctness" of database operations.

The properties of transactions are collectively known by the acronym *ACID:*

- **Atomicity** Transactions are either committed or rolled back as a whole. Unless all database transformations are successful, none of them will be committed to the database.
- **Consistency** A transaction never leaves the system in a state that cannot be recovered if the transaction fails.
- **Isolation** Until the transaction has completed successfully, its modifications are not visible to other users.
- **Durability** Committed transactions persist beyond any subsequent software or hardware failures. Transactions that have not committed are rolled back when the system is restarted.

At the most basic level, transactions ensure that data is protected from accidental modifications that would invalidate it. If an event occurs that upsets the intended sequence of changes, all the previous changes can be undone to restore the database to its original form. What happens if the second (or third) update fails? What if the application that is making changes crashes? What if the computer is accidentally turned off? Transactions that are stopped short are guaranteed to have no lasting effect on your data.

A transaction defines a boundary that encapsulates several database interactions and makes them appear as a single atomic interaction. Once a transaction has begun, an application can make changes to multiple records, and the effect of these changes is isolated from the rest of the database and from other users. When the transaction is committed, all the changes appear to happen simultaneously, in such a way as to guarantee that no data is lost or compromised.

Although transactions are important in commerce, transactions aren't just about money. They arbitrate the contention that occurs with demand for any "hard" resource that cannot be created or destroyed.

Extending the Limits of Transactions

Although you can perform transactions using the ADO **Connection** object, the transaction is limited to a single server. In fact, most transaction processing systems allow only one server to participate in a transaction at a time.

But what if, for example, you are transferring money from checking accounts on two separate systems, and the systems use different database management systems? In this case, transaction protection limited to a single connection isn't enough. To manage both database operations as a single transaction, you need a distributed transaction coordinator like Component Services.

With Component Services, applications can easily use transactions that access multiple databases with true two-phase commit. The two-phase commit protocol ensures that transactions that apply to more than one server are completed on all servers or none at all. Two-phase commit is coordinated by the transaction manager and supported by resource managers.

Component Services supports a component-based programming model and run-time environment for developing and deploying enterprise Web server applications. You can configure components to run in the Component Services environment by adding them to a Component Services package. The packaged components run together in the same server process. Packages also define declarative security constructs that define how access to components is controlled at run time. With packages, developers and administrators can configure applications at deployment time to match the topology and security requirements of their target environment.

Transactional ASP

IIS 5.0 makes it possible to easily write scalable, reliable, and transactional Web applications. The features of Component Services are available to all ASP and ISAPI applications. IIS frees developers to focus on what is really important in an application —the business logic.

For all ASP transactions, the "unit of work" is the ASP page. Transactions are limited to a single page; they cannot span multiple pages.

To perform a transaction, the ASP file containing the application script simply needs to include a command at the beginning of the script to declare that a transaction is needed: `<%@ transaction=required %>`. For these pages, a new ASP built-in object, **ObjectContext**, is defined that is used to commit or abort the transaction.

You indicate when the transaction is over by using the **ObjectContext** object, and then, if all is well, you commit the transaction using the **SetComplete** method. If the operation has failed, use **SetAbort**. Unless one of these methods is called explicitly, the transaction will commit automatically when the page has completed processing. Two event handlers are available on transactional pages, **OnTransactionCommit** and **OnTransactionAbort**.

The following example demonstrates each of these transaction elements by randomly aborting a transaction. Although it doesn't do any real database processing, it illustrates how easy it is to add the power of transactions to Web-based applications. Click the browser's **Refresh** button repeatedly to see the event handlers at work.

```
<%@ LANGUAGE=VBScript Transaction=Required EnableSessionState=False %>
<HTML>
  <HEAD>
    <TITLE>Transactional ASP</TITLE>

  <SCRIPT Language=VBScript Runat=Server>
    Sub OnTransactionCommit()
    'Code used when transaction succeeds.
    Response.Write "<FONT color=green>committed</FONT>"
    End Sub
    Sub OnTransactionAbort()
    'Code used when transaction fails.
    Response.Write "<FONT color=red>aborted</FONT>"
    End Sub
  </SCRIPT>
  </HEAD>

  <BODY BGCOLOR=#FFFFFF>
    The transaction was:
    <% 'Randomly abort the transaction.
    Randomize
    If Rnd > 0.5 Then
    ObjectContext.SetAbort
    End If
    %>
  </BODY>
</HTML>
```

Because the transactional event handlers aren't called until the page has completely finished processing, the final messages in the example appear after the closing HTML tag. A better implementation would consist of pure script in ASP pages, whose event handlers redirect to a separate page containing the appropriate HTML response. For a discussion of redirection, see "Developing Web Applications" in this book.

Any components used on the transactional ASP page indirectly affect the outcome of the transaction, even if the components themselves are not transactional.

Business Objects vs. Script in ASP Pages

It is a common mistake to code business logic by exclusively using script in ASP pages. ASP is interpreted at run time, so pure script implementations are much slower than those that use precompiled objects. Also, because the business logic is exposed as script, it can be viewed by anyone with access to the server, making it vulnerable to tampering. Lastly, scripted business logic is not easily and cleanly reusable. As the application grows, so does its complexity. Objects become enmeshed with other objects, the system becomes hard to manage, and the bugs multiply.

For all these reasons, ASP should only be used as the "glue" that holds components together. If you're serious about scalability, you must implement your business logic as components.

Transactional Components

Applications that access databases can be built two ways: Business logic can reside in a particular database, or it can be packaged into components. The advantage of using components is that business rules can be generalized to work with any database, and can be reused in multiple applications.

When business logic is segregated into components that work in multiple scenarios, the components become building blocks for applications. The same components can be used for both network and Web-based applications. Reusing components speeds development time and lowers costs.

Business Logic in Components

A component doesn't have to access a database to be considered a business object. In fact, some business objects simply perform complex calculations and automate common business tasks. This section presents some guidelines for designing effective COM components.

Component Granularity

The number of tasks a component performs determines its granularity.

A *fine-grained* component consumes and releases resources quickly after completing a task. Components such as these isolate individual tasks in well-defined modules, which are easily reused in other packages. For example, you might design a component to facilitate adding and removing customer records in a database. Because its task is simple, the component can be written efficiently and is easy to debug and maintain. It is also more likely to be reused in other applications that maintain customer data.

A *coarse-grained* component performs multiple tasks that are often unrelated to each other. For example, a component called PlaceOrder might not only create a new order, but modify inventory and update customer data, too. Coarse-grained components are somewhat harder to debug, and are less likely to be reused.

Component State

Objects that discard information between uses are considered *stateless*. Business objects usually don't need to maintain state to correctly process new requests; nor do they need to maintain database connections between calls. Stateless objects use fewer system resources, so they can scale better and perform more quickly.

Stateful objects, on the other hand, accumulate information over several method calls. Because a COM+ application cannot commit transactions until **SetComplete** is called, stateful objects effectively prevent the COM+ application from completing its work. Frequent network roundtrips and slower connections extend the lifetime of the object. The extra delay might cause server resources, such as database connections, to be held longer, and thus decrease the system resources available for other clients. A decision to hold state information in an object should be considered carefully.

As the number of concurrent transactions increases, stateless objects begin to outperform stateful ones significantly. In other words, stateless components scale better. Despite the limitations of stateful objects, it sometimes makes sense to maintain some state information, especially if it is the result of a complex or time-consuming query. Stateful objects might be used if reconstructing the object state is potentially more costly than the resources held open while it remains active.

> **Note** Objects that need to retain state across multiple calls from a client can protect themselves from having their work committed prematurely by the client. By calling the **DisableCommit** method of the **ObjectContext** object before returning control to the client, the object can guarantee that its transaction cannot be committed until the object has called **EnableCommit**.

Participating in Transactions

You can create a transactional component that takes advantage of the benefits of a COM+ application, with only a few extra lines of code.

To create a transactional component

1. Call **GetObjectContext** to get a reference to the **ObjectContext** object, which enables your component to "vote" on the success or failure of the transaction in progress.
2. Create other components by using the **ObjectContext** object's **CreateInstance** method. When a base client instantiates an object by using the **CreateInstance** method, the new object and its descendants will participate in the transaction, unless the new object's transaction attribute is set to **Requires a new transaction** or **Does not support transactions**.
3. Call either **SetComplete** when the object has completed successfully, or **SetAbort** to cancel the transaction.

Just-In-Time Activation

This discussion about component state really only makes sense within the scope of a transaction. But what happens to the object once the transaction has been completed? Component Services deactivates it. Whenever an application calls **SetComplete**, whether from an ASP page or from within a COM component, the system recycles all the objects involved in the transaction—even stateful ones. In the process of deactivation, the object's member variables are reinitialized to their initial values.

This process is part of *just-in-time activation*, which allows the Component Services run-time environment to free up object resources, including any database connections it holds, without requiring the application to release its references to component objects. Components are activated only as needed for executing requests from clients, allowing otherwise idle server resources to be used more productively. Just-In-Time activation allows your application to conserve system resources as it scales up to multiple users.

This is yet another reason to be careful about maintaining state in objects. Clients of a stateful COM component object must be aware of how it uses **SetComplete** to ensure that any state the object maintains won't be needed after the object undergoes just-in-time activation.

The following Visual Basic code template demonstrates how to incorporate these elements into a component:

```
Sub DoMTSTransaction()
Dim objCtx As ObjectContext
Dim objNew As NewObject

On Error Goto ErrHandler
Set objCtx = GetObjectContext()
Set objNew = objCtx.CreateInstance("MyObject.NewObject")

   '--------------------------------
'--- More component logic here ---
   '--------------------------------

objCtx.SetComplete
Exit Sub

ErrHandler:
objCtx.SetAbort
End Sub
```

Be sure not to use **CreateObject** or **GetObject** when creating objects in a COM component. Role-based security in Component Services works only if you use **Server.CreateObject** from ASP or **ObjectContext.CreateInstance** from within your COM component. When you use **CreateObject** or **GetObject**, the component identity is inherited from the application process. Conversely, when you use **Server.CreateObject**, the identity is that of the impersonated user. In order for COM role-based security to work properly, the correct caller identity must be determined.

For more information about COM security and roles in Component Services, see "Security" in this book. For more examples of how to create transactional components, refer to the Exploration Air sample site included on the Resource Kit companion CD.

Using Database Access Interfaces with Component Services

Because the ODBC Driver Manager is a Component Services resource dispenser, data accessed via ODBC is automatically protected by your object's transaction. For object transactions, an ODBC-compliant database must support the following features:

- The database's ODBC driver must be thread-safe. Component Services must be able to connect to the database, utilize the connection, and disconnect by using different threads. (The Access driver cannot participate in Component Services transactions, because it is not completely thread-safe.)
- If ODBC is used from within a transactional component, the ODBC driver must also support the **OdbcSqlAttrEnlistInDtc** connection attribute. It's through the use of this attribute that the ODBC Driver Manager can allow the ODBC driver to enlist a connection on a transaction. If you are using a database without a resource dispenser that can recognize Component Services transactions, contact your database vendor to obtain the required support.

Table 7.6 summarizes database requirements for full COM+ application support.

Table 7.6 ODBC Driver Requirements for Component Services Compliance

Requirements	Description
Support for the OLE transactions specification, or support for XA protocol	Enables direct interaction with the Distributed Transaction Coordinator (DTC) by using the XA Mapper.
ODBC driver	Platform requirement for COM+ application server.
Support for ODBC version 3.0's **SqlAttrEnlistInDtc** attribute	COM+ applications use this call to pass the transaction identifier to the ODBC driver. The ODBC driver then passes the transaction identifier to the database engine.
Fully thread-safe ODBC driver	ODBC driver must be able to handle concurrent calls from any thread at any time.

Because ADO is an indirect consumer of ODBC data sources, it makes a good candidate for data access from COM components. In fact, there is very little semantic difference between using ADO inside a COM component, and using it from a Web page.

Distribution and Scaling Issues

Hosting a large volume of business transactions from clients around the world introduces a demanding set of programming and administrative requirements. Distributed applications very often rely on diverse resources beyond the application's scope of control. Web applications must be prepared to host a large volume of connections without suffering a significant loss in performance.

The key factors that influence a component's ability to service a large volume of users include:

- Resources allocated per user (memory, data connections).
- Amount of information retrieved from and sent to the browser.
- Location of processing (client, server, or both).
- Impact of component distribution topology on response time.
- Time to process common requests.

Components should be located as close as possible to the data source. If you are building a distributed application with a number of COM component packages running on local and remote servers, try to group your components according to the location of your data. For example, the Accounting server should host both the Accounting Component Services package and the Accounting database.

Pool your resources by server process. Note that Component Services runs each hosted package in a separate server process. The fewer pools that are running on a server, the more efficiently resources are pooled. Try to group components so that they share "expensive" resources, such as connections to a specific database. If you reuse these resources within your package, you will improve the performance and scaling of your application. For example, if a database lookup and a database update component are running in a customer maintenance application, package those components together so that they can share database connections.

Introducing Message Queuing

For components that are not directly connected, diminished network throughput can be a severe impairment to the scalability of an application. This section discusses the use of Message Queuing to increase the reliability of transactions that use disconnected components.

Message Queuing is a fast store-and-forward service that enables applications to communicate across heterogeneous networks and systems. Applications send messages to Message Queuing, and Message Queuing ensures that the messages eventually reach their destination. Message Queuing provides guaranteed message delivery, efficient routing, security, and priority-based messaging.

Most importantly, Message Queuing also supports Component Services transactions. Because it is a resource manager under the control of the DTC, you can use Message Queuing to safely implement transaction-compliant applications that ensure that message operations either succeed or fail in conjunction with other transactions. A transactional application can send a message through Message Queuing and update a database as part of the same transaction. The transaction coordinator ensures that both actions succeed or fail together.

For instance, a single online purchase might consist of a credit card validation, an adjustment to inventory on hand, and a notification to an external supplier. Of course, the external supplier's system may be temporarily offline, which prevents the system from notifying the external supplier. In the context of a transaction, failures such as this dictate that the entire process be aborted, whether or not the inventory was available and in spite of a successful credit card debit.

A better solution is to use Message Queuing, which guarantees that the vendor will be notified eventually. In this way, Message Queuing enables systems to respond more flexibly to factors that can cause transaction failures; as a result, this effectively eliminates errors caused by server outages and long pauses caused by network latency, making Message Queuing ideal for "time-independent" transactions.

Time-Independent Transaction Processing with Message Queuing

Complex transactions may have several smaller subtransactions, each with differing levels of importance. Determining which operations are required to complete the transaction, and which may be eligible for deferral, is a crucial first step in designing a robust system with Message Queuing. You should only consider using messages for operations that are not essential to the overall stability of the system.

Message Queuing operations always commit, and return, control to your application quickly. When used to process part of your transaction, Message Queuing saves a "message" to an on-disk queue, instead of performing a database operation. The message should contain enough information to describe the operation that would have been performed on the remote system, had it been connected.

Some time later (or at a time you choose), Message Queuing establishes a connection with Message Queuing that is running on the remote system and then transmits the message. As soon as the remote system has successfully received it, the message is dequeued and processed as part of a new database transaction. If all goes well, a confirmation message is sent back to the Message Queuing system that initiated the transaction. This message path is shown in Figure 7.5:

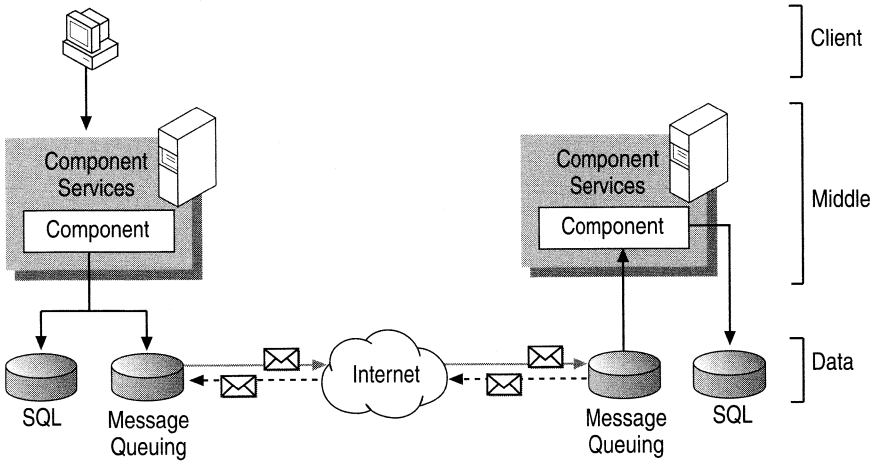


Figure 7.5 Diagram of a Message Queuing Transaction

Since Message Queuing messages are system-specific, you are responsible for collecting the information your system will need to complete the transaction. Message Queuing merely guarantees that your message will be delivered to the appropriate queue on the remote system. The following example demonstrates how you might add support for Message Queuing to a COM component within a system that processes orders for a vendor's products:

```
Sub OrderProduct(
OrderID As Int,
ProductID As Int,
Quantity As Int)
Dim objCtx   As ObjectContext
Dim query    As New MSMQQuery
Dim msg      As New MSMQMessage
Dim infoSend As New MSMQQueueInfo
Dim infoResp As MSMQQueueInfo
Dim queue    As MSMQQueue

On Error Goto ErrHandler
Set objCtx = GetObjectContext()

'Open destination queue.
infoSend.PathName = "Contractor\ProductOrderQueue"
Set queue = infoSend.Open(MQ_SEND_ACCESS,MQ_DENY_NONE)
'Construct application specific message.
msg.Label = "Product Order"
msg.Body  = Str(OrderID) & ";" & Str(ProductID) & ";" & Str(Quantity)
msg.PrivLevel = MQMSG_PRIV_LEVEL_BODY

'Lookup response queue.
Set infoResp = query.LookupQueue(Label:="ContractorResponse")
infoResp.Reset
'Set application specific response queue.
Set msg.ResponseQueueInfo = infoResp.Next

'Send message to remote queue.
msg.Send queue, MQ_MTS_TRANSACTION

queue.Close
objCtx.SetComplete
Exit Sub

ErrHandler:
objCtx.SetAbort
End Sub
```

A Message Queuing transaction merely sends a message, so the application must assume that the work can be performed. In this case, no effort has been made to determine if the contractor can supply the requested number of products. This inability to determine, in advance, if the transaction will be honored is a problem known as "deferred integrity." A good example of deferred integrity in the real world might be that of a bank handling the case of insufficient funds for a check that has already been cashed.

Once the decision has been made to allow portions of a transaction to take place asynchronously, a business policy must be implemented to determine what happens if the transaction cannot be satisfied. In certain cases, transaction failures may require an application-level rollback of prior-committed transactions. Or, they may require correspondence with the customer in order to decide the best course of action.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server. It also provides resources for Web server data access and transactions.

Web Links

http://www.apexsc.com/
: The definitive source for information about a variety of data-bound grid controls. As a service to DBGrid users everywhere, Apex Software Corporation provides free online help, samples, and downloads.

http://www.microsoft.com/data/
: Provides a central location for information about Universal Data Access and the technologies that make it possible. Here you will find information and the latest news on ADO, OLE DB, ODBC, and much more.

http://www.microsoft.com/com/
: The Microsoft® Component Object Model (COM) technologies Web site encompasses information about COM-based technologies such as Microsoft® Distributed Component Object Model (DCOM), Component Services, ActiveX controls, and more.

http://msdn.microsoft.com
: The Microsoft Web site for application developers and Web site builders provides information about developing Web applications. Includes information about access to data and transactions.

Books

Teach Yourself Active Web Database Programming in 21 Days by Fleet, Warren, Chen, and Stojanovic, 1997, Indianapolis: Sams.net Publishing.

A step-by-step tutorial of ADO, and data-centric business object development fundamentals.

ADO 2.0 Programmer's Reference by Sussman and Homer, 1998, Chicago: Wrox Press Ltd.

Provides a concise, comprehensive guide to the way ADO 2.0 can be used in all kinds of applications. Demonstrates ADO using VB, C++, J++, and scripting languages with ASP.

Teach Yourself Database Programming with Visual C++ in 21 Days by L. Robinson, 1998, Indianapolis: Sams.net Publishing.

Covers Visual C++ programming for ADO, RDS, OLE DB, ODBC, Component Services, COM, as well as DAO and the MFC ODBC classes. You will learn how to choose the best database model/technology for your application. You will also learn about database design, ways to leverage a database server, how to create C++ component software for database applications, and multitier application development.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 8

Administering an ISP Installation

This chapter is for Internet service providers (ISPs), Enterprise Service Providers, and Application Service Providers, of all sizes and configurations. Whether your ISP installation contains only one server, or encompasses clusters of servers in various locations, the information in this chapter will help you to administer your installation in an efficient and cost-effective manner.

Organized by task, the material in this chapter shows you how to leverage the power of Internet Information Services (IIS) 5.0 in order to streamline administration. For example, in the "Configuring IIS 5.0" section, you'll learn how to create a company Web site and a personal Web site, and how to restrict content to certain groups of users. In the "Managing Your Installation" section, you'll see how to *automate* tasks, including Web site creation and changing site permissions. You'll also see how to leverage the replication and clustering features of Microsoft® Windows® 2000 Server in order to guarantee the reliability of your installation. In addition, you'll find information about how to take advantage of other Microsoft products that work in conjunction with IIS 5.0, adding to its power. For example, integrating IIS 5.0 with products such as Microsoft® FrontPage® Server Extensions can make it easier to automate, customize, and administer your installation. And in the "Customizing Your Installation" section, you'll learn how to save money on hardware by configuring multiple sites on one computer, with one Internet Protocol (IP) address.

The last two sections of this chapter show you how to put everything together. The first of these, "Building a Web Cluster," tells you how to set up a three-tier Web cluster. The last section, "Adapting IIS 5.0 to a Sample Installation," describes a real-world ISP installation and tells you how to use IIS 5.0 for various configurations, depending on your customers' needs.

In This Chapter

Configuring IIS 5.0

Managing Your Installation

Customizing Your Installation

Building a Web Cluster

Adapting IIS 5.0 to a Sample Installation

Additional Resources

Configuring IIS 5.0

On an ISP installation, you will probably host many different kinds of Web sites. Commercial sites generate the most money, but you can earn money by hosting personal Web sites as well. IIS 5.0 contains features that can help you set up different kinds of sites, enabling you to get the most out of your investment.

Creating Web Sites

An ISP typically hosts two general types of sites: large, company sites with domain names, and smaller, personal sites with dial-in services. Large, company sites often consist of static Web pages that might include advertising, interactive pages, or e-commerce. Personal sites usually contain the Web pages of individual subscribers. The following sections tell you how to register and set up both types of sites.

Before setting up any sites, configure all disk drives on your servers in the NTFS file system format in order to take advantage of the complete Microsoft® Windows® 2000 security system. For information about setting up security, see "Security" in this book.

Creating a Company Web Site

The following series of procedures show you how to set up a full-domain site, which will have an address in this format: http://domain.com. You will learn how to configure the site so that anonymous users will be able to read the content, and so the owner will be able to transfer content to the site through File Transfer Protocol (FTP) or FrontPage Server Extensions. If there will be more than one owner authoring the site, create user accounts for the additional authors and add them to the user group corresponding to the Web site's domain.

To begin setting up your site

1. Register a domain name with InterNIC (http://www.networksolutions.com/internic/internic.html) or with the appropriate authority for registration in your area.

Top-Level Domain	Registry
.com, .edu, .net, .org	http://www.networksolutions.com/internic/internic.html
.us	http://www.isi.edu/in-notes/usdnr/
.gov (U.S. government)	http://www.registration.fed.gov/
.mil (U.S. military)	http://www.nic.mil/
.ca (Canada)	http://www.cdnnet.ca/
.mx (Mexico)	http://www.nic.mx/
Europe	http://www.ripe.net/
Asia/Pacific	http://www.apnic.net/

   After you have registered a domain name, you can assign an IP address to the domain. If clients will be able to access your site only through HTTP, but not through FTP you can assign a host-header name in IIS 5.0 in order to associate multiple domain names with the same IP address. For details, see the "Naming Web Sites" topic in the IIS 5.0 online product documentation.

2. In the DNS Management tool, add the Domain Name System (DNS) entry for the new domain.

   For information about how to use this tool, see the Microsoft® Windows® 2000 Server online product documentation.

3. Add the IP address to the host system's Transmission Control Protocol/Internet Protocol (TCP/IP) configuration.

   Note You may need to restart your computer in order to activate the IP address.

4. Create a home directory for the domain Web site. One option is to create it as a subdirectory of the C:\Inetpub directory.

Next, you need to create and configure an anonymous user account. This is accomplished through the Microsoft® Windows® 2000 Computer Management tool.

To create an anonymous user account

1. Right-click **My Computer** on the Windows desktop, and select **Manage**.
2. In the Computer Management tool, double-click **System Tools**, and then **Local Users and Groups**.
3. Create a Windows user account for a primary administrator of a Web site, and grant Log on locally permissions.
4. Create a new anonymous user account for the new Internet domain. An example would be: IUSR_*sitename*, where *sitename* is the name of the domain.

   This step allows you to set unique discretionary access control lists (DACLs) for the home directory of the Web site domain. For detailed information about NTFS DACLs, see the "About Access Control" topic in the IIS 5.0 online product documentation.
5. Assign the following settings to the new anonymous user account:
   - **User cannot change password**

     Choose this option as a minor security precaution to prevent the user from changing the password on the proxy server.
   - **Password never expires**

     Choose this option unless you want to periodically dispose of the password and create a new one.
6. Make the anonymous user a member of the **Guests** group.
7. Create a new Windows global group for the domain, and add the new anonymous user to that group as well.

   In order to more easily identify which domain the group belongs to, create the group name so that it is similar to the domain name.

For detailed instructions about the Computer Management tool, see the Windows 2000 Server online documentation.

Next, set the file system and IIS 5.0 permissions.

To set permissions

1. On the home directory for the Web site, set a DACL for the global group that you have just created.

   The DACL should grant members of this group Full Control over the home directory and its contents. Figure 8.1 shows the Windows 2000 Server dialog box in which you set Full Control permission for a user or group.

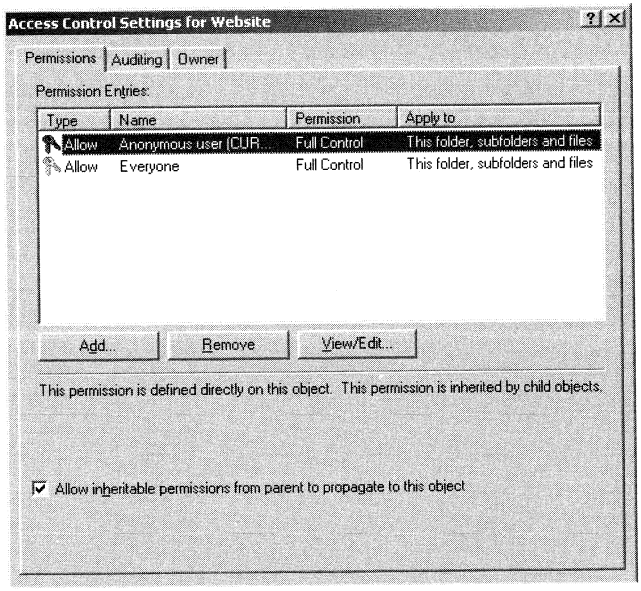


Figure 8.1 Dialog Box for Full Control Permission

2. Create a virtual server for the domain, granting Read permission.
3. If the virtual server will contain script programs (.asp files) that clients will be able to run, in the IIS snap-in, on the **Home Directory** property sheet for the server, select **Scripts only** in the **Execute Permissions** box.
4. In order to store your executable programs (.exe files) in one location, create a \Bin subdirectory, establish it as a virtual directory, and grant Execute permission.
5. Set a DACL in order to grant Read permission to IUSR_*sitename* for the home directory and its contents.
6. Set the anonymous user for the virtual server so that it uses the anonymous user account for the domain (which you created earlier as IUSR_*sitename*).
7. Set the default pages for the site in this order: Default.htm, Index.htm, Default.asp.

If you've installed FrontPage Server Extensions, you will need to create a home page (file), set up the Web site administrator, and configure the virtual server in order to publish information on the site.

To set up publishing

1. Create a file named Default.htm and add it to the home directory.

   You can indicate in this file that the site is under construction.

2. If you want to upload files to the site through FTP, create an FTP virtual directory for the virtual server.

   Note You can only do this if you have a dedicated IP address for the Internet domain. For information about uploading through FTP, see "Uploading Content through FTP" later in this chapter.

3. If necessary, create an additional user account and designate the user as a Web-site administrator.

   Do this only if you want to allow the additional user to control the IIS 5.0 configuration settings for the domain.

4. Set up the virtual server and the anonymous user as the owner/author of the site.

   For more information, see "Configuring FrontPage Server Extensions" later in this chapter.

If you want to allow a user to dial in from a remote computer, you can grant Dial-in permission.

To grant Dial-in permission

1. In the Computer Management tool, double-click **System Tools**, **Local Users and Groups**, and then **Users**.

2. Right-click the user to whom you want to grant Dial-in permission, and click **Properties**.

3. Click the **Dial-in** tab and, on the **Dial-in** property sheet, grant the user Dial-in permission.

Troubleshooting

Now that you've set up your domain site, you need to make sure it's running.

To verify that your site is running

1. Ping the IP address of the domain, using the Ping utility at the command prompt.
2. Run the NSLookup utility at the command prompt in order to find the domain name. First, type the command with the www prefix in order to see if the secondary domain name has been registered. Then, run the command again without the prefix in order to see if the top-level domain name has been registered.

   For example, at the command prompt type:

   nslookup domain.com

   And then type:

   nslookup domain.com

   This step verifies that there is a DNS entry.

   Note The DNS entry might not be active for one or two days after the Internet domain name has been registered.

3. As a final check, view the home page in Microsoft® Internet Explorer.

Creating a Personal Web Site

This section tells you how to set up Web sites for individual users. These sites will not have their own domain names, but will have an address in this format: http://domain.com/*username*/.

The procedure for creating a personal Web site is similar to the procedure described in "Creating a Company Web Site," except that the following items will not be created:

- User groups
- DNS server entry
- Separate anonymous user account

To create a personal Web site

1. Create a Windows user account for the user who will be publishing on this Web site.

   Optionally, create an e-mail account for the user.

2. Create a home directory for the user, and set DACLs to grant the user full control over the directory and its contents.

   For detailed information about NTFS DACLs, see the "About Access Control" topic in the IIS 5.0 online product documentation.

3. Create a virtual directory for the user and grant Read and Write permissions. This step will allow the user to add content to the site.

4. If you want clients to be able to run scripts in Active Server Pages (ASP) on the site, in the IIS snap-in, on the **Home Directory** property sheet, select **Scripts only** in the **Execute Permissions** box.

5. If you want executable (.exe) files to be available to clients, create a \Bin subdirectory and then create a virtual directory for it, with Execute permissions.

6. Create a file called Default.htm, add it to the home directory, and indicate in the file that the site is under construction.

7. If you want users to be able to upload content to the site, create an FTP virtual server for the directory that you created in step 2.

   For details, see the "Adding Sites" topic in the IIS 5.0 online product documentation.

8. If you are running FrontPage Server Extensions, set up the virtual server, and allow the user to be an owner/author.

Restricting Content

You might need to restrict access to some of your Web sites, or to portions of them. For example, you might want to restrict most of a site's content to members only, while allowing the general public to see just a generic page containing a submission form in order to join the site. You can restrict a site's content in several ways:

- Disclose the site's URL only to those people who should have access to the site's content.

  Because this method is not completely reliable, use it only for sites that contain nonsensitive information.

- Require members to log on, by assigning each user a logon account. Create a user group for those users who are members and restrict the site's content to this particular group.

  You can also restrict content of a site simply by setting a DACL that denies access to the anonymous user, but that allows access to authenticated users.

- Set a password page that will be verified by an ASP program.

  You can set a single password for all users or you can attach it to a database. This method, however, will prevent you from setting Windows DACLs. As a result, someone knowing a URL could bypass the password (except for the ASP content, which can check within each script). For more information, see "Security" in this book.

- Require Secure Sockets Layer (SSL) with client certificates either for logon or for comparison to a database, using a script in an ASP page.

- Create your own Internet Server Application Programming Interface (ISAPI) authentication filter.

  Creating your own ISAPI authentication filter becomes more complex if you need to restrict different areas of the site to different groups of people. You can set this configuration up through an ISAPI script or a script in ASP pages, or you can create Windows accounts. For information about setting up an ISAPI filter, see the "Installing ISAPI Filters" topic in the IIS 5.0 online product documentation.

- Set DACLs on individual files within the site in order to further restrict the content to specific users or groups.

Managing Your Installation

This section discusses features in IIS 5.0 that can help you manage your installation. Here you will learn how to enhance reliability, so that your sites will continue running even when your installation is being overloaded by simultaneous user connections. You'll also be introduced to two tools, included on the *Microsoft® Windows® 2000 Server Resource Kit* companion CD. These tools enable you to stress test your applications before publishing them to an active server, in order to make sure they won't overload your installation. IIS 5.0 comes with three features that allow you to automate administration tasks through the sample command-line scripts and to create your own customized administration scripts. After learning how to automate administration using these features, you'll see how to leverage several additional features of IIS 5.0 and Windows 2000 Server in order to administer sites from a remote computer. The remainder of this chapter discusses ways to manage content, to create sites through FrontPage Server Extensions, to upload content to sites with FTP, and to manage sites with Microsoft® Site Server 3.0.

Enhancing Reliability

As the Internet continues to become more and more popular as a method for communication and for information retrieval, providing reliable Web hosting becomes more of a challenge. Grouping servers into a Web cluster is an effective way to enhance reliability, allowing you to manage several servers as a single unit. With your servers working together in a Web cluster, you will be able to more quickly detect and recover from server failure.

A *Web cluster* is any Web site that is served by more than one computer. You can set up a Web cluster by using Microsoft® Cluster Service, and then let Network Load Balancing for Microsoft® Windows® 2000 Advanced Server distribute user requests to the different servers in the Web cluster. This will prevent any one server from becoming overloaded by requests. For suggestions about setting up Cluster Service, as well as information about Network Load Balancing in a three-tier configuration, see "Building a Web Cluster" later in this chapter.

If you administer a site consisting of several servers together in a Web cluster, all of the servers need to be kept running constantly, so that connections from outside users will not be interrupted. Windows 2000 Server, IIS 5.0, and *the Microsoft® Internet Information Services Resource Guide* offer suggestions to help you prevent a potential overload, as well as tips about how to handle an overload, if one does occur.

This section includes information about the following:

- Windows replication and clustering
- Site Server 3.0 Content Deployment
- IIS 5.0 fault tolerance
- Network Load Balancing feature of Windows 2000 Server
- Testing applications
- Troubleshooting applications
- Process isolation
- Tips on crash recovery

Replication and Clustering in IIS 5.0

Replication consists of copying content and configuration settings from one server to another, so that both servers will be able to offer identical resources to users. You must replicate the configuration settings for all servers within the Web cluster, regardless of whether or not they share content. *Content* replication is unnecessary, however, if your servers share a data storage device such as a disk drive.

IIS 5.0 comes with a command-line utility for replicating the IIS 5.0 metabase from one server node to other nodes. For details about how this utility works, see the "Replication and Clustering in IIS" topic in the IIS 5.0 online product documentation.

Many clustering applications support replication of both content and configuration settings. For details, see the documentation provided with your clustering application.

Replicating through Content Deployment

The Content Deployment feature (previously known as the Content Replication System) of Site Server 3.0 allows an installation to deploy content between Web servers quickly, securely, and reliably. (This content can include files, directories, DACLs, and metadata.) The servers can be local, on a corporate intranet, or on the Internet. Typically, you would initially deploy content on a testing site or *staging* server, where you can test and refine content and components before publishing them. Once testing is complete, you can then move the content to a production Web server (also known as an *end-point server*) connected to an intranet or the Internet.

Content Deployment simplifies the staging and deploying of Web pages and other file-based information, including applications. With this feature, it is easy to deploy content between directories, among local servers, as well as across geographically remote, secure networks, to multiple end-point servers.

Content Deployment Servers

Although a single server can deploy content between directories, most Content Deployment installations are more complex, involving multiple servers that perform different functions. The following list briefly describes the functions of two types of content deployment server:

- **Staging Server** Receives and deploys content from content authors or from other staging servers, and can also receive content from an HTTP or FTP server during an Internet retrieval. On this server you test, or *stage,* the content before deploying it to end-point servers. Testing allows you to review the content and make sure that all the links are functioning properly. Once testing is complete, the staging server deploys the content, either to other staging servers at remote sites or to end-point servers.

  Each staging server can contain multiple project directories for storing Content Deployment projects. When replicating content, Content Deployment searches the project directories for all relevant content files, directories, and subdirectories, based on filters that have been applied. The feature then deploys those files to the next server.

  Staging servers operate in the Windows 2000 Server environment and should have a network or Internet connection to the other Content Deployment servers.

- **End-Point Server** Receives deployed content from the staging server. This server is not capable of deploying content, but it is able to respond to user requests for Web pages. The end-point server contains a destination directory, which receives deployed content. The publishing administrator identifies the destination directory when creating a project.

  End-point servers can be Windows- or UNIX-based.

  Note UNIX-based end-point servers can only receive content that has been deployed from Windows-based servers. In addition, UNIX-based end-point servers cannot perform Internet retrievals.

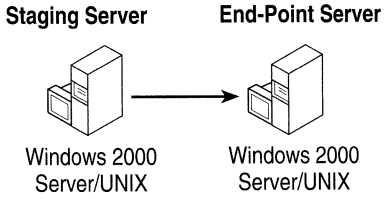
Benefits of Content Deployment

Content Deployment is not a tool for authoring Web pages or other content. Rather, it moves content from one server to one or more other servers. In addition to deploying content, Content Deployment quickly, securely, and reliably deploys and installs server applications, including COM components and Java applets.

Content Deployment does the following:

1. Uses the TCP/IP protocol and Windows 2000 Server authentication in order to create secure connections among Content Deployment servers.
2. Deploys data reliably across an intranet or Internet network through sophisticated data validation.

   For example, for small sites you can move content from a staging server to a production directory on the same server. For large sites you can deploy content from a staging server to a midpoint distribution server, in order to perform a final check and test. Once the content has been approved, you can disseminate it to Windows- or UNIX-based production Web servers around the world.

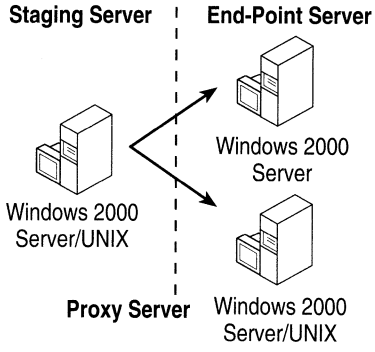


Main Features

The following list describes the new offerings in the latest version of Content Deployment. For more information, see the Site Server 3.0 product documentation.

- **Improved User Interface** Globally stages and deploys content across several servers. In addition, it sets up, administers, and monitors deployments—all through the IIS snap-in.
- **Same Server Deployment** Deploys content from a staging directory to a production directory within the same server.
- **Component Deployment** Deploys server applications, such as Microsoft® ActiveX® controls and Java applets, as quickly and easily as every other type of file.
- **Metabase Deployment** Deploys IIS 5.0 and virtual server settings to other Web servers, allowing you to clone your Web servers quickly and easily.
- **Improved Event Reporting** Logs server events to a database where you can easily monitor project activity and server load for your entire site.
- **Improved Performance** Deploys data faster than before, over slow connections.

- **Improved Content Posting** Works with the HTTP Post (RFC 1867) protocol in order to accept content from clients such as Microsoft® Web Publishing Wizard, Microsoft® Internet Explorer version 3.02 or later, and Netscape Navigator 2.02 or later.
- **Integration with FrontPage** Detects and deploys new Web content received from Microsoft® FrontPage®.
- **Integration with UNIX** Deploys content to Windows 2000 Server– and UNIX-based destination servers. The following illustration shows this type of deployment through a proxy.



Type of Projects

Content Deployment can replicate information by:

- Moving content.
- Moving components (files, directories, DACLs, and metadata).
- Retrieving information from the Internet or intranet.

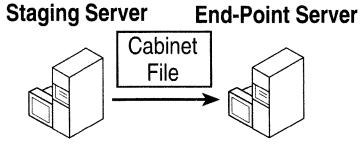
In Site Server, these methods of deployment are referred to as *projects*. All three types of projects, which are discussed subsequently, move content from a source to a destination. Both the source and destination could be on a single server or could be dispersed among multiple servers at local and remote locations.

- **Moving Content** Stages and deploys content by moving it from one directory to another within a single server, or from one server running Content Deployment to another. The content can range from static to dynamic pages, and can include applications and databases.

  Each server involved deploying content should contain an identically named project. The publishing administrator can create these projects manually or automatically, using the project wizard.

- **Moving Components** Disseminates compressed application files across a network, where they can be uncompressed and automatically installed on remote servers. Content Deployment allows you to replicate active content, including Java applets and Microsoft® Component Object Model (COM) components, to all Windows servers on your site.

  Java applets and COM components are small applications that you install on your servers. Content Deployment looks at the authenticode signature in order to verify that the Java applets and COM components are safe and certifiable. It then distributes them in cabinet (.cab) files that are installed and registered when they arrive at the destination servers.

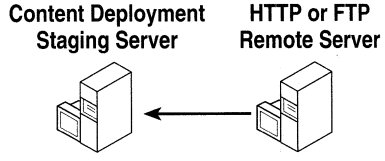


To deploy components

1. Create a project on every server involved in the deployment.
2. Issue the **Start** command.

   Note Components cannot be deployed to UNIX-based end-point servers or to Windows-based servers running earlier versions of Content Deployment.

- **Retrieving Information** Retrieves information from an Internet or intranet server by using HTTP or FTP protocol. Typically, the Internet or intranet server is not a Content Deployment server.



  The administrator creates a project on the Content Deployment server that will retrieve information over the Internet or intranet. When creating the project, the administrator references the base URL (pointing to the virtual directory where the content is stored) and specifies the *crawl depth* (the number of subdirectory levels under the base URL). Content Deployment will retrieve only those links and files that reside on the same Internet or intranet server as the URL.

  For example, if the administrator specifies that the HTTP retrieval should go two levels deep, Content Deployment will pull all of the content from the specified URL, plus the first level links, if those links are on the Internet or intranet server.

Clustering

Clustering allows two or more servers to appear to users as though they are one computer. The servers are connected not only physically by cables, but also programmatically through clustering software. This connection allows them to take advantage of features (such as fault tolerance and load balancing) that are unavailable to stand-alone server nodes. Clustered servers can also share disk drives that contain important information, such as a database.

If you have installed Windows 2000 Advanced Server, you already have software that will allow you to manage clustering. When linked together, Cluster Service and Network Load Balancing offer comprehensive availability and scalability for customers who are building applications with multiple tiers:

- Cluster Service gives you reliable application, transactional, file, and printing services on a two-node cluster. When combined with Microsoft® SQL Server™ Enterprise Edition or Microsoft® Exchange Enterprise Edition, Cluster Service adds reliable database and messaging services to your installation.
- Network Load Balancing extends the IIS 5.0 clustering technology in several ways. For example, in a multitier application, Network Load Balancing supplies load balancing and high availability for the first tier—the user interface. Because this kind of load balancing works through TCP/IP, a variety of workloads can benefit, including the load balancing of IIS 5.0 sessions, as well as Point-to-Point Tunneling Protocol (PPTP) and Virtual Private Network (VPN) sessions. Up to 32 servers can be formed into a Web cluster. Network Load Balancing is an extraordinary solution for the most demanding Web sites in the world, such as sites that need to support tens of thousands of simultaneous connections. Microsoft Web sites, including MSN (the third-busiest Web site in the world), prove the value of Network Load Balancing every day.

While Cluster Service reinforces the availability of database and messaging applications (back end), Network Load Balancing delivers reliability to IIS 5.0 Web servers (front end). For example, on an e-commerce Web site, you can cluster your front-end Web servers that are running IIS 5.0 with Network Load Balancing, and have them access a back-end cluster that is running SQL Server Enterprise Edition.

Additional features in Windows 2000 Advanced Server make it easy to set up and manage clustering on your installation. These features include:

- Improved usability, including wizards for creating applications, setting up virtual servers, and configuring and managing clusters on your installation.
- Failover of system services, including all existing services—Web, file, print, and Message Queuing—plus support for Dynamic Host Configuration Protocol (DHCP), Windows Internet Name Service (WINS), and distributed file system (DFS).
- Orderly rolling upgrade to individual cluster nodes without taking the cluster, as a whole, offline.
- Appropriate support for Windows Management Instrumentation (WMI) and the IIS snap-in, as well as integration with Microsoft® Active Directory™.
- Network adapter failure detection and forced failover.
- Plug and Play support for network adapters and disk drives (allowing hot swappable components to be replaced without shutting down the system).

All of these clustering components work together to greatly increase reliability through fault tolerance and load balancing. For details, see the "Replication and Clustering in IIS" topic in the IIS 5.0 online product documentation.

Fault Tolerance and Load Balancing

These features enhance the reliability of your installation by minimizing any potential interruption of service. If one server node stops working, another server node will immediately pick up the request load, with minimal disruption to users. Fault tolerance is made up of two components, *failover* and *failback*.

Failover transfers one server node to another node. Failback restores the load to the failed server node, after that node is back online.

Load balancing refers to two or more servers that support large amounts of activity by equalizing the request load among them. With Network Load Balancing, you can assign Web or FTP sites to a specific preferred server node, either manually or programmatically. In other words, a group of front-line Web servers would handle requests in the following manner: If an HTTP request comes through when one of the servers is already very busy processing requests, Network Load Balancing would direct the HTTP request to the idle server. In this way, you make sure users can access information quickly, even though a site is receiving a large number of simultaneous requests.

> **Note** Typically, you should configure clustering to enable both fault tolerance and load balancing.

Running Applications

On the Resource Kit companion CD, you'll find two tools that can further enhance reliability, by testing applications before they are made available to users. These tools are called the Web Application Stress Tool (http://webtool.rte.microsoft.com) and the Web Capacity Analysis Tool (WCAT). This section tells you how to tune your applications by using throttling processes, and gives guidelines for running applications in isolation.

Testing Applications with the Web Application Stress Tool

The Web Application Stress Tool, available on the Resource Kit companion CD, simulates having multiple browsers request pages from a Web application at the same time. By masking some of the complexities of Web server testing, the Web Application Stress Tool allows you to focus on gathering performance data on a Web site. This version of the Web Application Stress Tool offers the most up-to-date features for stress testing three-tier personalized Web sites that contain ASP pages and that run on Microsoft® Windows® NT Server version 4.0 and Windows 2000 Server. For more information about three-tier sites, see "Building a Web Cluster" later in this chapter.

The following list highlights some of the features offered by the Web Application Stress Tool. These features allow you to:

- Create scripts by hand, by recording browser activity, by pointing to an IIS 5.0 log file, or by selecting files in your content directory. For details, see the Web Application Stress Tool documentation included on the Resource Kit companion CD.
- Run a test script with any number of client machines, all of which are controlled from one centralized master client. The Web Application Stress Tool handles the allocation of threads and users, and collects report data from all computers.
- Control Web stress tests from a remote location through either the C++ or the ASP version of the Web Application Stress Tool client.
- Create multiple users so that authenticated and personalized Web sites can be accurately stress tested.
- Personalize realistic test scenarios. By default, support for ASP sessions allows cookies to be associated with users.
- Log off the client without interrupting a test, because the Web Application Stress Tool runs as a Windows service.
- Simulate modem throughput and increase the number of concurrent users, utilizing the Web Application Stress Tool's built-in bandwidth throttling.

- Customize your own stress run by using the stress templates provided with the Web Application Stress Tool. These templates change the number of users who are requesting pages on a site, simulating real-life site activity.
- Import, store, and edit complex query-string name-value pairs by using the query-string editor.
- Use the object model in order to create your own test client. By exposing all of the properties, methods, and constants, you can customize your own interface.
- Assign the IP address or DNS name of a specific server to make the Web Application Stress Tool round-robin among the servers in your cluster.
- Test for racing conditions, because the Web Application Stress Tool contains a delay between requests that is changeable.
- Complete online definition help using expert discussions about stress testing techniques.
- Customize all your test data, which the Web Application Stress Tool stores in a Microsoft® Access database.

Testing Applications with WCAT

WCAT, available on the Resource Kit companion CD, simulates various workloads on client-server configurations. Using WCAT, you can test how your IIS 5.0 and network configurations will respond to different client requests for content, data, or HTML pages. With the results of these tests, you can determine the optimal server and network configurations for your computer. For more information about WCAT, see "Monitoring and Tuning Your Server" in this book, and the WCAT documentation available on the Resource Kit companion CD.

Because Web servers are vulnerable to overload, it is crucial to virtually stress your Web server before rolling it out into a production environment. Since your server might be idle one minute, but then might suddenly receive 10,000 simultaneous requests, you should stress your application in order to see how it will react in an overloaded environment. When running stress on an application, increase the number of requests, starting at 100 and going as high as 100,000. You can then use System Monitor to see the effects that the stressed application is having on the server. Make any necessary changes to ensure the availability of your application.

The next section briefly tells you how to run a WCAT script, how the script can test a sample workload on your site, and how to interpret the results.

Running a WCAT Script

A WCAT script is split across three files, as shown in Table 8.1:

Table 8.1 Distribution of Files

File Name	Type of File	Contents
Script.cfg	Configuration file	The number of clients, the number of threads, the duration of the run, and so on. You can overwrite most of this information on the command line.
Script.dst	Distribution file	A collection of pairs (*ClassID*, *Weight*). The class IDs come from Script.scr. The Weights should add up to 100.0.
Script.scr	Script file	One or more transactions, each identified by a class ID. A transaction is one or more GETs or POSTs and an associated set of headers and other attributes to be sent to the server.

A large collection of sample scripts is available in the Scripts subdirectory.

To run a WCAT script

1. On the client computer(s), type the following at the command prompt:

```
cd \wcat\client
client.bat
```

   If the client computer(s) are correctly configured to point to the controller (not the server), you don't need to do anything more to the client(s).

2. On the Controller, type the following at the command prompt:

```
cd \wcat\control
run.cmd some-script [parameters]
```

   Note To see a full list of parameters, type **wcctl -?** at the command prompt.

Checking Performance Counters

After running a WCAT script, you should check the WCAT log file. You can also check the counters in the System Monitor of Windows 2000, which graphically displays the results of running WCAT. For details about System Monitor and for a description of the counters, see "Monitoring and Tuning Your Server" in this book. In addition, see the "Counters Reference" topic in the IIS 5.0 online product documentation.

Interpreting the Results

WCAT writes copious details to the .log file. The line in the log file that is most useful to administrators is **Pages Read**, which displays the following information:

- The first numeric column displays the total number of pages read by all client machines.
- The second column shows the rate of pages per second seen by the first client machine.
- The third column shows the total number of pages seen by the first client machine.

If you have more clients, you will see the rate and total numbers for each machine in subsequent columns.

WCAT Troubleshooting

The following tips will help you to gain the maximum performance benefit from WCAT:

- Because WCAT assumes that all clients are the same, and because it is also unable to recognize HTML code, you must spell out the contents of framesets and image URLs explicitly in your transactions, by clustering them under one class ID.
- If performance suddenly improves, verify that an error has not occurred.
- Turn off IIS logging while stressing your applications; otherwise, you'll receive huge logs full of useless data.

Testing Conditions

This section gives you tips about how to test the performance of your applications and how to stress them before making them available on the Internet or on an intranet. With these tips, you can monitor an application's maximum performance and also see how much of a load the application can take before it breaks.

Performance Testing

In order to test a Web application's maximum possible performance, you'll need the following:

- An isolated private network
- Enough clients to saturate the server
- At least 100 megabits per second (Mbps) of network bandwidth
- Several network adapters to distribute the load
- A multiprocessor machine for scalability testing

First, test the pages with your browser in order to verify that the application is running correctly.

Note You will need to rerun tests in order to verify that results are reproducible.

Because you don't run an application in a vacuum, maximum performance is not always what you want to test. For example, you might also want to know how well your application performs under an average load, coexisting with other Web sites and other non-Web applications on the server. To test for average performance, set some goals for the application and see if it can meet them. For example:

- More than 20 pages/second
- Less than 50 percent capacity utilization of CPU
- Less than 10 seconds response time

Stress Testing

For high-stress testing, as mentioned earlier, you need a private network and a collection of client machines. But for average stress testing, you might not necessarily need a private network and lots of clients. Putting your application under a moderate load is often enough to expose bugs. Therefore, running the controller and a client on the server will suffice. Finally, running stress tests on a multiprocessor computer is a great way to find threading bugs.

Running Applications in Isolation

Once you've tested your applications and are satisfied with their performance, you're ready to make them available on the Web. IIS 5.0 allows you to run applications in one of two ways: in process or out of process (in isolation). Process isolation allows multiple Web applications to run on one Web server without crashing, by running each application that is on the server in a separate memory space (out of process).

The advantage of running everything out of process means that if one application fails, it won't cause the others to fail. The disadvantage is that this adds an extra step when diagnosing problems. Despite the disadvantage, it is recommended that you run an application out of process when you're not sure of its integrity.

In the IIS 5.0 architecture, all the IIS services run in the Inetinfo process. (You can look at this process by opening the Task Manager and clicking the **Process** tab.) When you run an application out of process, you will find an associated Mtx.exe process in the Task Manager. (For example, if you were running 10 applications out of process, you would find 10 separate Mtx.exe files.) If your application fails, the associated Mtx.exe fails, leaving either the Inetinfo process or your Web server intact. Once you're confident that the application is reliable, bring it into the Inetinfo process by clearing the **Run in own memory space** check box, which can be found in the application's property sheets.

Running applications out of process is an important feature for ISPs because many of their customers want to use custom components. If these components are run in process, it is common to see that the Inetinfo process is exceeding normal performance parameters and, as a result, to falsely assume that IIS 5.0 is running poorly. But in reality the poor performance is usually caused by a custom component that is leaking memory. Had that component been run in its own memory space first, the associated Mtx.exe would have reflected the performance cost. The solution, accordingly, would have been to end the Mtx.exe process, not the Inetinfo process.

It can strain your computer's resources to run applications out of process, since doing so takes away resources from the Web server. Despite this drawback, it is best to initially run new applications out of process to make sure they are stable and meet your requirements. Once you have done that, you can then safely run them in process.

Here are some guidelines on running various types of applications in and out of process.

- Common Gateway Interface (CGI) applications are run out of process, but they offer poor *scalability*. In other words, every instance of a CGI application requires the creation of a new process with its associated overhead, and application parameters are passed through environment variables.
- ISAPI applications can also be run out of process. When you do this, you'll attain the same level of application integrity as CGI applications, but with faster performance. You can further improve performance by running ISAPI applications in the same memory space as the server (in-process). Remember that in-process applications don't yield as much application integrity as those run out of process.
- Some ISAPI dynamic-link libraries (DLLs) can't be isolated. Examples include DLLS that rely on locking resources, such as open files, shared memory, and so on. In such cases, you should rewrite the application to remove the dependencies.
- An individual server can host a mix of in-process and out-of-process applications. With this mix, you can run well-tested applications in-process for faster performance, and run others out of process during development or beta stages for "fail-safety" reasons.

> **Note** While both ISAPI and ASP process isolation are managed through Microsoft® Component Services, process isolation is managed through the IIS snap-in.

Out-of-Process Application Pooling

With IIS 5.0, ASP applications run in an out-of-process application pooling by default. When you create a virtual directory with the wizard in the IIS snap-in and then create applications to run in that directory, the applications will run in a separate process from IIS 5.0. The separate process is a special COM+ application, of which each pooled ASP application is a component.

The following guidelines will help you determine when to run applications in the pool for maximum stability and performance:

- If you're running 20 or fewer applications, run them out of process.
- If you're running 21 or more, add them to the pool.
- Run a stateful application out of process.
- If you're running 20 or more applications, some stateful and some stateless, run the stateful ones out of process and add the stateless ones to the pool.

You can set the pooling parameter in the IIS snap-in, or through Adsutil.exe on the command line. For details about pooling and for procedures, see the IIS 5.0 online product documentation.

Hosting Applications

ISPs often establish preproduction sites (or staging servers) to test customer applications before they are hosted on the production site. (For more information about staging servers, see "Replicating through Content Deployment" earlier in this chapter.) During this test phase, you can save resources by using one server to host multiple customer test sites.

ISPs hosting Web-based applications should establish guidelines for those who are writing them. For example, it is not good to write applications that can be run *only* in process, since those that can be run out of process are safer for the installation as a whole.

Recovering from Crashes

Even after rigorous testing, it's still possible that, at one time or another, an application will fail when it's run on a production Web server. To handle crashes and still protect data, IIS 5.0 offers two convenient ways of recovering when the Component Services environment hosts applications (ISAPI and ASP applications running out of process, for example):

- If a fatal application error occurs, the process is terminated automatically.
- If a transactional application crashes, all transactions in progress are stopped and any changes to the data are rolled back.

  The Windows® Event Log stores a record of the error that occurred, and the run-time Component Services environment automatically restarts the application process. In this way data is protected and the application can be rerun.

Automating Administration

Administering Web sites can be time consuming and costly, especially for people who manage large ISP installations. Consequently, many ISPs support only large, company Web sites, at the expense of personal Web sites. Nevertheless, a cost-effective way to support both would be to automate administrative tasks and let users administer their own sites from remote computers. This solution reduces the amount of time and money it takes to manually administer a large installation, without reducing the number of Web sites supported. IIS 5.0 offers three technologies that help you automate administration:

- Microsoft® Windows® Script Host (WSH)
- IIS Admin Objects
- Microsoft® Active Directory Service Interfaces™ (ADSI)

With these three technologies, you can administer sites from the command line of a central computer, as well as group frequently used commands in batch files. All you need to do is run the batch files to add new accounts, change permissions, add a virtual server to a site, and so forth.

In this section, you'll learn how to carry out basic administrative tasks, by running the sample ADSI scripts installed with IIS 5.0. These scripts can give you faster, more cost-effective administration. For example, you'll see how to create a new virtual directory on a remote server and then change that directory's write access. You'll also see sample custom scripts that can change Windows permissions on a server.

ADSI sample scripts and the WSH environment are installed by default, when you install Windows 2000 Server and IIS 5.0. Although the sample scripts are fully functional, they also serve as templates from which you can create your own scripts.

For detailed information about WSH and ADSI scripts, see the "Administration Scripts" topic in the IIS 5.0 online product documentation.

Windows Script Host

WSH is a language-independent scripting environment for 32-bit Windows platforms. Microsoft offers both Visual Basic® Scripting Edition (VBScript) and JScript® scripting engines with WSH, while third-party companies supply ActiveX scripting engines for other languages, such as Perl.

WSH can automate administrative tasks on a server. For example, an administrator can write a program in VBScript that will create a new virtual directory. With WSH, the script file can then be run from the command line, in order to create a new virtual directory on the Web site. In addition, an administrator can write a single script that will target multiple Web sites or multiple physical servers. For more information, see the Windows Script Host material in the Windows documentation.

IIS Admin Objects and ADSI

The IIS Admin Objects installed with IIS 5.0 make programmatic administration straightforward. Based on ADSI, these objects are compatible with automation and, as a result, can easily be accessed and manipulated by any language that supports automation, such as VBScript or JScript, Microsoft® Visual Basic®, Java, or C++.

When you install IIS 5.0, the sample IIS Admin Objects scripts are copied into the Inetpub\AdminScripts directory by default.

You can use these sample scripts to configure your IIS 5.0 installation, to create virtual directories, to display information about a Web site (see the example in the following section), or to manage the status of Web sites by stopping, pausing, and starting IIS.

In addition to using the sample scripts as they are, you can customize them by extrapolating from the annotated examples. With custom scripts, you can administer your IIS 5.0 configuration by changing the settings stored in the metabase. The structure of the metabase parallels the structure of your IIS 5.0 installation, and the property inheritance feature of the metabase allows you to set up IIS 5.0 configuration settings in an efficient manner.

For details about ADSI objects, see the "ADSI Object Methods" topic in the IIS 5.0 online product documentation.

Executing Scripts

Adsutil.vbs, an administration utility installed with Windows 2000 Server and IIS 5.0, runs on VBScript in conjunction with ADSI. Use Adsutil.vbs together with Cscript.exe, which is installed with WSH, in order to manipulate the IIS 5.0 configuration.

You can execute an administration script by typing the appropriate syntax on the command line. The following example shows the current settings for the root of the default Web site:

```
cscript.exe inetpub\adminscripts\adsutil.vbs enum w3svc/1/root
```

To execute any other script on the command line, type:

Cscript.exe *path\ScriptName*

Instead of typing the syntax on the command line, you can type it in a batch (.bat) file, and execute your scripts from there. If you execute the same script repeatedly, putting it into a batch file will save you from having to retype the script every time you want to run it.

If you've added the path to the Inetpub\AdminScripts directory, you don't have to type the full path to Cscript.exe. For information about editing your path, see the Windows 2000 Server online documentation.

> **Note** If you have registered Cscript.exe as your default scripting host, you don't have to type **Cscript.exe** in front of the scripts to execute them. (To learn how to register Cscript.exe, see the first two steps in the next section.)

Examples

The following examples show you how ADSI scripts can automate tasks that ISPs must perform repeatedly. For a detailed explanation of the script syntax, see the IIS 5.0 online product documentation.

The first two examples assume that you've already set the path to the administration scripts within the Windows environment, or that you are running the scripts in the C:\Inetpub\AdminScripts directory, where they reside. The following procedure tells you how to set the path to the administration scripts.

To add a path to the Windows environment

1. On the Windows 2000 Desktop, right-click **My Computer**.
2. Select **Properties**.
3. On the **System Properties** dialog box, click the **Advanced** tab.
4. Click the **Environment Variables** button.
5. In the **System Variables** box, select the line beginning with the word "path."

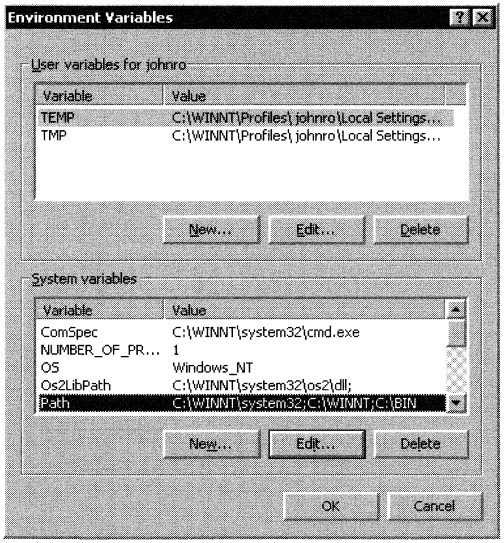


Figure 8.2 Dialog Box for Environment Variable

6. Click the **Edit** button.
7. In the **Edit System Variable** dialog box, put the cursor at the end of the line in the **Variable Value** box.
8. Type the following to add the path to the sample ADSI scripts:

```
;C:\Inetpub\AdminScripts
```

> **Note** Each element in the path variable must be separated by a semicolon, which explains this initial punctuation.

Changing Permissions

This example shows you how to change permissions for a virtual directory located on a different server. Suppose you want to grant Write permission on a virtual directory called *Test*. The Web site is on your company's intranet on a computer named *Server2*, located in another building. Instead of walking over to the other building, you prefer to make this change from the computer (*Server1*) in your office.

To change permissions for a Web site on another server

1. On your administration server, open a command prompt.
2. Type **adsutil** and accept the defaults in order to register Cscript.exe.

   If you don't want to register Cscript.exe (because you are running another scripting host by default), you have to type **Cscript.exe** in front of the script you want to run. See the example in the previous section, "Executing Scripts."
3. At the command prompt on Server1, type the following line:

   ```
   Adsutil set w3svc/1/root/test/accesswrite "true"-S:server2
   ```

   test is the virtual directory on which permissions are reset.

   accesswrite changes Write permissions.

   true grants Write permissions.

   -S:server2 is the server where the Web site is located.

Users can now upload files to the virtual directory Test on Server2.

Creating a Web Site

This example creates a new Web site with the sample ADSI scripts. Suppose you want to create a Web site on your production Web server for a new client that you will be hosting. The repetitive work of creating site after site can easily be automated by running the Mkw3site.vbs script.

Once you've set the path, as explained earlier, you can create a Web site.

To create the Web site

1. On your administration server, open a command prompt.
2. If you have not registered Cscript.exe, register it as described in step 2 of the previous example.

3. At the command prompt on the server, type the following line:

```
mkw3site -r c:\webs\customer1 --DontStart -t "Customer 1 Site" -o 80
-i 172.16.100.1
```

mkw3site includes the script to make the Web site.

–r is the root directory of the Web site.

–DontStart creates the Web site in a stopped state. To start it, you must activate it through the IIS snap-in, or from the command line by typing **net start w3svc**.

–t defines the title of the Web site.

–o defines the port number.

–i defines the IP address for the Web.

Customizing Scripts

The following sample ASP pages show you how to set up customized scripts. The first deals with setting permissions in general, while the second shows you how to set up a virtual directory with Read, Script only, and Directory browsing permission.

Web server permissions control how users access and interact with specific FTP and Web sites. For example, permissions determine whether users visiting a Web site are allowed to see a particular page, upload information, or run scripts on the site. Unlike NTFS permissions, Web server permissions apply to all users accessing a Web or FTP site. This distinction is very important, because NTFS permissions apply only to a specific user or group of users with a valid Windows account.

The following example contains customized scripts that set permissions on a virtual directory in two ways:

- **GET/PUT notation** Choose this type for script that contains variables.
- **Dot notation** Choose this type for hard-coded scripts.

The example also shows how inheritance works. In the first part, the GET/PUT notation denies Write permission (sets it to FALSE) from the root level of the default Web site on MyComputer. When you set permissions at the root level, all directories below the root level inherit this setting. However, the dot notation (the second part of the example) grants Write permission (sets it to TRUE) on the virtual directory named *VDir1a,* overwriting the inherited setting that is at the root level.

```
<%
  Dim WebServerRootObj
  Dim VDirObj
  Dim WritePerm

  'Open the object for the first virtual Web server root.
  Set WebServerRootObj = GetObject("IIS://MyComputer/W3SVC/1/Root")

  'Deny write access for all directories and files
  'for the server (except those already specifically set)
  'Using the Put method.
  WebServerRootObj.Put "AccessWrite", False

  'Save the changed value to the metabase.
  WebServerRootObj.SetInfo

  'Get a directory subordinate to the Web server root.
  Set VDirObj = GetObject("IIS://MyComputer/W3SVC/1/Root/Vdir1/VDir1a")

  'Overwrite the inherited value for write access
  'Using the dot method equivalent to Put.
  VDirObj.AccessWrite = True

  'Save the changed value to the metabase.
  VDirObj.SetInfo
%>
```

The next example shows a customized script that creates a virtual directory with Read, Script only, and Directory browsing permissions.

```
<%
  '''''''''''''''''''''''''''''''''''''
  'ADSI ASP Sample Program.
  'This is a sample of how to create a virtual directory using ADSI.
  '
  '''''''''''''''''''''''''''''''''''''
  Option Explicit
  On Error Resume Next

  ''''''''''''''''''''''''''
  'First, open the path to the Web server you are
  'trying to add a vdir to.

  Dim ServObj
  Dim VdirObj
  Dim Testpath
```

```
  Set ServObj = GetObject("IIS://LocalHost/w3svc/1/Root")
  if (Err <>0) then
     Response.Write "GetObject (""IIS://LocalHost/w3svc/1/Root"") Failed!     <BR>"
     Response.Write "Error! " & Err.Number & "(" & Hex(Err.Number) & "):   " &
Err.Description & "<BR>"
     Response.End
  end if

  '''''''''''''''''''''''''
  'Second, Create the vdir path.
  Set VdirObj = ServObj.Create("IIsWebVirtualDir", "MyVdir")
  VdirObj.SetInfo
  if (Err<>0) then
     Response.Write "CreateObject   (""IIS://LocalHost/w3svc/1/Root/MyVdir"") Failed!<BR>"
     Response.Write "Error! " & Err.Number & "(" & Hex (Err.Number) & "):   " &
Err.Description &

  "<BR>"
     Response.End
  end if

  '''''''''''''''''''''''''
  'Finally, create a Path variable containing the VR path and
  'set the permissions to read, script, and directory browsing.
  VdirObj.AccessRead = True
  VdirObj.AccessScript = True
  VdirObj.EnableDirBrowsing = True
  Testpath = "C:\Temp"
  VdirObj.Put "Path", (Testpath)

  VdirObj.SetInfo
  if (Err<> 0) then
     Response.Write "Put (""Path"") Failed!"
     Response.Write "Error! " & Err.Number & "(" & Hex (Err.Number) & "):  " &
Err.Description &

  "<BR>"
     Response.End
  end if

  Response.Write "VDIR successfully created"

  '''''''''''''''''''''''''
  'The minimum amount necessary to create a virtual directory has now
  'been completed.  If you need to add more, do it here.
%>
```

For more sample scripts and for details on writing your own scripts, see the "Programmatic Administration Examples" topic in the IIS 5.0 online product documentation.

Administering a Site Remotely

With IIS 5.0 and Windows 2000 Server, you can administer a server from a remote computer. IIS 5.0 supplies three methods for administering sites remotely: the IIS snap-in, Internet Services Manager (HTML), and IIS Admin Objects working in conjunction with ADSI scripts. Under Windows 2000 Server, you can administer servers remotely through Microsoft® Windows® 2000 Terminal and Telnet Services.

IIS Snap-in

Windows 2000 Server comes with an Administration Tools package, which you can install on a computer that is running Microsoft® Windows® 2000 Professional. You can then set up the IIS snap-in on the remote computer, connect to an IIS 5.0 server, and administer IIS 5.0 through the snap-in. You can perform any IIS 5.0 administrative task on the remote server that you could if you were actually there. For details about how to handle administrative tasks through the IIS snap-in, see the IIS 5.0 online product documentation.

Internet Services Manager (HTML)

With Internet Services Manager (HTML), you can administer your Web installation over an intranet or the Internet. Although you can't perform all the tasks you can with the IIS snap-in, you can change access privileges, create Web sites, read log files, check or change properties. You can also start, stop, or pause servers from any computer on the network that is running Microsoft® Internet Explorer 5. Figure 8.3 shows the Internet Services Manager (HTML).

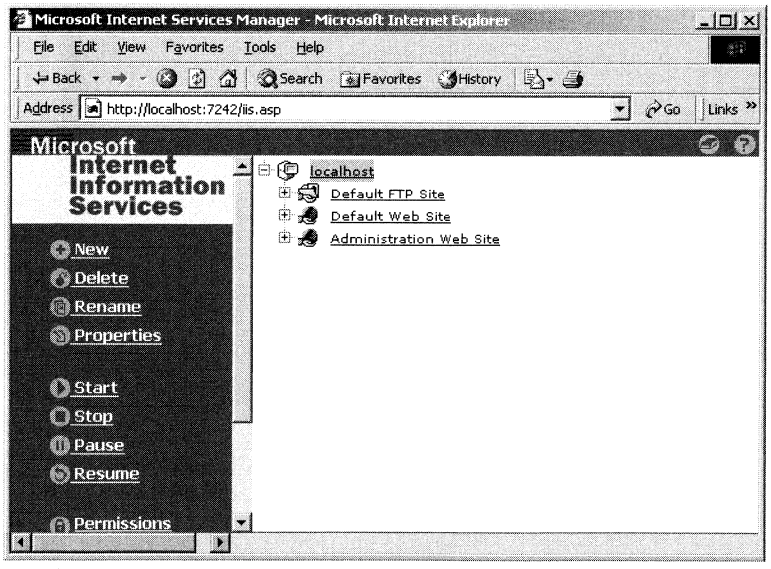


Figure 8.3 Internet Services Manager (HTML)

For details about how to handle administrative tasks through the Internet Services Manager (HTML), see the IIS 5.0 online product documentation.

Command Line

Based on Microsoft's ADSI, the IIS Admin Objects simplify command-line administration through automation. Using these features, you can perform many administrative tasks remotely. Possibilities include the following:

- Creating new FTP and Web sites
- Creating virtual directories
- Changing properties and permissions on Web sites and virtual directories

In addition, you can write your own ADSI scripts in order to customize administrative tasks.

Terminal Services

Although you can remotely manage IIS 5.0 Web sites from the IIS snap-in or the Internet Services Manager (HTML), you cannot perform Windows 2000 administrative tasks through them. However, with Terminal Services, you can administer a Windows 2000 Server as well as IIS 5.0 from a remote computer.

Terminal Services was originally created so low-powered machines (*thin clients*) could run applications that required more resources than the computer could offer. For example, you could run Microsoft® Office 2000 applications on a computer that is running Microsoft® Windows® version 3.1. But with Terminal Services client software installed, a client computer can connect to a computer that is running Windows 2000 Server (with Terminal Services server software), and run these applications through the server.

To install Terminal Services, you must select it when installing Window 2000 Server. Once the server is set up, you can install the client software on a computer that is running Windows 2000 Professional, Microsoft® Windows® 98, or even a 16-bit version of Windows such as Windows 3.1. From the client, you connect to the server through a local area network (LAN), PPTP, or dial-up connection and perform any administrative tasks you want, even though the server you're administering may be halfway around the world.

In addition to setting DACLs and performing other Windows 2000 administrative tasks, a Terminal Services client can administer IIS 5.0 remotely through the IIS snap-in. First, connect the client to the Windows 2000 Server. Next, mirror the server's desktop and the IIS snap-in on the server. Through the snap-in, you can perform any IIS 5.0 administrative task on the remote server that you could if you were actually at the site. For example, you can create Web sites, change access permission, or administer IIS 5.0 security. You can basically do anything that you could with the IIS snap-in.

For detailed information about Terminal Services, see the Windows 2000 Server online documentation.

Telnet

Remote administration through a Telnet client and server lets you remotely log on to and execute commands on operating systems based on Windows 2000 or UNIX. Microsoft® Telnet Service logs remote clients onto a server as though they were connected to the server directly through a local terminal. Telnet Service works like a basic TCP/IP Telnet Server. A computer running the Telnet Server tool under Windows 2000 Server can support connections from various TCP/IP Telnet clients, including UNIX-based and Windows-based computers.

The remote administration feature of Telnet lets users log on to UNIX-based and Windows-based computers in another location and execute commands on them. Telnet Service is now part of the Windows 2000 Services for UNIX. You can download Telnet from the Microsoft Web site, at http://www.microsoft.com/.

When you download and install Telnet, a Telnet client is installed in the **Accessories** folder. The Telnet client connects to another computer that is running a TCP/IP-based Telnet server.

The Telnet Server tool can execute commands (such as TCP/IP tool or Microsoft® MS-DOS® commands) in the command-prompt window. Here's how the command process works:

1. Once logged on to a Telnet server, you can enter a command at a remote Telnet client computer.
2. The Telnet server sends the results of the command back to the remote Telnet client computer.

Each Telnet-client session started on a Telnet server is allocated a console session, which doesn't interfere with any local-user sessions on the server. At the start of a session, the remote Telnet client must send a user name and password to log on to the Telnet server. This user name and password create a connection with the security attributes of the user.

For example, you can:

- Log on to a computer running Windows 2000 Professional and the Telnet client.
- Start the Telnet client.
- Connect to a computer running Windows 2000 Server and the Telnet Server tool.
- Run command-line scripts to add users, services, and so on.

For more information about Telnet service, see "Telnet Server Admin Utility" in the Windows 2000 online documentation.

Turning Users into Web Site Operators

Delegating administrative rights to your customers can present a security challenge. To meet this challenge, you have to make sure that each customer can administer only his or her own site. For ISPs who are hosting multiple sites on a single server, the Web site Operators feature of IIS 5.0 is the ideal tool for this. Web site Operators are Windows user accounts that have limited administration privileges on a Web site.

With this feature, you can give individual site administrators complete control over the properties, applications, and security of their site—without jeopardizing the security or configuration of other sites running on the same server. A Web site Operator cannot change bindings or port numbers, configure the anonymous user name and passwords, change the identification of the Web site, throttle bandwidth, change accounts on the server, add virtual directories, configure process isolation or ISAPI filters, or stop, pause, or restart a site.

For more information see "Assigning Web Site Operators" in the IIS 5.0 online product documentation.

Configuring FrontPage Server Extensions

This section tells you how to set up and configure FrontPage Server Extensions. You'll learn how to properly structure newly created virtual servers for FrontPage and how to transfer content from one server to another. You'll also learn how to set Windows 2000 and IIS 5.0 permissions that work compatibly with FrontPage Server Extensions. Finally, you'll see tips on how to properly configure e-mail to work with forms created in FrontPage Server Extensions.

Although FrontPage offers most of the same features as FrontPage Server Extensions, you must install the FrontPage Server Extensions on your production Web server in order for the following to work:

- Confirmation Field
- Discussion Form Handler
- Guest Book
- Registration Form Handler
- Save Results Form Handler
- Scheduled Image
- Scheduled Include Page
- Search Form
- Live Authoring

Introducing the FrontPage Snap-in

The FrontPage snap-in is a Windows 2000 interface similar to the IIS snap-in. While the IIS snap-in allows you to administer IIS 5.0, the FrontPage snap-in allows you to administer FrontPage Server Extensions and FrontPage-extended webs. A *FrontPage web* is a project containing all the pages, images, and other files that make up a Web site.

Since the FrontPage snap-in appears in the IIS snap-in, commands, property sheets, and other tools required to administer FrontPage Server Extensions are added to the IIS snap-in shell.

To access the FrontPage snap-in

1. On the Windows Toolbar, click **Start**, and then click **Programs**.
2. Click **Administrative Tools**, and then click **Internet Services Manager**.

You can verify that the FrontPage snap-in is installed by right-clicking the **Default Web Site** in the IIS snap-in and selecting **All Tasks**. Under **All Tasks**, you'll find FrontPage administration commands. If the Server Extensions are not installed on the Default Web Site, you will see the command **Configure Server Extensions**. If they are installed, you will see the following FrontPage administration commands: **Check Server Extensions**, **Open With FrontPage**, **Recalculate Web**, and **Remove Server Extensions**.

The FrontPage snap-in replaces and significantly improves upon the Fpsrvwin utility, an administrative program that was included with previous versions of FrontPage. The FrontPage snap-in's capabilities are similar to those of the Fpsrvwin, Fpsrvadm, and Fpremadm utilities.

With the FrontPage snap-in, you can do the following:

- Extend a virtual server with FrontPage Server Extensions
- Check and fix FrontPage Server Extensions on a web
- Upgrade FrontPage Server Extensions on a web
- Remove FrontPage Server Extensions from a web
- Delete a subweb (virtual directory under a FrontPage web) that has been extended with FrontPage Server Extensions
- Convert a subweb into a folder, and vice versa
- Recalculate all hyperlinks in a web
- Add an administrator
- Enable or disable authoring on a web
- Tune web performance
- Log authoring operations
- Require SSL for authoring
- Specify that a folder can contain executable scripts or programs
- Enable source control
- Set e-mail options

There are only two things that you can't do with the FrontPage snap-in:

- Administer the FrontPage Server Extensions from a remote computer
- Perform command-line scripting

For command-line scripting, use the Fpsrvadm utility, and for remote administration, use the HTML Administration Forms or the Fpremadm utility. All of these components are installed in the following directory by default: C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\40 Folder. For information about configuring these utilities, see the *Microsoft® FrontPage® 2000 Server Extensions Resource Kit* at http://officeupdate.microsoft.com/frontpage/wpp/serk.

You will know FrontPage Server Extensions are enabled if you right-click on a virtual server, select **All Tasks**, and see either **Configure Server Extensions** or any of the commands associated with a server that already has the extensions installed, like **Check Server Extensions** or **Recalculate Hyperlinks**. If you do not see these commands, enable the extensions.

To enable FrontPage Server Extensions

1. Click **Console** and then click **Add/Remove Snap-in**.

   If the Console menu shows only the **Options** item, choose **Options**, and select **Always open console file in author mode**. Click **OK**, and exit the IIS snap-in. Reopen it, and from the Console menu, select **Add/Remove Snap-in**.

2. Select the **Extensions** tab, and make sure **FrontPage Server Extensions** is selected.
3. If it is not selected, click the **FrontPage Server Extensions** check box, and click **OK**.

Managing Content

When creating and transferring FrontPage content, there are a few tips you should keep in mind. This section teaches you how to successfully move content from one server to another. It also tells you how to configure any content you might want to nest among virtual servers.

Transferring Content

The FrontPage **Publish** command transfers FrontPage content from one FrontPage-extended web server to another through HTTP. All of the functionality of installed FrontPage components is also transferred.

However, if you choose another method of publishing or transferring the content, such as FTP, Windows **Copy** command, or Site Server's Content Deployment, the FrontPage components will not be functional until you run FrontPage **Recalculate Web** on the production Web server. This command enables the FrontPage Server Extensions on the production Web server to parse the content and to recognize FrontPage components. If the _vti directories from the source server are also copied by FTP, you may need to remove the FrontPage Server Extensions and reinstall them on the destination server.

Because the _vti directories contain server-specific information, transferring them from one server to another (by copying or replicating content) moves the configuration information from one server to a new server, where the information might not apply. If this happens, FrontPage will not work. Should you encounter this problem, remove and reinstall the Server Extensions on the new server.

Nested Content

Before installing FrontPage Server Extensions, make sure you do not have content nested in overlapping virtual servers. Virtual server directories should not be configured under the root directory for another server.

For example, because the following configurations contain nested directories for the virtual servers, they will not work with FrontPage Server Extensions.

Example 1: Server running on Windows 2000

```
Default Site C:\Inetpub\wwwroot
Virtual Server 1 C:\Inetpub\wwwroot\virsvr1
Virtual Server 2 C:\Inetpub\wwwroot\virsvr2
```

Example 2: Server running on UNIX

```
Default Site /usr/local/www
Virtual Server 1 /usr/local/www/user1
Virtual Server 2 /usr/local/www/user2
```

If you have configured your virtual servers as in the examples, you must reorganize them in order for FrontPage Server Extensions to work properly.

Changing the Root of a Web Site

To change the root of an IIS 5.0 Web site with FrontPage Server Extensions

1. Uninstall FrontPage Server Extensions.
2. Change the document root.
3. Reinstall FrontPage Server Extensions.

> **Note** When you change the document root directory in IIS 5.0, you must also reset the permissions on that directory.

FrontPage 2000 now administers permissions at the content root level. Therefore, if you remove and reinstall FrontPage Server Extensions, your permissions are not lost. When you install them, FrontPage assigns the following permissions on the content root:

Type of User	Level of Permission Assigned
Site visitor	Read, Execute
Author	Read, Execute, Write, Delete
Administrator	Read, Execute, Write, Delete, Change Permissions

Overlapping Virtual Servers

You cannot install FrontPage Server Extensions onto a virtual server that overlaps another virtual server (nested virtual servers). If you try to install them, you'll get the following error message:

```
Unable to create a web for the url "/" because its root directory,
"c:\Inetpub\wwwroot\web", falls below the root of another web in the directory
"c:\Inetpub\wwwroot".
```

Overlapping Configurations

The following examples show overlapping configurations:

Example 1

```
C:\Inetpub\wwwroot\ [Default Web site on IIS]
C:\Inetpub\wwwroot\virtualserver1\
C:\Inetpub\wwwroot\virtualserver2\
```

Example 2

```
C:\Inetpub\wwwroot\
C:\Inetpub\webs\virtualserver1\
C:\Inetpub\webs\virtualserver1\virtualserver2\
```

Example 2 shows virtual servers in different directories with an overlapping configuration.

If you have a configuration like one of these, reconfigure it so that virtual servers do not overlap, before you install FrontPage Server Extensions.

Nonoverlapping Configurations

The following examples show configurations on which you can install FrontPage Server Extensions:

Example 1

```
C:\Inetpub\wwwroot\
C:\Inetpub\virtualserver1\
C:\Inetpub\virtualserver2\
```

Example 2

```
C:\Inetpub\wwwroot\wwwroot2\ [Default Web site remapped to \wwwroot2]
C:\Inetpub\wwwroot\virtualserver1\
C:\Inetpub\wwwroot\virtualserver2\
```

Example 3

```
C:\Inetpub\wwwroot
C:\webs\virtualserver1\
D:\websites\virtualserver2
```

Example 3 shows how to configure your virtual servers in different directories without any overlapping.

Setting Security on an IIS 5.0 Server

Because FrontPage Server Extensions has no security of its own, you have to set up IIS 5.0 security to work with FrontPage. If IIS 5.0 isn't configured correctly, someone without authorization could edit a FrontPage web without being challenged for a user name and password.

IIS 5.0 uses Windows NTFS permissions in order to enforce security. Because FrontPage borrows the security mechanism of the Web server it is extending, you have to configure IIS 5.0 security in conjunction with NTFS permissions. For information about how IIS 5.0 authenticates a user, see "Security" in this book.

Resecuring a Site

This section tells you how to restore security, if you've determined that unauthorized users are able to log on to a Web site by using the IUSR_*computername*. Without examining a specific problem and configuration, it's hard to make generalizations. However, the following troubleshooting procedure suggests a way to tighten security on a site.

To resecure a site

1. If your server is configured with nested content or overlapping virtual servers, fix those problems first.
2. If the IUSR_*computername* account is a member of the **Administrators** group, remove it from the group. No anonymous user should have administrator privileges.
3. In FrontPage, on the **Tools** menu, select **Security** and then **Permissions**.
4. On the **Permissions** property sheet, if the **Everyone** group is listed, remove it, and click **OK**.
5. Reopen the **Permissions** property sheet, and on the **Users** tab, select **Everyone has browse access**.

   The IUSR_*computername* account is a member of the **Everyone** group, and should not be a member of any group that has **Author** access. If **Permissions** on the **Tools** menu is grayed out, you are on a file allocation table (FAT) partition. You must store Web content on an NTFS partition in order to have FrontPage security.

   If the IUSR_*computername* account can still open the Web site, see step 6.
6. For the root Web site and all its sub-webs that use unique permissions, set **Only registered users have browse access** on the **Permissions** property sheet.

   Similar to a directory hierarchy, a sub-web is a site under a main Web site.
7. To tighten security, open the IIS snap-in, right-click the virtual server, click **Tasks**, click **Check Server Extensions**, and select **Yes**.

8. Open the FrontPage web. (If you receive an error message, give the IUSR_*computername* account Read, Write, and Delete permissions to the _vti_pvt directory of the web in question.)
9. Set the site back to **Everyone has browse access**. Exit the FrontPage Explorer, and retest the security of the web in FrontPage.
10. If you've tried each of the previous steps to no avail, or if you get unexpected results when managing content with FrontPage, reset NTFS permissions back to the default, and let FrontPage retake control.

    This step applies to a single sub-web site, or to an entire virtual server.

Setting Permissions

Unless you understand how the FrontPage security model works, it's best to let FrontPage manage Web permissions for you. This section gives several examples of improperly set permissions and tells you how to correct them.

For a detailed description of the FrontPage security model on IIS 5.0 and guidelines for the minimum permissions needed, see the "Security on Windows NT" section of the *FrontPage 2000 Server Extensions Resource Kit*.

Common Problems and Resolutions

Problems with permissions can range from a hit counter that does not increment (or an author who cannot open a web) to issues that deal with multiple virtual servers. The following is a list of common problems and resolutions:

- The hit counter on a Web page shows up as a broken image or is stuck at 1.

  Make sure the file _private/Filename.htm.cnt has Read, Write, and Delete permissions for the user (usually the IUSR_*computername* account) accessing the page. Before trying to edit the permissions directly, run the FrontPage command **Check Server Extensions**. This will reset the permissions on the files in the _private directory.

  ### To run this command

  1. On the **Windows Taskbar**, select **Start**, point to **Programs**, **Administrative Tools**, and **Internet Services Manager**.
  2. Right-click the virtual server that's having the problem, point to **All Tasks**, and select **Check Server Extensions**.

  If the command fails to resolve the problem, then IUSR_*computername*, Network, Interactive, or Everyone is more than likely listed on the **Permissions** property sheet, (which you can access from the **Tools** menu in FrontPage). If that is the case, follow the steps in "Resetting Permissions in a Sub-Web" later in this chapter. The same solution applies to problems with forms.

- When a user tries to submit a form, FrontPage returns an error message in the browser, or asks for a user name and password.

  Make sure that the resulting storage file has Read, Write, and Delete permissions for the user (usually the IUSR_ *computername* account) submitting the form. You can find the name of this file by looking in the source HTML of the form, or by checking the **Form** properties in the FrontPage Editor.

- The Web root is busy or cannot open Service.lck for Read/Write permissions.

  Lock files (*.lck) ensure that FrontPage has exclusive access to files as necessary. When a Web page is opened, the main lock file, _vti_pvt/service.lck, must be accessible. If there are insufficient permissions for the IUSR_*computername* account or if the file is corrupt, an error message is returned. To correct this problem, give Read, Write, Execute, and Delete permissions to all FrontPage users, for the _vti_pvt directory.

  Unless the web is restricted, the IUSR_*computername* account will be a FrontPage user. To resolve this, turn off anonymous browsing in IIS 5.0, and reset the permissions, as described later in this chapter. When you have finished resetting permissions, turn anonymous browsing back on.

- An author cannot open the web to add or edit content, but a Windows administrator can.

  Make sure that the author has been added to the FrontPage web as an author and that the content is correctly structured, as described earlier in this chapter. If Basic authentication has been set, the author must have the right to log on locally and might have to enter *domainname\username*, rather than just *username* when prompted.

- Anyone can open the FrontPage web without being asked for a user name and password.

  To solve this problem, see the suggestion for tightening security on a site earlier in this chapter.

- When trying to open, rename, or save a file to a web in the FrontPage Explorer or Editor, one of three errors is returned. For example, suppose you receive one of the following errors for a file called "Test.htm":

  - Cannot open file "Test.htm"
  - Cannot rename file "Test.htm"
  - Cannot save file "Test.htm"

  To eliminate these errors, if you've selected integrated Windows authentication, make sure you're not authoring as the anonymous user.

Also, examine the NTFS permissions on the file that is giving these error messages, or on the parent directory. The user you are authenticating to the FrontPage web must have sufficient permissions on the file, such as Read, Write, and Delete. If the permissions are incorrect, adjust them manually. However, if there are several thousand files in the web with incorrect permissions, correcting each one individually will take too long. A more efficient alternative is to remove the author in question from FrontPage permissions. You can change permissions on the **Permissions** property page, which is accessed by selecting **Permissions** from the FrontPage **Tools** menu. Another method is to run the Fpsrvadm.exe command-line utility and reinstate the author with FrontPage permissions.

> **Note** If you have to change permissions on thousands of files, security changes may take several minutes to propagate.

- You are unable to tighten security on a particular file or directory in a FrontPage web.

  By default, FrontPage does not allow you to increase security on just one file or directory. You should create a new sub-web with unique permissions and then set the desired permissions through FrontPage.

  Alternatively, you can target one file or directory by changing NTFS permissions. First, create a virtual directory that is physically outside the FrontPage content area. Next, manipulate NTFS permissions on this directory, or on a file within it. There is no danger of FrontPage overwriting NTFS permissions. The only drawback to this method is that FrontPage will not recognize that virtual directory and, therefore, will not be able to manage it.

Granting Log on Locally Rights

This section tells you how to grant users rights to Log on locally, as well as how to verify that a user has these rights.

To grant Log on locally rights

1. On the Windows 2000 Desktop, right-click the **My Computer** icon.
2. Select **Manage**.
3. In the left-hand pane of the Computer Management tool, double-click **System Tools**.
4. Double-click **Group Policy**, then **Computer Configuration**, then **Windows Settings**, **Security Settings**, **Local Policies** and, finally, **User Rights Assignments**. Figure 8.4 shows the user interface.

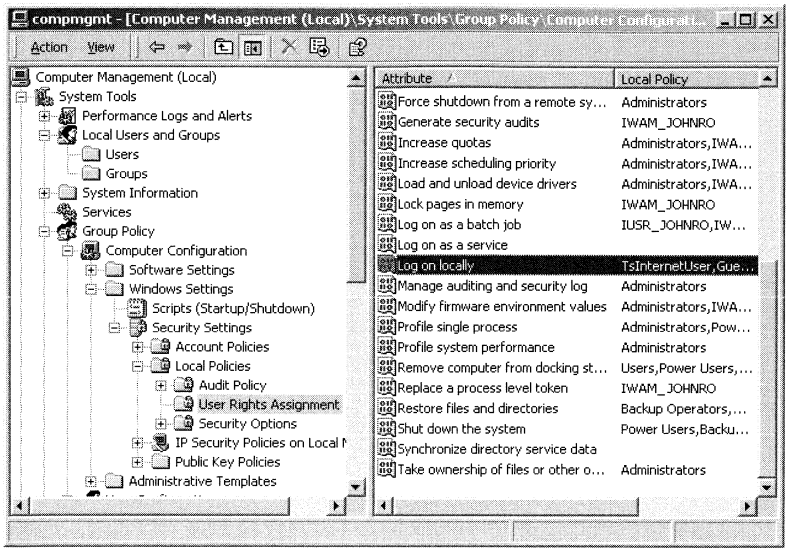


Figure 8.4 Computer Management, User Rights

5. In the right-hand pane of the Computer Management tool, double click **Log on locally**.

   Figure 8.5 shows the resulting Log on locally dialog box, where you'll see a list of users for that computer. You'll also see a box to the right of the user name, under the heading **Local Policy**.

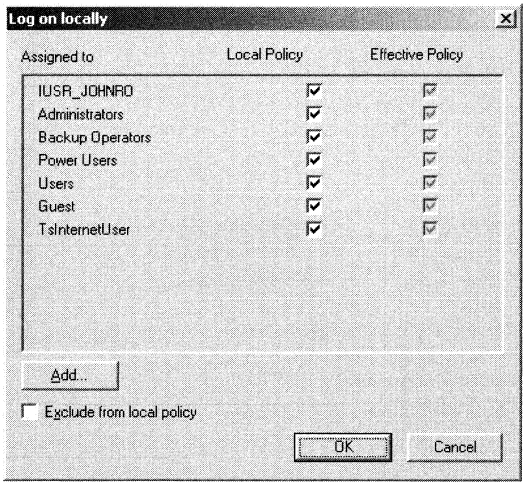


Figure 8.5 Log on Locally Dialog Box

6. Select the **Local Policy** box for the user, or verify that the box has been selected.

Resetting Permissions in a Sub-Web

The easiest way to fix permissions problems in FrontPage is through the client. This section gives two sets of procedures for you to correct these problems. The first tells you what to do if a user, who has permission, cannot access a web. The second tells you how the **Check Server Extensions** command can fix authoring problems.

Cannot Access a Web Site

These steps will ensure that the user can access a web.

To give access

1. If you get errors relating to the Service.lck file, turn off anonymous browsing on the Web server.
2. Log on as an administrator, and open the web that is having the problem.
3. Remove the user account that is receiving the error, and click **Apply**.
4. If the IUSR_*computername* account is listed under the **Users** tab, remove it.
5. Click the **Groups** tab. If Network, Interactive, or Everyone is listed, remove them, and click **Apply**.
6. Under the **Users** tab, set browse access to **Only registered users have browse access**, and click **Apply**.
7. After the settings have been applied, reset browse access to **Everyone has browse access**.
8. Add the problem account back in with author access, and click **Apply**.
9. If you had to disable anonymous browsing, enable it.

If the user is not able to write to a Web site, try the next procedure.

Cannot Perform Authoring Tasks

Occasionally, an author of a web might not be able to upload web content, modify the web directly on the server computer, or perform other authoring functions supported by FrontPage Server Extensions. The cause of the problem can range from accidentally deleting or renaming essential files to corrupting the Windows 2000 DACLs. The **Check Server Extensions** command addresses many authoring problems by doing the following:

- Checks that files have Read access permission.
- Checks that the files Service.cnf and Service.lck have Read and Write access.
- Updates the files Postinfo.html and _vti_inf.htm.
- Verifies that the files _vti_pvt, _vti_log, and _vti_bin are installed and that _vti_bin is executable.
- Determines whether virtual server roots or metabase settings are correct and up-to-date.
- Verifies that the IUSR_*computername* account doesn't have Write access.
- Warns you if you are running on a FAT file system. If you are, you won't be able to supply any Web security whatsoever (provided you're running IIS 5.0).

To check server extensions on a web

1. In the console tree, right-click the web that you want to check, and fix the server extensions.

   The selected web will be checked and fixed, as well as all sub-webs below it.

2. Click **Tasks** on the shortcut menu, and then click **Check Server Extensions**.

   The **Check Web** window displays a status log for each web that is checked.

3. Select **Yes** if you see the following prompt: **Do you want to tighten security as much as possible?**

Resetting Permissions on a Virtual Server

This section contains two sets of procedures. You can use them to fix permissions on an entire server, without losing any custom permissions that have been set in each sub-web.

Before you work through these procedures, make sure that there are no custom script files in the directory structure you are working on. Otherwise, you may need to reset the permissions on these files after you have finished.

Note While you work through these steps, users will get a password prompt when trying to browse webs until permissions have been reset.

One Virtual Server

The following procedure shows how to reset permissions for a configuration of a single virtual server with only a few sub-webs. If you have many sub-webs, work through the procedure that follows this one.

To reset permissions

1. Open a sub-web.
2. If the IUSR_*computername* account is listed under the **Users** tab, remove it. Also remove the Network, Interactive, and Everyone groups if they are listed under the **Groups** tab.
3. Click **Apply**.
4. Set browse access to **Only registered users have browse access**, and click **Apply**.
5. Verify that none of the users or groups you just removed has reappeared under the **Users** or **Groups** tabs. If they have, remove them.
6. Repeat the previous steps for each sub-web on the virtual server.

   Save the root web for last, in order to avoid random password prompts.
7. From the IIS snap-in, right-click the virtual server for which you want to change permissions, select **All Tasks**, and then click **Check Server Extensions**.
8. Select **Yes** when you see the following prompt: **Do you want to tighten security as much as possible?**
9. Open the root web with FrontPage.

   If you get an error message such as **Cannot open service.lck for writing**, turn off anonymous browsing on the Web server. (You can turn it back on once you have opened the root web.)
10. From the **Tools** menu in FrontPage, click **Permissions**, and on the **Permissions** property sheet, select **Everyone has browse access**, and click **Apply**.
11. Repeat steps 9 and 10 for each sub-web.

Multiple Virtual Servers

The following procedure shows how to reset permissions for multiple virtual servers, each with multiple sub-webs that have unique permissions. First, you have to restrict browser access, and then reset permissions on each virtual server.

To restrict browser access

1. From the document root (which contains the virtual servers on which you need to reset permissions), open a command prompt.

   If you are dealing with the Default Web Site (found in the C:\Inetpub\Wwwroot directory, by default), open a command prompt and change to the C:\Inetpub directory. Then, type the following command:

   ```
   cacls wwwroot /T /E /R iusr_computername network interactive everyone
   ```

   However, the Wwwroot directory is just an example. Substitute your content directory's path in the preceding command.

2. From the IIS snap-in, right-click the virtual server on which you want to change permissions, select **All Tasks**, and then click **Check Server Extensions**.

3. Select **Yes** when you see the following prompt: **Do you want to tighten security as much as possible?**

   At this point, Browse access will be restricted.

Once you are in the site with FrontPage, turn Anonymous access on.

To enable Anonymous access

1. Open the root web with FrontPage.
2. From the **Tools** menu in FrontPage, click **Security** and then **Permissions**.
3. On the **Users** tab of the **Permissions** property sheet, select **Everyone has browse access**, and click **Apply**.
4. Repeat the previous step for each sub-web that was set with unique permissions.

How These Solutions Work

The majority of the problems that FrontPage encounters are caused by changes to NTFS permissions. Normally, IUSR_*computername* is granted Read and Execute permission from the top of the document tree. When IIS 5.0 finds that IUSR_*computername* can execute Admin.dll and Autor.dll, it never prompts FrontPage for a password. From this point, one of two things will happen:

- Everything will appear to work, but all documents are created by the anonymous user. This occurs only if IUSR_*computername* was given Change or Full Control permission in the document tree.
- When you try to save a file, FrontPage returns an error message saying it cannot write to a given file, or FrontPage corrupts the document (caused by problems in the _borders and _derived directories).

The previous solutions remove problem accounts either through FrontPage or the **Cacls** command. Using the Windows Explorer to reset permissions replaces all other permissions with the new ones. Once **Cacls** or FrontPage have done their work, the FrontPage Server Administrator verifies the results.

After making the appropriate change, reactivate anonymous browsing through the FrontPage client, so that FrontPage will put IUSR_*computername* only where it has to be, and with the tightest possible security.

Configuring E-mail

This section offers solutions to two common problems that can occur when you try to submit a form through FrontPage. These problems are illustrated in the following scenarios:

Scenario 1 Errors appear in a client's browser when a form is sent through e-mail.
Scenario 2 After filling out a form, the client receives a confirmation, but the form isn't sent.

Scenario 1

Suppose a client publishes a form on the Web server. Although the form is designed to send e-mail results, when it is submitted a FrontPage error appears in the Web browser and no e-mail is sent. Or, suppose a client is authoring live on the server and attempts to e-mail a form. The client receives the following error message, instead of the form:

```
The FrontPage Server Extensions have not been configured to send e-mail.
Please direct your system administrator or Internet Service Provider
to follow the instructions at "Setting Up Your E-mail Transport" in the
\SERK\enu\admin.htm file on the FrontPage CD-ROM.
Would you like to remove the e-mail recipient? Yes/No
```

Solution

To keep your clients from receiving an error message, make sure you have configured mail settings in the IIS snap-in.

Some features, such as an e-mail form handler, require that e-mail be sent to visitors who browse a web. To allow these features to send e-mail, you must specify the following in the IIS snap-in:

- Web server's mail address
- Contact address
- Simple Mail Transfer Protocol (SMTP) host
- Mail-encoding scheme
- Mail character set

To configure mail settings

1. In the IIS snap-in console tree, right-click the Web site for which you want to set e-mail options.
2. Click **Properties** on the shortcut menu, and then click the **Server Extensions** tab.
3. In the **Options** box, click **Settings**.
4. Enter the appropriate settings:
   - **Web Server's Mail Address** Type the e-mail address that you want to appear in the **From** line of any e-mail messages that FrontPage components send. (For example, the form results component sends e-mail when someone fills out a form on a Web page and sends the information to the author of the form.) This setting is equivalent to the MailSender setting in Microsoft® FrontPage® 98, but is now stored in the registry, instead of in the Frontpg.ini file.
   - **Contact Address** Type the e-mail address that users should write to if they have problems. This address will appear on any error messages that are displayed, should FrontPage users encounter problems.
   - **SMTP Host** Type the name of the SMTP server that will deliver all mail sent from the web, or accept the default value, port 25. When a user submits a form whose results are to be sent through e-mail, the FrontPage Server Extensions connect to the SMTP server, in order to deliver the mail. By default, FrontPage assumes that the server is listening on port 25 (the standard for SMTP), but you can override this port by appending ":nn" to the name and IP address. (The *nn* is the number of the new port.) This setting is equivalent to the SMTPHOST setting in FrontPage 98, but is now stored in the registry instead of in the Frontpg.ini file.

     For example:

     reskit.microsoft.com (standard mail server name)

     - 172.16.29.38 (mail server's IP address)

     reskit.microsoft.com:31 (standard mail server name on different port)

     - 172.97.31.87 (mail server's IP address on different port)

- **Mail-Encoding Scheme** Select the mail-encoding scheme you want, or accept the default. This scheme encodes the contents of your mail messages in binary format. You should select default encoding if you want to allow FrontPage to detect the correct encoding for you, which it will, in most cases. However, you should change the encoding, if you know that the receiver of your e-mail has a different encoding setting from the default.
- **Mail Character Set** Select the appropriate character set to be used in e-mail messages, or accept the default. Each character set is the alphabet or character set of a different language. Any given byte does not necessarily map to a fixed character, since not all machines necessarily use the same character sets. When you accept the default setting, FrontPage automatically detects the character set for you.

Scenario 2

The second common problem can occur when a form appears to be sent and the person filling out the form receives a confirmation page, but the form never reaches the target address.

Solution

A filter at the local or remote mail server could be deleting junk mail. Many mail servers check that the **From** address of e-mail they process is valid. By default, FrontPage server extensions send e-mail indicating that the user has submitted the form. So, on an IIS 5.0 server the e-mail might appear to come from IUSR@*webserver* or, on a Netscape Server, from system@*webserver*. You can solve this problem by verifying that the mail server accepts the Web server's mail address specified in the IIS snap-in. If it does not, change the account setting to one that the mail server is configured to accept. See the previous scenario for instructions about configuring this setting.

If mail still doesn't reach the server, see if mail sent from another e-mail client on the same server is received.

Uploading Executable Scripts or Programs

When a web author wants to include a CGI script or a script in an ASP page within a web, make sure that the author isn't inadvertently uploading a virus or a program that has bugs onto your Web server. You can avoid this risk by preventing an author from uploading scripts and programs to any folder that is part of a web.

To control the uploading of executables

1. In the IIS snap-in, right-click the root web for the appropriate folders.
2. On the menu, click **Properties**, and then click the **Server Extensions** tab.
3. To prevent the folder from containing executable scripts or programs, make sure that the **Allow authors to upload executables** check box is cleared. To allow the folder to contain executable scripts or programs, make sure that the **Allow authors to upload executables** check box is selected.

Because the default setting prevents authors from uploading executables, you may find that your authors get an error message similar to the following:

```
Server error: The folder "/Cgi-bin" is marked executable. You are not allowed
to put files into an executable folder on this server.
```

Note In this example error message, the Cgi-bin directory can be any folder with Executable permission on the server.

This error will occur if the author tries to upload files from an executable folder that is on a client computer to an executable folder that is on the server. When this happens, ask the author to let you test the files before publishing them. If you are confident that the executables are safe, allow the author to upload these files by following the steps that were just listed.

Uploading Content through FTP

Once content for sites is created, you need to upload it onto a server for publication. The advantage to uploading content through FTP is speed.

FTP, which is integrated into the Microsoft® Windows® operating system as an Internet service, publishes information on a Web server through a standard FTP client. Depending on the client, you can operate FTP through the command-line or through a GUI-based interface.

Although FTP is efficient at uploading, it has two drawbacks:

- **Minimal Support for Querying and Retrieving Information on the Server** Although searching was not part of the original purpose of FTP, you can set up a filter on the client (or server, for that matter). This filter can return result sets for specific information. While this solution might seem a bit cumbersome, it is a viable one.
- **Lack of Strict Security** FTP connections are not secure, because passwords are exchanged as clear text. At this time, extensions are being developed that will create a secure way of authenticating clients and transmitting files. Of course, sensitive information should never be sent as clear text because it can be intercepted, modified, and replayed. When these new extensions become available, FTP will be a secure solution to the current problem.

Internet Connection Services for Remote Access

The components that previously made up Internet Connection Services for remote access are now components of Windows 2000 Server. They are no longer referred to as Internet Connection Services for remote access, but by their individual component names:

- Connection Manager Administration Kit
- Connection Point Services
- Internet Authentication Services

These connection components are designed to help corporations and ISPs build comprehensive Internet-access solutions, including dial-up VPNs. Whether you are building an Internet service or managing a corporate network, these connection components help you quickly set up, customize, and manage a remote-access network. You can give your subscribers a seamless connection to the Internet, a global dial-up service, as well as secure connections over the Internet to a private network.

Whether you're an ISP or a corporate network administrator, the steps for installing these components are the same. If you installed Internet Connection Services for remote access as part of IIS 4.0, you can upgrade to the latest versions of the individual components through Windows Setup.

To install the three components during Windows Setup

1. On the Windows **Optional Components** screen, click **Networking Options**, and then click **Details**.
2. Select the check box for the component you want to install.

If you have already installed Windows, you can install the components by using **Add/Remove Programs** in Control Panel.

To add the components after installing Windows

1. Click **Start**, point to **Settings**, and then point to **Control Panel**.
2. Click **Add/Remove Programs**.

For details about how to use these connection components, see the Windows 2000 Server online product documentation.

Administering Older Version Servers

This section tells you how to use the IIS snap-in to administer servers running previous versions of IIS. (Previous versions are also referred to as *downlevel* versions.)

If, for example, you have five Web servers running IIS 3.0 and you initially upgrade one of them to IIS 5.0, you can administer all of them from the IIS snap-in.

To administer older version Web servers

1. Create a temporary folder.

   This folder lets you rename the files that you copy in step 2, so that you don't overwrite existing files (that currently live in your C:\WINNT\System32\Inetsrv directory) with the same name.

2. From the Windows 2000 Server CD, copy the following files into the temporary folder you have created:

File Name	Directory Location
Fscfg.dll W3scfg.dll Gscfg.dll	\I386\inetsrv\
Fscfg.hlp W3scfg.hlp Gscfg.hlp	\I386\inetsrv\help\

3. Rename the files in the temporary folder. For example:

Old Name	New Name
Fscfg.dll	Fscfg3.dll
W3scfg.dll	W3scfg3.dll
Gscfg.dll	Gscfg3.dll
Fscfg.hlp	Fscfg3.hlp
W3scfg.hlp	W3scfg3.hlp
Gscfg.hlp	Gscfg3.hlp

   The Gopher DLL (Gscfg.dll) does not have to be renamed, but this has been done for consistency.

4. Copy the renamed files into your C:\WINNT\System32\Inetsrv directory.

5. Add the following registry values (with data type REG_SZ) to
   HKEY_LOCAL_MACHINE
   \Software
   \Microsoft
   \InetMgr
   \Parameters
   \AddOnServices

```
WWW3 =
C:\WINNT\System32\inetsrv\w3scfg3.dll::SUPCFG:C:\WINNT\system32\inetsrv\w3scfg.dll
FTP3 = C:\WINNT\System32\inetsrv\fscfg3.dll::SUPCFG:C:\WINNT\system32\inetsrv\fscfg.dll
GOPHER3 = C:\WINNT\System32\inetsrv\gscfg3.dll
```

   Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft® Management Console (MMC) whenever possible.

   This step appends "::SUPCFG:<newer dll>" to the existing DLL file name, indicating that the current files have been superseded.

6. From the IIS snap-in, connect to a server running IIS 3.0. To do this, open the IIS snap-in, select **Internet Information Services**, click **Action**, then click **Connect**. Enter the name of the Web server you want to connect to.

At this point, you can see the Web server from the IIS snap-in.

The **Properties** toolbar button in the IIS snap-in is not available for some older sites. Instead, you must right-click the name of the site and select **Properties**. You can also select **Properties** on the **Action** menu.

Figure 8.6 shows the IIS snap-in connected to a server (called "someone") that is running IIS 3.0.

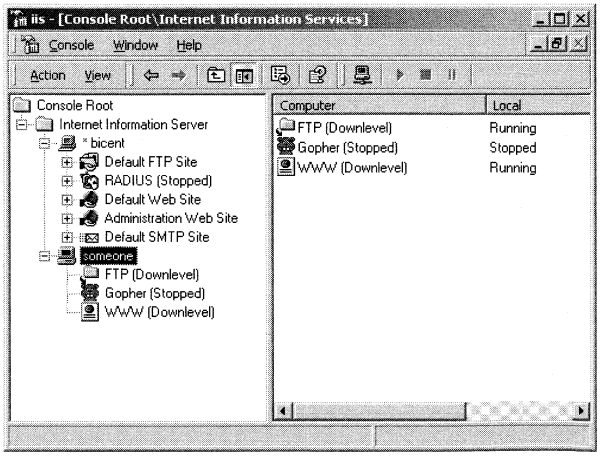


Figure 8.6 Remote Connection to an Older Version Server of IIS

Allocating Resources

This section discusses logging activity on sites. It gives examples of how to configure logging, in order to record information needed for managing your Web sites efficiently. It shows how to adjust bandwidth in order to improve performance on each server in your installation. This section also describes extended logging through process accounting and tells you how to throttle processes in order to prevent any one site from taking up too much processor time. Also discussed is the HTTP Monitoring Tool, which monitors the performance of Web sites.

Logging Resources

Historically, logging has been a complex and time-consuming task. Web servers had to collect information about activity on a server and put it into a single file (sometimes several hundred megabytes in size). ISPs were then forced to extract information about specific Web sites from this file.

IIS 5.0 simplifies logging. You can now set various levels of granularity when analyzing the activity on your server. Log files can include or exclude information about individual Web sites, individual directories, and individual files.

By using IIS 5.0 logging, you can track which users access a site and when, which helps to identify potential security and performance problems. You can direct logging either to a file that can be studied offline, or to an Open Database Connectivity (ODBC) Data Source Name (DSN) for online evaluation.

IIS 5.0 has several logging features to help customize information about an IIS 5.0 Web site:

- **Customized Extended Logging** IIS 5.0 supports the latest industry-standard W3C extended logging format, a customizable ASCII format that allows you to rerecord a variety of fields (items). You can gather detailed information and still limit log size by omitting unneeded fields. These fields include about 20 different items, such as date, time, client IP address, and browser type.
- **Resource Logging** As with most configuration settings in IIS 5.0, logging can be set on a per-file basis. With resource logging, you can limit logging to specific resources. This will improve performance, reduce the size of the log file, and make it easier to interpret log files. For example, to reduce the log file size, you could put all the image files in one directory and choose not to log the files in that directory.
- **COM Logging Interface** Developers can create custom modules to log information about a Web site. Each module is responsible for processing request events and writing either to a SQL Server data source or to a log file. IIS 5.0 logging capabilities can be extended by "plugging in" additional logging modules that developers or third-party vendors create.

To see the various levels of logging

1. Right-click the Web site for which you want to record a log.
2. On the **Web Site** property sheet, click the down arrow in the **Active log format** box.
3. Select the style of logging you want, and click the **Properties** button.
4. On the **General Properties** (or **ODBC Properties**) dialog box, select the desired granularity of logging.
5. If you've selected the **W3C Extended Log File Format**, select your preferred logging options in the **Extended Logging Options** dialog box.

For example, in the following log file, requests for files in the Images directory are included. If this page were to be requested several thousand times, the GET request for the image would be logged several thousand times as well.

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 1999-01-01 22:20:57
#Fields: time c-ip cs-method cs-uri-stem sc-status
22:20:57 172.16.1.1 GET /samplecorp/ 404
22:21:03 172.16.1.1 GET /Default.htm 304
22:21:37 172.16.1.1 GET /images/undercon.gif 304
```

The following example shows a log file that doesn't log the Images directory. Note that it doesn't include the GET request for /Images/Undercon.gif.

```
CLIENT REQUEST
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 1997-10-16 22:23:32
#Fields: time c-ip cs-method cs-uri-stem sc-status
22:23:32 172.16.1.1 GET /samplecorp/ 404
22:24:14 172.16.1.1 GET /Default.htm 304
```

To further customize logging, you can create other logging modules with Microsoft® Visual C++® or Visual Basic. For more information, see the IIS 5.0 online product documentation.

For information about how to set extended logging options, see the "Customizing W3C Extended Logging" topic in the IIS 5.0 online product documentation. For a list of logging options and other information, see the "Logging Properties Reference" topic in the IIS 5.0 online product documentation.

Controlling Bandwidth

One of the administrator's tasks is setting and monitoring bandwidth for various sites. As the bandwidth utilization of a Web site approaches the limit set by the administrator, response times slow down. If requests are received faster than the site can respond (meaning that the queue is full), the server will return an error message (500: Server Too Busy) to the client.

IIS 5.0 allows you to control the bandwidth of each Web site on a server, distributing it where needed most.

To throttle bandwidth on a server

1. In the IIS snap-in under **Internet Information Services**, right-click the Web site on which you want to throttle bandwidth.
2. Click **Properties**.
3. On the **Internet Information Services** property sheet, select **Enable Bandwidth Throttling**, and click **OK**.

By throttling (adjusting) bandwidth, an ISP can offer Web hosting solutions that are tailored to the bandwidth needs of individual customers. In IIS 5.0, you can throttle bandwidth for static content, including .htm, .jpg, and .gif files. However, you can distribute bandwidth on your server by throttling only the static HTML (files with .htm or .html extensions).

The metabase stores the configuration options for the bandwidth throttler, and IIS 5.0 is notified as soon as changes are made. You can manipulate the threshold values (bandwidth limits), and turn on or turn off bandwidth throttling, without stopping and restarting IIS 5.0.

Disabling Socket Pooling

IIS 5.0 sites that are bound to different IP addresses—but the same port number—share the same set of sockets. This feature allows more sites bound to an IP address to be created on the same machine than was possible in IIS 4.0. In IIS 5.0, these sockets are used flexibly among all of the sites, in order to reduce resource consumption.

Socket pooling is now the default for IIS 5.0, and in general, you don't need to change it. However, both bandwidth throttling and performance adjustments will apply to all Web sites configured for the same port. So, if you intend to enable bandwidth throttling or do performance tuning on a per-site basis, you will need to disable socket pooling.

- To disable socket pooling, type the following at the command prompt:

```
c:\inetpub\adminscripts\cscript adsutil.vbs set w3svc\disablesocketpooling true
```

The command prompt will reply:

```
disablesocketpooling : (BOOLEAN) True
```

Process Accounting and Process Throttling

In addition to throttling static content, you can throttle processes on Web sites. If one of your customers is consuming too many resources, to the point that other Web sites are slowed down or otherwise affected, you might want to restrict the amount of resources this customer can consume. To do so, you must first measure resource consumption, in order to see which site is consuming too much. Then, limit its amount of resources. IIS 5.0 comes with two new features to help you do this:

- Process accounting logs the CPU time and other resources used by applications.
- Process throttling limits the amount of resources a Web site can consume.

Process accounting and process throttling work only for applications run out of process. You cannot activate accounting for in-process applications, or for applications run in the new IIS 5.0 out-of-process pool.

You can track the CPU time for applications, log information, and put throttling limits on the site, if necessary. Generally, process accounting and process throttling apply to an entire site and cannot be targeted toward any individual application, unless only one application is running on the site. If you have more than one application running on the site, you can target a specific application by writing a script in the metabase, using the appropriate property:

CpuAppEnabled Enables process accounting and throttling for an out-of-process IIS 5.0 Web Application Manager (WAM) application.

CpuCgiEnabled Enables process accounting and throttling for a CGI application.

For details, see the "CpuAppEnabled" topic in the IIS 5.0 online product documentation. For information about activating process accounting and process throttling, see the "Tracking Processor Use" and "Throttling Processes" topics in the IIS 5.0 online product documentation.

Measuring Resource Consumption

To measure the amount of resources consumed by the sites in your installation, turn on process accounting. Once you know which site is using too many resources, you can decide whether to limit resources on that customer's Web site (through process throttling) or whether you should take other actions.

To turn on process accounting

1. On the site's property sheet, click the **Home Directory** tab.
2. Configure the **Application Settings** to **High (Isolated)**.

   These first two steps make all applications on the site run out of process.
3. On the site's property sheet, click the **Web Site** tab, and make sure **Enable Logging** is selected.
4. On the **Web Site** property sheet, click the logging **Properties** button, and select **Process Accounting**.

   The last two steps activate process accounting for that site.

With process accounting activated, you extend logging so that figures for the Job Object counters are recorded. With the information gathered from process accounting, you can then decide whether to upgrade the servers in your installation, to adjust the billing for this particular customer, or to limit the amount of resources the site can consume.

For more information about activating process accounting, see the "Tracking Processor Use" topic in the IIS 5.0 online product documentation.

Limiting Resources

After determining the amount of resources the customer's site is consuming, you might want to limit that customer to a certain percentage of your available resources. This will free up resources for other customers. To limit a site's resources, run the site's applications out of process, and then turn on process throttling.

To turn on process throttling

1. On the site's property sheet, click the **Performance** tab.
2. Select **Enable process throttling**.
3. In the **Maximum CPU use** box, set the percentage of CPU resources dedicated to the site.
4. Enable **Enforce limits**.

When the site reaches a predefined limit, it will take the defined action, such as reduce process priority, halt processes, or halt the site.

For information about process throttling, see the "Throttling Processes" topic in the IIS 5.0 online product documentation.

Stopping CGI Applications

You might discover that a customer has a CGI application that consumes CPU resources continuously. To keep this application from using CPU resources, you can turn on process throttling for the site (as described in the previous section), and then limit the total CPU time for each instance of the CGI application. When the true limit is reached (which is usually a maximum of 200 percent of the limit you set), the applications on that site are stopped.

For more information, see the "About Processor Utilization" topic in the IIS 5.0 online product documentation.

Monitoring Performance with the HTTP Monitoring Tool

The HTTP Monitoring Tool is a browser-based tool, available on the Resource Kit companion CD, that monitors the performance of Web sites in an installation. To use this tool, you must first install Microsoft® SQL Server 6.5 or later on a computer that can be accessed by computers that run the HTTP Monitoring Tool.

The tool's interface looks similar to the screen shown in Figure 8.7.

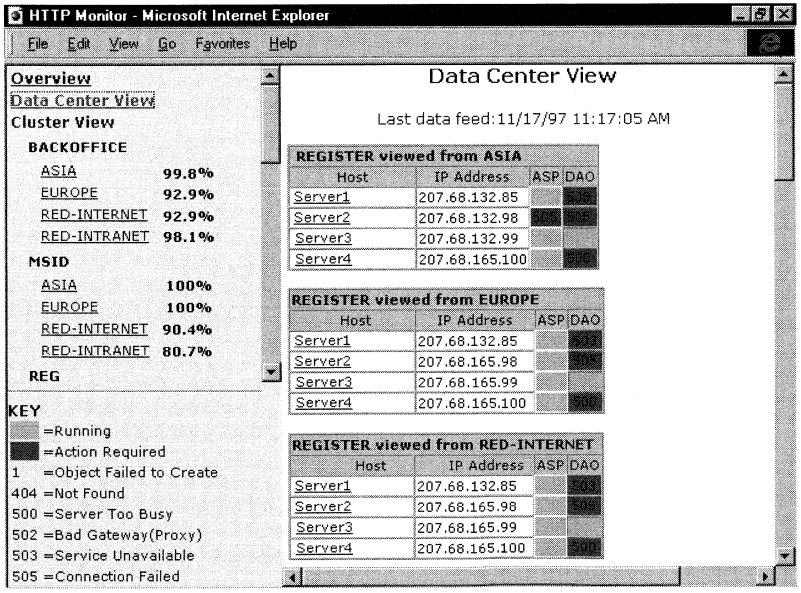


Figure 8.7 The HTTP Monitoring Tool's Real-Time Display in Internet Explorer 5

Using the HTTP Monitoring Tool

You can run the HTTP Monitoring Tool on several servers concurrently, in order to test connectivity and to display the connectivity statistics of the sites in your installation. The database for these Windows-based servers is centralized on a single SQL Server.

The HTTP Monitoring Tool, which emulates the way that users attempt to connect to Web sites, is a multithreaded process that operates according to the parameters and values set in its associated HTTPMon.ini file. These parameters specify the Web sites to be tested, as well as the number of times and frequency to retry a failed connection, before moving on to the next specified site. You can set other parameters that optimize the HTTP Monitoring Tool for your particular installation, including a maximum thread pool count, an output file and directory, a working directory, and so on.

Customizing Your Installation

In this section, you'll learn how IIS 5.0 can help simplify the task of managing multiple Web sites and content. You'll learn how to host multiple Web sites or domains on one IP address, by assigning host header names to each site. You'll also see how to bring older browsers into compliance with HTTP 1.1, so that they can support host headers. Since sites often change locations, you'll learn how to redirect requests to the new location. Next, you'll see how to control Web content that is held in a browser's cache, by using custom HTTP headers, as well as how to use custom HTML footers for all pages contained in a Web site. And finally, for a site that needs a footer in each document it publishes, you'll learn how to embed a link to an include file. This will make it easier to update the footer content throughout the site.

Hosting Multiple Sites with One IP Address

Many customers will not only request access to the Internet, but will also want to host their own domain (for example, domain.com) or subdomain on an existing server. To handle this type of situation, IIS 5.0 offers two models for site configuration:

- Multiple IP addresses on one server
- Multiple host headers on one server with one IP address

Previous versions of IIS allowed you to host multiple Web sites (virtual servers or domains) on a single computer running Windows NT Server 4.0. By assigning a different IP address to each site (all on one computer), you could reduce the number of computers required in an ISP installation.

With IIS 4.0, a new feature was introduced that allowed hosting multiple Web sites with a single IP address. Continued in IIS 5.0, this feature works well if you do not have a lot of IP addresses to spare, but want to host several Web sites on one computer anyway. Instead of assigning each site a unique IP address, you give it a unique host header name. This feature works best for smaller sites with less traffic than it does for sites on a computer with unique IP addresses for each site.

Host header names are the key to hosting multiple Web sites on one computer. However, as mentioned earlier, not all browsers support them. Because host headers are part of HTTP 1.1, browsers must comply with HTTP 1.1 in order to access sites that do not have unique IP addresses. Microsoft® Internet Explorer 3.0, Netscape Navigator 2.0, and later versions of both browsers support host header names, but earlier versions of the two browsers do not. For information about allowing older browsers to access a site through host header names, see "Supporting Non-HTTP 1.1–Compliant Browsers" later in this chapter.

The following examples show how to use host headers. The first example shows unique IP addresses assigned to multiple Web sites on one computer. The second example shows one IP address on one computer hosting multiple Web sites, differentiated only by their unique host header names.

One Computer, Multiple IP Addresses

IIS 3.0 offered this option as the sole way to publish multiple sites on the Internet. ISPs could direct customers to two or more separate IP addresses and have the addresses and the domain names registered with InterNIC. This option remains cost-effective for two reasons:

- ISPs can minimize their hardware costs.
- As a specific site grows, it can be migrated onto its own dedicated server without interrupting customer access.

Having multiple IP addresses enhances security of Web sites contained on the server. For example, in some cases an ISP might host customers' personal Web pages on one IP address. The ISP might then set up a second IP address, with IIS 5.0 and the FTP protocol enabled in order to let customers post content to their Web pages. This particular configuration will frustrate any would-be intruder, who might try to access the Web site. This intruder might try to capture FTP user names and passwords in two ways: either by attacking an FTP port on the site, or by performing a network "sniff" of the site.

The following example shows two Web sites (reskit1.microsoft.com and reskit2.microsoft.com) hosted on one computer with two different IP addresses (reskit1.microsoft.com = 172.21.13.45 and reskit2.microsoft.com = 192.168.114.201). Although you can see the host header field in the HTTP log, it is the network IP address (not the host header field) that differentiates between the Web sites.

This type of configuration works well for large installations that contain hundreds of Web sites and attract lots of hits.

```
CLIENT REQUEST FOR reskit1.microsoft.com
IP: Destination Address = 172.21.13.45
HTTP: Request Method = GET
HTTP: Uniform Resource Identifier = /
HTTP: Protocol Version = HTTP/1.1
HTTP: Accept = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.
HTTP: Accept-Language = en-us
HTTP: Accept-Encoding = gzip, deflate
HTTP: User-Agent = Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
HTTP: Host = reskit1.microsoft.com
HTTP: Connection = Keep-Alive

CLIENT REQUEST FOR reskit2.microsoft.com
IP: Destination Address = 192.168.114.201
HTTP: Request Method = GET
HTTP: Uniform Resource Identifier = /
HTTP: Protocol Version = HTTP/1.1
HTTP: Accept = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.
HTTP: Accept-Language = en-us
HTTP: Accept-Encoding = gzip, deflate
HTTP: User-Agent = Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
HTTP: Host = reskit2.microsoft.com
HTTP: Connection = Keep-Alive
```

One Computer, Same IP Address, Multiple Hosts

IIS 4.0 introduced host header names as a way to create many sites very quickly, without the need for IP addressing or subnet calculations. The advantage of host headers is that an Internet browser can access the IIS 5.0 Web server with a known URL (for example, reskit3.microsoft.com). The IIS 5.0 server will then map that URL path to an established location that matches each Web site's host header name. With this option, it is very easy (through the IIS snap-in) to create new sites, define their home pages, allocate bandwidth, and set specific access rights.

The following example shows two Web sites (reskit3.microsoft.com and reskit4.microsoft.com) hosted on one computer with the same IP address (172.21.13.45). In this case, the host header names differentiate between the sites.

```
CLIENT REQUEST FOR reskit3.microsoft.com
IP: Destination Address = 172.21.13.45
HTTP: Request Method = GET
HTTP: Uniform Resource Identifier = /
HTTP: Protocol Version = HTTP/1.1
HTTP: Accept = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.
HTTP: Accept-Language = en-us
HTTP: Accept-Encoding = gzip, deflate
HTTP: User-Agent = Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
HTTP: Host = reskit3.microsoft.com
HTTP: Connection = Keep-Alive

CLIENT REQUEST FOR reskit4.microsoft.com
IP: Destination Address = 172.21.13.45
HTTP: Request Method = GET
HTTP: Uniform Resource Identifier = /
HTTP: Protocol Version = HTTP/1.1
HTTP: Accept = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.
HTTP: Accept-Language = en-us
HTTP: Accept-Encoding = gzip, deflate
HTTP: User-Agent = Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
HTTP: Host = reskit4.microsoft.com
HTTP: Connection = Keep-Alive
```

For information about setting up host headers, see the "Naming Web Sites" topic in the IIS 5.0 online product documentation. For a general discussion about name resolution and IP addresses, see the "Name Resolution" topic in the IIS 5.0 online product documentation.

Supporting Non–HTTP 1.1–Compliant Browsers

Although many non-HTTP 1.1–compliant browsers support host headers, some do not, particularly older browser versions. To make your site available to browsers that do not comply with HTTP 1.1, you can send *cookies* from a Web site. These are objects that are sent to a Web server along with client requests. Another option is to embed a host name in the URL of the Web site (also called *URL munging*).

Sending Cookies

Making a site accessible to noncompliant browsers through cookies targets only those browsers that can accept them. Cookies can be attached to hosts on the client side, and will be sent with a request whenever a client accesses the Web site in question. An older browser that supports cookies can use a cookie as a pseudo host header. To allow this, the administrator must set up several components, including registry settings, and add .asp files to the Scripts directory of the Web site.

When a noncompliant browser first accesses a site with multiple hosts, the server cannot detect which site the user wants to visit. With IIS 5.0, you can display a custom menu to these users, listing the Web sites available. This menu can be any document—an .htm file or an .asp file. There are two possible host menu documents that the administrator can set up: one for clients that support HTTP cookies, and the other for clients that do not. The menu documents must exist in a virtual directory of one of the sites on the Web server. The administrator must specify the document names and the host name of the site where these documents can be found. For example, the following process shows what happens if a client tries to access reskit1.microsoft.com through a noncompliant browser:

1. The client attempts to access http://reskit1.microsoft.com.
2. Since there is no host header with the request, the server does not know which Web site the client wants.
3. IIS 5.0 presents the client with an HTML menu prepared by the administrator, which lists the available sites on this particular IP address.
4. The client selects (and is redirected to) the desired site on the server.
5. A cookie is returned to the client. This cookie ensures that all subsequent requests for http://reskit1.microsoft.com directly access the appropriate documents. There is no need for a menu, at this point.

In the following example, a request was made for the reskit1.microsoft.com and reskit2.microsoft.com sites. Notice the HTTP 1.0 GET requests and the host lines below them. While initiating an HTTP 1.0 request, Microsoft® Internet Explorer 3.02 can communicate with IIS 5.0 through a host header.

```
CLIENT REQUEST FOR http://reskit1.microsoft.com (from an HTTP/1.0 browser)
GET / HTTP/1.0
Accept: application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint,
image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en
UA-pixels: 800x600
UA-color: color16
UA-OS: Windows 2000
UA-CPU: x86
User-Agent: Mozilla/2.0 (compatible; MSIE 3.02; Windows 2000)
Host: reskit1.microsoft.com
Connection: Keep-Alive

CLIENT REQUEST FOR http://reskit2.microsoft.com (from an HTTP/1.0 browser)
GET / HTTP/1.0
Accept: application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint,
image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en
UA-pixels: 800x600
UA-color: color16
UA-OS: Windows 2000
UA-CPU: x86
User-Agent: Mozilla/2.0 (compatible; MSIE 3.02; Windows 2000)
Host: reskit2.microsoft.com
Connection: Keep-Alive
```

In the example that follows, notice that the GET request does not have a host header. Clients that do not support host headers, such as Microsoft® Internet Explorer 2.0, will return the error message "404 Object Not Found."

```
CLIENT REQUEST FOR http://reskit1.microsoft.com (from Internet Explorer 2.0)
GET / HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: en
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0d; Windows 2000)
Connection: Keep-Alive

SERVER RESPONSE
HTTP/1.1 404 Object Not Found
Server: Microsoft-IIS/5.0
Date: Wed, 14 Oct 1998 17:05:55 GMT
Content-Type: text/html
Content-Length: 102
```

Components for Administration (Sample Setup)

The following registry configuration is necessary to make the next example work. All of the host menu documents have been placed on the Web site http://reskit1.microsoft.com. Registry values have been added to

HKEY_LOCAL_MACHINE
\System
\CurrentControlSet
\Services
\W3SVC
\Parameters

Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft Management Console (MMC) whenever possible.

The registry values include:

DLCMenuString = /HostMenu (Data Type is REG_SZ)
This parameter specifies the prefix of URLs that apply to the host menu. The server will check against this string on all requests from older browsers (in other words, requests without a host header).

DLCHostNameString = reskit1.microsoft.com
This parameter supplies the host name of the instance that contains the menu documents.

DLCMungeMenuDocumentString = /scripts/munge.asp
This host menu document will be sent to older browsers that don't support cookies.

DLCCookieMenuDocumentString = /scripts/cookie.asp
This host menu document will be sent to older browsers that do support cookies.

DLCCookieNameString = PseudoHost
This parameter specifies the name of the special cookie that will act as a pseudo host header. This can be used with clients that support cookies.

DLCSupport = 1 (Data Type is REG_DWORD)
This parameter enables support for older browsers.

The host menu documents include three files (Munge.asp, Cookie.asp, and Redirect.asp).

Contents of reskit1.microsoft.com/scripts/munge.asp

Copy this sample code into a text file and name it Munge.asp. Then copy the file into the Scripts directory of the instance you are using. In this example, it is reskit1.microsoft.com.

```
<HTML>
  <HEAD>
    <TITLE>Server Selection Page</TITLE>
  </HEAD>
  <BODY>

    <A HREF="http://reskit1.microsoft.com/*reskit1.microsoft.com/<%=
    Request.QueryString() %>">Try this Site reskit1.microsoft.com</A><BR>
    <A HREF="http://reskit1.microsoft.com/*reskit2.microsoft.com/<%=
    Request.QueryString() %>">Try this Site reskit2.microsoft.com</A><BR>

  </BODY>
</HTML>
```

Contents of reskit1.microsoft.com/scripts/cookie.asp

Copy this sample code into a text file and name it Cookie.asp. Then copy the file into the Scripts directory of the instance you are using. In this example, it is reskit1.microsoft.com.

```
<HTML>
  <HEAD>
    <TITLE>Server Selection Page</TITLE>
  </HEAD>
  <BODY>

    <A HREF="/HostMenu/Scripts/Redirect.asp?Host=reskit1.microsoft.com&NewLocation=<%=
    Request.QueryString() %>">Try this Site reskit1.microsoft.com</A><BR>
    <A HREF="/HostMenu/Scripts/Redirect.asp?Host=reskit2.microsoft.com&NewLocation=<%=
    Request.QueryString() %>">Try this Site reskit1.microsoft.com</A><BR>

  </BODY>
</HTML>
```

Contents of reskit1.microsoft.com/scripts/redirect.asp

Copy this sample code into a text file and name it Redirect.asp. Then copy the file into the Scripts directory of the instance you are using. In this example, it is reskit1.microsoft.com.

```
<%

  Option Explicit

  Dim DLCCookieNameString

  DLCCookieNameString = "PseudoHost"

  Response.Cookies(DLCCookieNameString) = Request.QueryString("Host")
  Response.Cookies(DLCCookieNameString).Domain = Request.QueryString("Host")
  Response.Cookies(DLCCookieNameString).Path = "/"

  Response.Redirect "http://" & Request.QueryString("Host") &
  Request.QueryString("NewLocation")

%>
```

Munging URLs to Support Older Browsers

Another way to support older browsers is to embed the host header name in the URLs that are being sent to the Web server. On the server side, these embedded hosts are recognized and stripped out of the URL and used as a pseudo host header, prior to regular URL processing. For URL munging to work, each request from the browser must have a host header embedded. If it does not (as is the case with an initial request), a host menu will be sent to the client. For an example of URL munging, see the ISAPI filter *Cookie Munger* provided on the Resource Kit companion CD.

Making Redirects Work for You

With the redirection feature of IIS 5.0, you can divert requests for one Web page to another. In an intranet environment, for example, many sites could point to a common resource such as a Human Resources Web site. If the location of that site were changed for some reason, all of the sites pointing to that page would also have to change, to reflect the new URL. Finding all links to the Human Resources site and then changing them can be an overwhelming project in a large installation. However, IIS 5.0 supplies an easier way.

You can use redirection in order to point the old URL to the new location. With redirection, the pages pointing to the old location will continue to work, giving you the chance to update the links in stages, as time permits.

See the IIS 5.0 online product documentation for information about how to configure redirection.

Setting Up Custom HTTP Headers

With custom HTTP headers, you can control various aspects of the Web page that is sent to clients. When you set an HTTP header, information is embedded with the page header and sent through the browser, allowing you to:

- Limit the time a page is stored in the client's cache.
- Extend the current HTML specification.
- Rate the content of Web pages.
- Determine the file types sent to the client.

Content Expiration Example

Enabling the content expiration feature works well for sites that publish time-sensitive information, such as announcements for events or special offers. By sending clients a custom HTTP header to delete the page from the client's cache after a certain period of time, you can be sure that the client won't access a cached version of a page that is out of date. For information about configuring custom headers, see the "Enabling Content Expiration" topic in the IIS 5.0 online product documentation.

Compare the following two examples, in which the client has made a request to a computer named http://bicent/samplecorp.

Without a Custom HTTP Header

This sample code shows a request that is sent without an HTTP header:

```
CLIENT REQUEST
GET /samplecorp/images/undercon.gif HTTP/1.1
Accept: */*
Referer: http://bicent/samplecorp/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
Host: bicent
Connection: Keep-Alive
Cookie: ASPSESSIONIDGQQGQGBE=OGIFFNMBNOHDMKLFECBOKFPM

SERVER RESPONSE
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 14 Oct 1999 17:49:38 GMT
Content-Type: image/gif
Accept-Ranges: bytes
Last-Modified: Tue, 14 Oct 1999 15:53:27 GMT
ETag: "0d1974fb9d8bc1:eca"
Content-Length: 293
```

With a Custom HTTP Header

This sample code shows the same request sent with an HTTP header that causes content to expire in 30 minutes:

```
CLIENT REQUEST
GET /samplecorp/images/undercon.gif HTTP/1.1
Accept: */*
Referer: http://bicent/samplecorp/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
Host: bicent
Connection: Keep-Alive
Cookie: ASPSESSIONIDGQQGQGBE=PGIFFNMBCAOPEHGGKOGJODID

SERVER RESPONSE
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Cache-Control: max-age= 1800
Expires: Tue, 14 Oct 1999 18:24:21 GMT
Date: Tue, 14 Oct 1999 17:54:21 GMT
Content-Type: image/gif
Accept-Ranges: bytes
Last-Modified: Tue, 14 Oct 1999 15:53:27 GMT
ETag: "0d1974fb9d8bc1:ecb"
Content-Length: 293
```

Notice the two lines enabling content expiration. While Cache-Control affects the proxy cache, Expires affects the browser cache.

You can set custom HTTP headers and content expiration globally or apply them to a specific directory.

To set HTTP headers for content expiration

1. In the IIS snap-in under **Internet Information Services**, right-click the Web site or virtual directory on which you want to set custom HTTP headers.
2. Click **Properties**.
3. Select **Enable Content Expiration**.
4. On the **HTTP Headers** property sheet, add a custom header in the Custom HTTP Headers box.

   For help in creating a custom header, click the **Help** button on the property sheet.
5. Click **OK**.

For more information, see the IIS 5.0 online product documentation.

Rating Content with PICS

Ratings were designed to help parents and teachers control what children could gain access to on the Internet. With IIS 5.0, you can apply Platform for Internet Content Selection (PICS) ratings to your site. PICS associates labels (metadata) with Internet content, by applying a custom HTTP header to the requested documents. By default, the assigned rating will apply for one year, but it can be customized to apply for any length of time.

The following example illustrates the HTTP custom header information, including a PICS rating. IIS 5.0 will append this information to all of the designated documents requested from the site.

```
CLIENT REQUEST
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
Host: reskit.microsoft.com
Connection: Keep-Alive
Cookie: HTMLA=FONTSIZE=SMALL; ASPSESSIONIDQGQGQGPQ=ADHNMFMDOCNECFKLJGGNMBBL

SERVER RESPONSE
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
PICS-Label: (PICS-1.0 "http://www.rsac.org/ratingsv01.html" l by "someone@microsoft.com" on
"1997.10.17T12:35-0400" exp "1998.10.17T12:00-0400" r
(v 0 s 0 n 0 l 1))
Content-Location: http://domain.com/Default.htm
Date: Fri, 17 Oct 1997 16:36:16 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Tue, 14 Oct 1997 22:55:26 GMT
ETag: "50f59e42f4d8bc1:cd7"
Content-Length: 288
```

Customizing HTML Footers

Many corporations place a standard footer on each page within a Web site. This can be difficult to maintain, because each individual page must be updated when a change is made to the footer. With IIS 5.0, you can include custom footer information in an HTML file that can be appended to the bottom of specified pages. When the footer is changed, all pages are updated automatically.

> **Note** Customized HTML footers currently work only with static HTML pages. Since they degrade performance on high-volume servers, you should use them with care.

The footer file should contain only the HTML tags necessary for formatting the appearance and function of the footer content. For example, to create a custom footer that contains copyright information, write an HTML document that contains only the following tag: <B> Example1 Copyright 1997–1998 </B>.

Building a Web Cluster

Windows 2000 Advanced Server offers customers highly scalable services. If you are an ISP, you are naturally concerned with meeting clients' needs. The best way to guarantee the availability and reliability of your services is by setting up your installation in Web clusters (also called Web farms). Once again, a Web cluster is any Web site served by more than one computer.

When building a Web cluster, you should choose three-tier Web architecture (consisting of Web server programs, COM+ applications, and database applications). The advantage of three-tier architecture is that you can separate the following layers onto different servers, rather than combine them all on one server:

- Publishing content
- Running applications
- Accessing databases

With several inexpensive computers, you can more easily handle large volumes of client requests, without creating any unwanted delays. Sharing the load among more than one computer is especially vital for an ISP that supports critical applications: for instance, those that conduct financial transactions, access databases, support corporate intranets, and perform other key functions on a daily basis.

There are two important terms associated with Web clusters: clustering and load balancing. The next two sections briefly review what these terms mean in the context of Web clusters.

Defining Clustering

Clustering involves grouping independent servers and managing them as a single system. The minimum requirements for a server cluster include the following:

- Two (or more) servers connected by a network
- Cluster management software, such as Network Load Balancing or Cluster Service

The cluster management software performs the following tasks:

- Detects failure on any computer in the cluster
- Recovers from any failure or server overload
- Balances the load of incoming requests over all the servers in the cluster
- Manages the servers as a single system

Clustering servers together into one system offers certain advantages:

- **Always Available to Handle Incoming Requests** Because incoming requests from clients can be balanced among several servers, the likelihood of overloading one server is diminished. If one server becomes overloaded and fails, load balancing can redistribute requests among the remaining servers, without interruption to clients.
- **Easier to Manage** Instead of managing each server separately, you can manage them as a unit.
- **Greater Scalability** You can easily add more servers to the cluster or upgrade existing ones without interruption to incoming client requests.

Defining Load Balancing

Load balancing involves distributing client requests across multiple servers within a Web cluster. Basically, load balancing boosts throughput while keeping response times low. With Network Load Balancing, which is built into Windows 2000 Server, the host detects every incoming IP packet, but only the intended recipient can accept it. Each Network Load Balancing host can specify the percentage of packets that it will handle. Alternatively, the packets can be equally distributed across all of the hosts. If one host fails, the load balancing mechanism redistributes the packets among the remaining hosts.

With the scalability that Network Load Balancing offers, as demand increases, you can upgrade the servers you already have or add new computers to the cluster in order to handle more IP traffic.

The next section describes the load balancing features offered by Windows 2000 Advanced Server: Network Load Balancing, and Cluster Service.

Grouping Load Balancing Features

Network Load Balancing and Cluster Service can enhance the reliability of any installation. Grouping them together is a powerful way to combine back-end database and transactional systems with a Web-based front end, in order to deliver the scalability, availability, and robustness that customers demand. The following list briefly describes each of these solutions and gives an example of how you can integrate them into a three-tier configuration.

- **Network Load Balancing** Balances the load that is primarily generated from incoming TCP/IP traffic. You can set up Network Load Balancing on the first tier, and balance access to your installation over clustered Web servers.
- **Cluster Service** Ideal for grouping database services such as Microsoft® SQL Server 7.0 and other data-intensive applications that require high availability. You can set up the Cluster Service on the third tier for tasks such as billing clients and ordering databases.

Creating a Three-Tier Web Cluster

Together or individually, these three load balancing features lend ideal support for three-tier applications. For example, if your ISP installation hosts a Web-based retail business, you could set up the clustering as follows:

- Use Network Load Balancing in order to balance and distribute client TCP/IP connections over multiple servers for the front-end, user interface (UI) layer. As traffic increases, upgrade your existing cluster or add computers to the configuration. Doing so ensures that the site will always be available to handle incoming connections.
- Use Cluster Service in order to offer two-node failover for the application and data-service layers of a three-tier application. This will create a reliable platform for services such as databases, messaging, and similar applications.

Since a traditional ISP will host all three tiers on a single Windows 2000 Server, this topology will introduce some new concepts.

Figure 8.8 shows what a sample Web cluster looks like, while the arrows show how content is replicated.

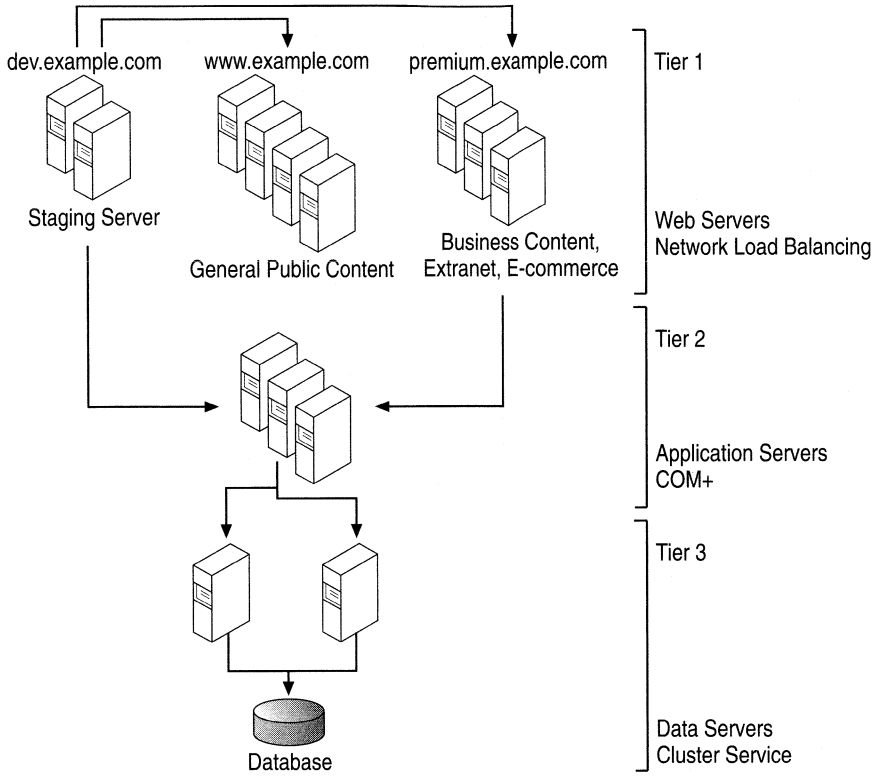


Figure 8.8 Sample Web Cluster

The next sections will help you create a Web cluster.

Calculating Hardware Needs

To set up an installation for a Web-based retail business, you need to determine how many Web servers are actually needed to host a certain application. You can start with one server, test it, and then upgrade it as necessary.

After testing the single server, you might find that the processor utilization is extremely high, that the cache-hit rates are low, and that the server has many requests lined up in the queue. At this point, you can solve the problem by adding processors, memory, and other hardware devices to increase performance. If these solutions fail to improve performance, you can add Web servers.

While there is no mathematical formula for determining the optimum number of servers you should add, you can select from a variety of Web stress tools to help you decide. One such tool is the Web Application Stress Tool. This tool allows you to simulate the load of thousands of users on your site. In conjunction with the System Monitor, you can easily determine at what point a single Web server will crash. A major bottleneck for any Web server is at the HTTP layer. To combat that bottleneck, you should add Web servers, with the ultimate goal of building a Web cluster.

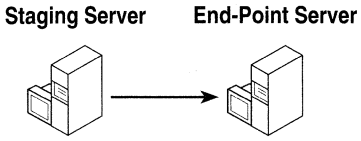
After stressing the single server, you might decide to upgrade so that you can handle a customer's Web site more efficiently. For example, you can add two more Web servers to make up the first tier of your Web cluster.

Building the First Tier

With two new Web servers added to your installation, you now have to balance the load of requests among three servers by running Network Load Balancing. The three servers will now appear as one to the client. Because you assign only one IP address to the servers grouped into a cluster, managing the cluster is very easy. To configure a DNS server, you would set up reskit1.microsoft.com with the IP address 172.16.1.10. That IP address represents all of the hosts in the cluster (three servers at this point). To learn how to assign an IP address, see the "Configure TCP/IP Settings" topic in the Windows 2000 online product documentation.

With this configuration, you have changed the topology of the site from one Web server to three. Now you can use the Web Application Stress Tool again to test the application. You should see an immediate improvement in performance. Next, stress the cluster again, but this time with thousands of simulated users, making sure that the site consumes far fewer resources than the single computer running IIS 5.0. If the application demands higher availability, you can add more hosts, up to a total of 32.

In addition to building a cluster, another way to improve reliability is to bar customers from writing to the production Web servers (also called end-point servers). Set up a staging server (domain.com, for example), where customers can publish content and test their applications. You can then relegate all authoring and development to the staging server.



If you install FrontPage Server Extensions on the staging server (domain.com), customers can then connect with FrontPage to the staging server, write content, and create applications without bringing an application down on the production Web servers.

Once the data is ready for production, use the Content Deployment feature of Site Server 3.0 to replicate the data from the staging server to the production Web servers. The only element of replication that must be the same across the three Web servers is the Web content. When you need to replicate data, Content Deployment, which is very easy to implement, will save you a lot of time.

As your site grows, you might need a separate log file for each server. If you have three servers in a Web cluster, therefore, you would have to analyze three individual log files, presumably by importing all of them into one source to help simplify the task. The Site Server Usage Analyst offers you an easy way to import these files into a source. You can then generate custom reports representing all the traffic on a Web site.

For even more reliability and availability, you can add another cluster and restrict it to business clients only. In Figure 8.8, this cluster was called reskit.premium.com. The strategy here is to divert mission-critical business requests to reskit.premium.com, while leaving reskit.microsoft.com for the general public.

Once you have added servers, created a staging server, and simplified logging, you will notice that the site has become quite complex. To handle the increasing complexity and increased number of applications, you will probably need to build a second tier, in order to maintain the availability and performance of the Web site.

Building the Second Tier

A site quickly becomes more complicated when ASP applications are run in conjunction with Windows 2000 components. Think of ASP applications as the glue that connects the presentation layer to the application and data services layers. ASP provides a rich and robust development environment. But to really increase the overall application performance, consider adding components.

Use an ASP application to invoke a component that is already compiled with your business logic, which resides on the application servers in the second tier. An application server can simply be a Windows 2000 Server that primarily supplies processor power for components.

At this point, the ASP applications that reside on the first-tier Web servers call on the application servers in the second tier to process the business logic. Part of that process could require that the component fetch data from the back-end database, located on the third tier.

Building the Third Tier

To achieve high availability on the third tier, you need to install Cluster Service. To store all of the back-end data that makes up the third tier, most enterprise customers require a single, high-end symmetric multiprocessing (SMP) server (for example, an eight-processor SMP with 4 gigabytes of RAM) running SQL Server 7.0. Cluster Service can handle mission-critical database management, file and intranet data sharing, messaging, and general business applications.

Instead of relying on one high-end Windows 2000 Server, it's best to add a second server for failover. Joined together, these two servers offer higher availability for incoming requests and simplify the task of managing data and applications. Cluster Service not only allows you to connect two servers into a cluster, but can also automatically detect and recover from server or application failures. In addition, it can juggle server workloads, in order to allow for planned maintenance without downtime.

Cluster Service has a built-in mechanism that can handle server failure. If Cluster Service detects that a server has failed, it automatically transfers ownership of resources (such as disk drives and IP addresses) from a failed server to a functioning one. It then restarts the failed server's workload on the functioning server. The entire process, from detecting a failure to restarting the other server, typically takes under a minute.

If an individual application fails (but the server does not), Cluster Service will typically try to restart the application on the same server. If that fails, Cluster Service moves the application's resources and restarts the application on the other server. For detailed information about Cluster Service, see the Windows 2000 Advanced Server online documentation.

Reviewing the Three Tiers

Now it's time to quickly review the growth of domain.com. In this sample scenario, you started with a single IIS 5.0 Web server, from which you essentially ran everything. Since the customer's application demanded highly available, yet scalable, performance, you built a three-tier Web architecture and implemented high-availability Microsoft technologies to each tier. You then tested the single server with the Web Application Stress Tool and realized that, to achieve even higher availability, you needed to build a Web cluster.

To do this, you initially calculated that three servers would suffice. Then you installed Network Load Balancing, whereby three servers essentially appeared as one. Next, you added a fourth Web server as the development or staging server, to which your customer could publish. In order to replicate the developed content, you implemented the Content Deployment feature of Site Server 3.0 on the staging server and the production Web servers. The customer could then connect through FrontPage to the staging server (domain.com), and you were able to easily replicate content to the production Web servers.

To increase performance in the second tier, you added a COM+ application and used ASP applications from the first tier to call on precompiled components from the second tier. These second-tier components called on the third-tier data service.

A high-end SMP server running SQL 7.0 made up the third tier of the system. To achieve high availability, you added Cluster Service, to ensure a fault-tolerant failover system. This topology provides high availability and scalability in spite of planned failures (such as service-pack or hardware upgrades) or unplanned failures (such as hardware failure or the loss of software integrity). If you are an ISP, you can deploy this architecture for your enterprise-level customers who run e-commerce sites or other mission-critical business applications on the Web. You can also use this topology for your common or commodity-type hosting, in which many customers participate within the single cluster.

Adapting IIS 5.0 to a Sample Installation

The following section presents solutions to possible real-life requirements, by showing you how IIS 5.0 can be adapted and tailored to various configurations. Obviously, there is no set solution for a specific ISP installation. Each installation has its own unique requirements, resources, and budget constraints, some or all of which may force you to:

- Adapt your existing network infrastructure, instead of building everything from scratch.
- Tailor solutions to meet customer needs. For example, the architectural design of an e-commerce solution would differ from a home-banking or a Web-publishing solution.

- Work with the unique connectivity requirements of your installation, which most likely was built with specific constraints to solve specific problems.

Although this scenario might not fit your installation exactly, it can serve as a guide for creating your own solutions.

Several Business Clients

Inspired Technologies ISP wants to deliver services for several businesses that cannot afford or do not want to spend money on an in-house Internet gateway. These businesses do, however, want Internet services such as:

- Simple Mail Transfer Protocol (SMTP) and POP3 access to e-mail.
- A Network News Transfer Protocol (NNTP) server for dedicated use (in other words, customer support).
- A Web-hosting server.

Ken from Mightyflight Toys has decided to build three networks in one—an architecture that he mapped out in Figure 8.9.

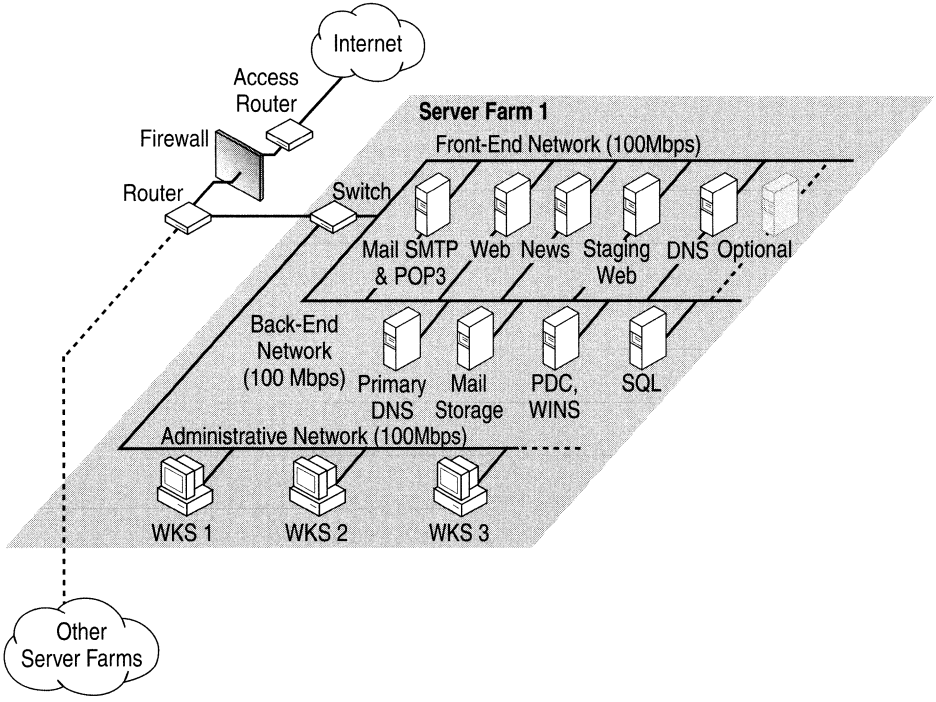


Figure 8.9 Topology with Several Business Clients

Three Networks in One

The architecture for Mightyflight Toys is set up so that a firewall filters access to the Internet. Specifically, the firewall lets in data directed at the services listed in the previous section (SMTP, POP3, NNTP, and so forth), while filtering out unwanted data directed at the other servers. Behind the second router, the architecture is divided into three different networks:

- Front-end
- Back-end
- Administrative

Front-End Network

Since the front-end network consists of all the servers connected to the Internet (production Web servers), it handles traffic coming from and going to the Internet. It can also host the following:

- SMTP and POP3 servers for e-mail connectivity
- One NNTP server for private newsgroups

  Note Mightyflight Toys doesn't support Usenet. If the company did, it would have to come up with a different topology, because Usenet news exchanges a large amount of data among the production Web servers.

- A production Web server and a staging Web server, in order to host and publish customer content.

  Both of these servers exchange data through the back-end network; here, the Content Deployment feature of Site Server copies content from one server to another. This synchronizes content on high-traffic, multiserver Web sites.

- A DNS server that holds the information for the Internet domains that are hosted in a server cluster.

  The secondary DNS for all of these domains can be here, too. This would balance the workload of the primary DNS, if it were to be overloaded, thus distributing primary and secondary domains between the two computers.

Back-End Network

The back-end network contains the following:

- Production Web servers
- Other servers (such as a mailbox storage server, as well as a SQL server for Web-based applications that require access to a database or to legacy systems)
- WINS Servers
- Domain Control Servers
- Internal DNS Servers

In this network, traffic flow is generated by the production-server applications, which do not go directly to the Internet.

Administrative Network

The administrative network contains computers that manage the platform. This network accesses the production Web servers through the back-end network. Note that the administrative network can be shared across many platforms. In other words, the computers dedicated to administration can also administer or monitor other servers or Web clusters in the same facility.

Advantages of This Topology

This topology works well for the following reasons:

- It does not overload the front-end network with traffic generated by applications or with network traffic coming from the administration servers. Therefore, the full front-end network bandwidth can be dedicated to the Internet connection.
- The back end, where the data is stored, is not directly exposed to the Internet. Instead, only the services in the production Web servers can gain access to the back-end network, thus interposing an isolation layer between the network where the actual data resides and the Internet.

  You might want to set up an isolation layer to protect the customers' stored data. If the data is eventually exposed to the Web through the Web server, you need to protect it from unwanted alteration. The domain controllers are in the back end in order to keep them from direct exposure to the Internet.

  > **Note** In this solution, all of the dual-homed computers must have the IP routing option turned off. Otherwise you will lose this layer of isolation.

In the front-end network you must use only officially released Internet addresses, because the front end is the only network connected to the Internet. In the back end, you should use private IP addresses (like 10.x.x.x class A addresses or 172.16.x.x class B addresses that are properly subnetted) in order to save officially released IP addresses for the front end. All traffic between the back-end and the front-end networks passes through the applications in the production Web servers.

Example: Querying a SQL Server

The following procedure shows how a query extracts data from a SQL server database:

1. The user sends a request through the Internet to the front-end network.
2. A Web application then queries the SQL server through the back-end network.
3. The SQL server sends the data back to the Web server through the back end.
4. The Web application builds the response HTML page dynamically and sends it back to the Internet user through the front-end network.

In this example, the network traffic can be divided in order to save bandwidth.

In summary, here are the steps that were used in planning a sample architecture:

1. The network segments were split in order to divide network traffic properly and save bandwidth.
2. An isolation layer was placed between the Internet and the back-end network, where the actual data was stored, thus enforcing the security of the data in the back-end network.

Every component (production Web servers, routers, firewall, switches, and so on) of a cluster can easily be duplicated to offer a greater failover. This way, if one of the components fails, the redundant one starts immediately, thereby reducing the number of off-line services.

Expanding the Platform: Considerations and Suggested Guidelines

You can expand this network architecture to fit your needs. When doing so, you should include the following factors in your plans.

Network Traffic

Even if 100 megabytes per second (MBps) of Ethernet seems to be enough bandwidth initially, remember that this bandwidth is actually shared by all the servers in the network segment. For example, if there are 10 Web servers accessed equally in a 100 MBps network segment, only 10 MBps of bandwidth will be available to each server. To increase bandwidth, you can add some other production Web servers to the existing architecture. Or you can build a similar architecture, if the traffic generated by the production Web servers will overload the existing architecture.

Legacy Applications

You can set up a similar architecture for hosting Web applications that require a connection with some legacy systems. Through the back-end network you can set up a gateway to the legacy system that supplies data to your Web application. You can also build an isolated architecture in order to better protect the legacy system from undesired access by other network segments.

Isolating the various networks helps you set up particular security policies (dictated by customers such as banks) for only a limited number of servers, rather than securing the entire facility. In addressing security issues, you also need to address the costs involved and strike a balance.

Future Expansion

Never underestimate the need for future expansion of Web clusters, because it is inevitable. Note, also, that in this architecture you can scale up the number of Internet connections (just before the firewall), if you need more bandwidth.

Bottlenecks

Sometimes the firewall can turn into a bottleneck if there is a lot of Internet traffic, because it must check every single packet of information flowing in and out of the cluster. To alleviate this problem, see if your firewall software allows you to set up multiple firewalls. If so, you can add other firewalls between the access router and the router connected to the switch, in order to meet the new bandwidth requirements. This way, traffic can be divided among multiple firewalls.

Some Notes on Security

While this section does not provide an exhaustive list of things to do in order to ensure a secure connection to the Internet, it does give some points to bear in mind when you set up an Internet connection. In general, it is better to talk about "levels" of security when discussing the subject. For detailed information about IIS 5.0 security, see "Security" in this book.

First, try to block all unwanted traffic at the router and firewall level. Routers can implement some very elementary rules to allow or deny packet traffic transported between networks or specific hosts, but they are hard to configure. In addition, routers usually do not implement every feature that you might need.

Second, try not to load the CPU of the router with complex access rules that can be easily implemented by a firewall. Usually, Internet Control Message Protocol (ICMP) traffic is blocked with routers. A firewall is a more sophisticated application than a router. It runs on a dedicated computer with two network interfaces, checking every network packet that flows in and out of these interfaces. It will block or allow packet traffic according to specific rules. Firewalls usually offer a better way of tracking dropped packets in log files, which could be a sign of potential network intrusion. For the highest level of security, monitor the log files constantly.

Third, you can set up the Windows operating system to audit accesses on specific resources and to have them recorded in the Security Log. For information, see the Windows 2000 Server online product documentation. For more information about auditing, see "Security" in this book.

In order to prevent unwanted Internet connections to your production Web servers, you can disable the bindings to the workstation, server, and NetBIOS interface.

To disable bindings

1. Open Control Panel.
2. Double-click the **Network and Dialup Connections**.
3. Right-click **Local Area Connection**, and click **Properties**.
4. Select the network adapter for which you want to disable the bindings.
5. Clear the check box in front of all components except **Internet Protocol TCP/IP**.
6. Select **Internet Protocol TCP/IP**, click **Properties**, and on the property sheet, click the **Advanced** button.
7. On the WINS property sheet, select **Disable NetBIOS over TCP/IP**, and click **OK**.
8. Click **OK** again and close each property sheet.
9. Restart your computer.

> **Note** Only experienced administrators should perform this task. If you choose the wrong network adapter, the services might not perform correctly in the back-end network.

For a broader perspective on security in the Windows environment, as well as in-depth information about many aspects of the Windows operating system and Internet security, see "Security" in this book, as well as http://www.microsoft.com/security/default.asp.

Additional Resources

The following Web sites provide additional information about IIS 5.0 and about other features of Windows 2000 Server.

Web Links

http://www.microsoft.com/products/prodref/458_ov.htm
Discusses Microsoft® Windows NT® Enterprise Edition.

http://www.microsoft.com/products/prodref/636_ov.htm
Discusses Microsoft® Site Server® Commerce Edition.

http://www.microsoft.com/backstage/
Gives you an insider's view of how Microsoft operates and maintains its Web site.

http://www.microsoft.com/security/default.asp
Supplies a list of security issues related to Microsoft products.

http://www.microsoft.com/isn/
Gives the latest news and information about Microsoft Internet technologies.

http://officeupdate.microsoft.com/welcome/frontpage.htm
Links to updated information about FrontPage.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 9

Security

The purpose of security is to minimize threats to resources and assets through a combination of policy, hardware, and software. This chapter addresses how to configure a secure Web server that is running Microsoft® Windows® 2000 Server and Internet Information Services (IIS) 5.0 for use on the Internet or an intranet. It is assumed that the reader is familiar with the IIS 5.0 security material in the Windows 2000 online product documentation.

This chapter begins with a discussion of core security principles and explains how the operating system adheres to those principles. It goes on to explore IIS 5.0 security, showing how IIS builds upon Microsoft® Windows® security. Once you have familiarized yourself with the security principles covered in this chapter, you will probably want to read "Site Security Planning" in this book, which describes how to set policies for securing Web resources, applications, and data in an intranet and over the Internet.

In This Chapter

Foundations of Computer Security

Using the Built-in Security Features of Windows 2000 Server

Configuring IIS 5.0 Security

An End-to-End Troubleshooting Example

Defending Against Malicious Attacks

Auditing Access with IIS 5.0 Logs

IIS 5.0 Security Checklist

Additional Resources

Foundations of Computer Security

What does it really mean to say that a computer is secure? To answer this, it is important to understand the core components of computer security:

- Authentication
- Authorization
- Privacy
- Integrity
- Availability
- Auditing
- Nonrepudiation

The following sections define each category of security. (The specific technologies discussed in these sections will be explained in further detail later in the chapter.)

Authentication

Authentication, sometimes referred to as identification, allows initial access to an operating system (such as Windows). The first step in authentication is presenting credentials, followed by system validation of those credentials. Once these are validated, the user can access resources controlled by the system. Today, most authentication credentials take the form of user names and passwords. However, Windows 2000 Server also supports credentials such as certificates held on smart cards. For more details about smart card authentication, see "Authentication" in the *Microsoft® Windows® 2000 Server Resource Kit Distributed Systems Guide*.

Different authentication schemes provide differing degrees of security. For example, passwords are a fairly insecure means of identification because they are relatively easy to guess. Certificates, which are discussed later in this chapter, are very difficult to forge and thus provide a greater degree of security.

Authorization

Once a user has been authenticated, he or she will want access to certain resources maintained by the system such as files, printers, and database tables. Authorization is determined by verifying that the authenticated user has access to the resource.

Note For authorization to succeed, authentication must be performed first.

Access is resolved by comparing information about the user with access control information associated with the resource. If Cheryl is given full access, she can read, write, and delete a file called Info.txt. But suppose Alice has read-only access to this same file. If Alice attempts to write to or delete the file, she will be denied access.

Privacy

Privacy, sometimes referred to as *confidentiality*, is the prevention of message flow to anyone other than the intended recipients. Alice may want the session between her workstation and the network printer to be private so that anyone using a network protocol analyzer (often referred to as a network *sniffer*), such as Microsoft® Network Monitor, cannot see what Alice is printing. This privacy is often achieved by *encrypting* or *scrambling* the data as it flows between the two computers. (Encryption is discussed later in this chapter.)

Privacy is optional; it does not require authentication or authorization.

Integrity

Integrity refers to the ability to protect data from being deleted or changed without the permission of its owner.

If Alice orders 100 widgets from Bob, she does not want the order to be modified en route to Bob (by a malicious attacker) to an order for 1,000 widgets. Like privacy, integrity is optional; it does not require any form of access control mechanisms.

Availability

In principle, availability is somewhat similar to integrity, but it applies to the flow of data and the accessibility of the system. For example, an attack that makes a system crash has made the system unavailable. These types of attacks, often called *denial-of-service* attacks, are extremely hard to prepare for and predict. All systems should offer a degree of availability, based on the necessity of the services a system provides.

Auditing

Auditing refers to maintaining a secure list of all the events on your system, such as who logged on and when, what files a user accessed, and so on. Auditing is considered mandatory in secure systems.

Nonrepudiation

Nonrepudiation is a method of proving either that a user performed an action (such as enrolling in a stock plan or applying for a car loan), or that the user sent or received some information at a particular time. This prevents the individual from fraudulently reneging on a transaction. By comparison, if you purchase an item, you might have to sign for the item upon receipt. If you decide to renege on the deal, the vendor can simply show you the signed receipt.

A comprehensive nonrepudiation plan usually requires authentication, authorization, data integrity, and auditing. In addition, nonrepudiation requires a message on the Web page, warning that the action the user is about to take is legally binding. This does not make the Web server more secure, but it does make Web transactions (such as purchasing an item) more secure.

Today, electronic nonrepudiation across the Internet is new and there is little in the way of legal precedent. This is sure to change as more business is performed across the Web.

Threats, Vulnerabilities, and Attacks

When considering the security of a system you will need to determine all the possible threats, vulnerabilities, and attacks. You will also need to consider the appropriate tradeoffs between security on one hand, and usability and cost on the other. For more information, see "The Bottom-Line Cost of Security" later in this chapter.

Threats

A threat is the possibility of system compromise. For example, a threat could be the potential for unauthorized people to gain access to sensitive information, such as credit card information or health records. Threats usually involve confidential information.

Vulnerabilities

A vulnerability is a weakness in a system that might allow a threat to become realized. For example, if a secure server is left unlocked, it might be possible for a malicious user to physically access the computer and copy files to a floppy disk.

Vulnerabilities in computer systems could be bugs in the software or hardware, or they could be the result of administrative error. Examples of administrative errors include choosing a weak password (one that someone can easily guess) for the Administrator account, or accidentally setting an insecure discretionary access control list (DACL) on a confidential file.

Attacks

An attack takes advantage of an existing vulnerability. For example, suppose a malicious user knows that some users have weak passwords and tries guessing them until gaining access to restricted resources.

Types of Attack

It is important to realize the different types of security attacks you might encounter. Once you understand these, you will learn the appropriate countermeasures to take.

The three main types of attack are:

- Disclosure of data
- Corruption of data
- Denial of service

Disclosure of Data

Disclosure refers to unauthorized or inappropriate access to sensitive data. This is probably the most common form of attack. An example of disclosure is a file that holds confidential payroll information. If this file finds its way into the hands of someone who should not be privy to the data, then the data has been disclosed.

Corruption of Data

This is a particularly insidious attack, especially if the victim has very poor backup policy. Data corruption is mainly the realm of the computer virus, rather than that of intruders. Very few intruders actually wish to destroy data; most attack computer systems for entertainment and for the intellectual challenge.

If data is corrupted, the only real remedy is restoration from a previous backup.

Denial of Service

Denial-of-service attacks have become one of the most common forms of attack on the Internet today because many can be started remotely.

There are two types:

- The first occurs when a system is so utterly consumed that it cannot spare any further resources for other users.
- The second occurs when a system is made to crash and hence it cannot service other users.

An example of the first type of denial of service is a program that swamps all the printers in an organization with massive high-priority print jobs. This will prevent all other users from printing any documents and, in the case of a company that relies on printing to perform its core business function, could result in a loss of earnings.

An example of the second type of denial of service is a program that sends a large block of data to a remote program, knowing that the server will fail when it attempts to process the data. This is often referred to as a "buffer overflow" attack.

The Bottom-Line Cost of Security

Practically all data held on computers has some value. Sometimes this value is small, such as simple e-mail messages between friends saying "Hi!" Other times the value is great, such as documents pertaining to military secrets or to business tactics and strategies.

When determining the appropriate level of security for your Web servers, you need to consider the following:

- The value of the data (the cost to create it, as well as the cost to the organization if the data is leaked to malicious users)
- The cost of securing the data
- Usability tradeoffs

Note The cost to the organization if data is leaked also includes the intangible cost involving loss of client or shareholder faith.

As shown in Figure 9.1, as a system becomes more secure, it also becomes less usable. At some point you might realize that a system is so secure that your intended audience cannot access your service.

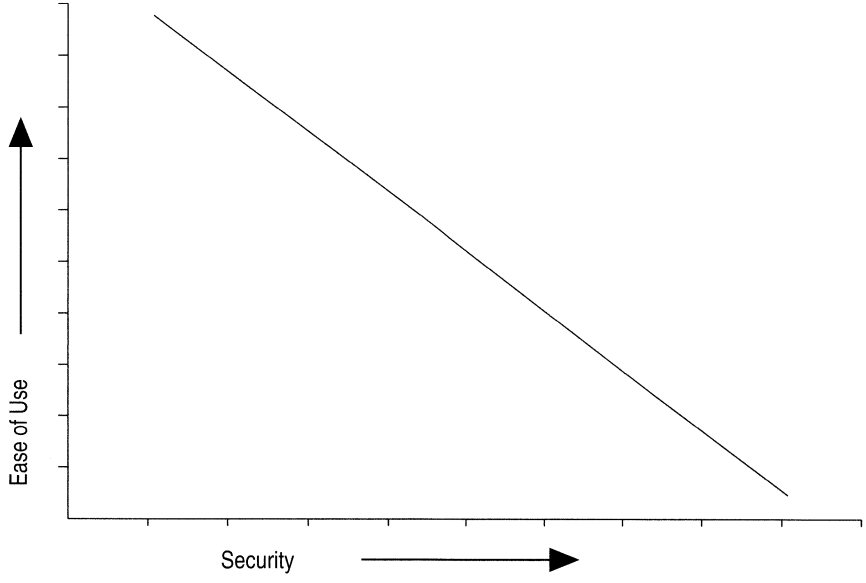


Figure 9.1 The Tradeoff Between Usability and Security

Also note that there is a point at which it is not worth further securing the system, as the cost of deploying the security becomes greater than the cost associated with the risk of a security attack. Is it worth spending $100,000 to secure data valued at only $25,000?

Once you understand the value of your data, the deployment cost, and the usability tradeoffs, you can begin planning how to secure your system. This is a very large field and is beyond the scope of this book. For more information about assessing values and costs, see "Additional Resources" at the end of this chapter.

Using the Built-in Security Features of Windows 2000 Server

This section covers the following topics:

- Authentication in Windows 2000 Server
- Authorization in Windows 2000 Server
- Privacy and Integrity Mechanisms in Windows 2000 Server
- Availability in Windows 2000 Server
- Auditing in Windows 2000 Server

Authentication in Windows 2000 Server

Windows 2000 Server supports authenticated logon, meaning that the user must present credentials (usually a combination of user name and password) for identification. Once the user is authenticated by the operating system, a security token is attached to all applications that the user runs. All processes (applications) must have a token associated with them that identifies the user and the Windows groups to which that user belongs. The token contains the user's security identifier (SID) and the SIDs of all the groups to which the user belongs. An SID uniquely identifies all users and groups (of users) in the Microsoft® Windows® operating system.

In order to log on, the user must have an account in either the security account manager (SAM) database or in the Microsoft® Active Directory™ directory service.

What Does an SID Look Like?

Fortunately, most administrators will never have to deal with SIDs directly. Here is a sample SID:

```
S-1-5-21-2127521184-1604012920-1887927527-1004
```

The first part, S-1-5, identifies Windows 2000 Server; the next four blocks of numbers identify the Windows domain or workgroup; and the last number identifies the particular user or group.

Well-Known SIDs

Each and every account and group in the Windows operating system has a unique SID, which is also unique to that domain of servers. However, some SIDs are termed *well-known*. In other words, they are the same regardless of what domain you use. These SIDs include:

Account	SID	Comment
LocalSystem	S-1-5-18	The account which most system services use
Everyone	S-1-1-0	All users; the Everyone group
Interactive	S-1-5-4	Users who can log on for interactive operation
Network	S-1-5-2	Users who can log on across a network

Logon Types and Token Types

Accounts can log on to a Windows server if they have the appropriate privilege.

You can set logon types using the Group Policy editor, shown in Figure 9.2. The Group Policy editor is accessed by loading the Computer Management administrative tool from Start/Programs/Administrative Tools.

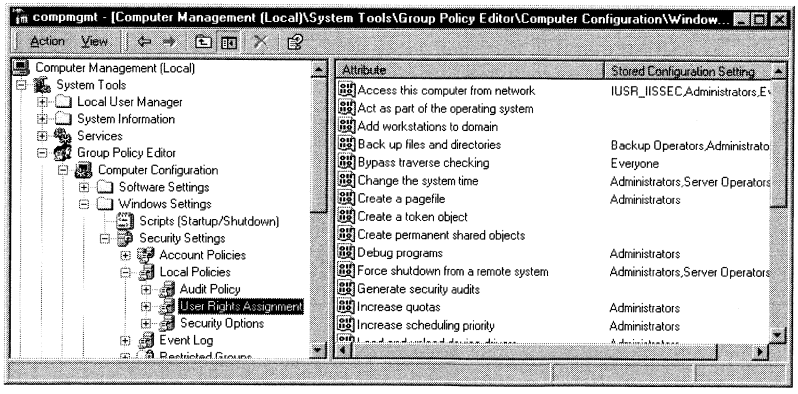


Figure 9.2 Setting Logon Privileges Using the Group Policy Editor

When a user with the appropriate privilege is authenticated and the security token is generated, the token has a logon type. The most common logon types in Windows are the following:

- Interactive logon
- Network logon
- Batch logon

Interactive Logon

An interactive logon is generated when the user is physically present at the computer while entering credentials. The account logging on must have the Log on locally privilege; if it does not, the account will fail to log on. An example of an interactive logon is pressing CTRL-ALT-DEL at the Windows prompt, then entering a user name and password.

Also note that a user can be logged on interactively (as if present at the computer) across the network. Most user accounts are logged onto IIS 5.0 interactively.

Network Logon

A network logon is generated when the user is connecting to a network computer. The account logging on must have the Access this computer from network privilege. If it does not, the account will fail to log on.

An example of a network logon is when the user is logged on to a computer and then accesses a resource, such as a printer, on a network computer that is running Windows. When the user attempts to connect to the network resource, the Windows operating system will automatically attempt a network logon.

Batch Logon

A batch logon is rarely used in the Windows operating system because it is usually reserved for applications that run as batches, such as bank account reconciliation and big print spools. The account logging on must have the logon as a batch job privilege. If it does not, the account will fail to log on.

> **Note** If you are familiar with Microsoft® Windows NT® 4.0, you will notice four new logon privileges in Windows 2000 Server: deny logon locally, deny network logon, deny batch logon and deny service logon. They provide the ability to deny logon privileges. For example, you can allow a group of users the ability to log on locally but not allow one specific, nontrusted account to do so.

Services and Service Security

The Windows operating system can run special applications called services, which are similar to UNIX daemons. Services generally start along with the operating system and run in the background, without any user interface. Examples include Microsoft® SQL Server, the Spooler, IIS Web Server, and the Windows® Event Log. Because they are applications, services must run in the context of a user account. The LocalSystem account, discussed below, is set aside for this purpose.

Services pose some interesting security issues that you must be aware of, most notably:

- Services, as mentioned, usually have no user interface. Therefore, any component that might work correctly from, for example, Microsoft® Visual Basic® might not work correctly from IIS 5.0, if the component expects a user interface.
- IIS 5.0 might be running under a different user account than the logged-on user, and thus might not have access to the same registry settings. A common example of this is Open Database Connectivity (ODBC) Data Source Names (DSNs); by default these are set per user. To make DSNs work with IIS 5.0, they should be System DSNs.

The LocalSystem Account

The LocalSystem account has no password, and no one can log on under this account. On the local computer, LocalSystem account can perform many tasks required of administrators. It has very few privileges beyond the boundaries of the server on which it is being used, thus helping to prevent attacks if an attempt is made to compromise a service.

LocalSystem is the most common account under which Windows services are run.

> **Important** Changing the Log on as option from the LocalSystem account to another user account can cause a service to fail on startup. IIS 5.0 requires the startup account to be LocalSystem.

Privileges and Attack Prevention

When a process runs, it does so in the context of a user account. If the process is running as a highly privileged account (such as the Administrator), and is compromised, any malicious attack will be in the context of the Administrator account, thereby making the potential for damage greater. Because of this, it is recommended that you always run in a low-privilege account. This is known as the principle of least privilege.

Discretionary Access Control Lists

To determine whether the user of an application is permitted access to a resource such as a file or a printer, the Windows operating system takes the user information from the security token associated with the application, and compares this information with the discretionary access control lists (DACLs) associated with that resource. A *DACL* is a list of access control entries (ACEs) that contain a user name or group, and include which permission that user or group has for each resource.

To use and set DACLs on files you must be using the NTFS file system.

The comparison of DACLs and user information is what determines who can gain access to a resource in the Windows operating system. If the DACL and the user information in the token are not the same, the user is denied access to that resource.

Impersonation

The Windows operating system also supports the ability to impersonate another user. Impersonation is useful in a distributed environment when servers must pass client requests to other servers or to the operating system. This way, the operating system can perform the request as if the original client had made it. You don't have to maintain a set of user accounts and passwords on the network server, nor do you have to ask the user to log on again in order to access the network resource.

Why Is Impersonation Important?

Impersonation reduces the number of times a user is required to enter a password. However, most services run as LocalSystem, which allows full access. This being the case, how does a service access a resource (such as a file) securely on behalf of the client?

Here's an example of how impersonation works. Cheryl wants to delete a file called Data.asp. The file is protected so that only the operating system (LocalSystem), administrators, and certain trusted users can access it. Since Cheryl is not on this list of trusted users, it would seem that she cannot access the file.

However, Cheryl *can*, in fact, delete the file because the application through which she deletes it is actually running under LocalSystem. Since Cheryl has full access, this is obviously not a safe, secure environment.

To safeguard against the file being deleted, all server processes should be configured to impersonate the requesting user before accessing the resource. If the server processes are configured in this way, the server will then impersonate Cheryl and attempt to delete the file. However, this will fail because she does not have delete permission or any other permissions to the file.

Authentication Methods in Windows 2000 Server

Windows 2000 Server supports a number of authentication schemes, most notably:

- Windows NT Challenge Response (now known as integrated Windows authentication)
- Kerberos v5 authentication
- Client authentication certificates

Windows NT Challenge Response authentication was the default authentication used prior to Windows 2000 Server. In Windows 2000 Server, Kerberos v5 has replaced Windows NT Challenge Response authentication (now known as integrated Windows authentication) as the default mechanism. For most users the differences between the two are transparent.

Windows 2000 Server also supports client authentication certificates (explained later in this chapter) as a means to provide authentication credentials. Rather than using a password to gain access information stored in the client certificate, the public key is used. Regardless of whether Windows 2000 Server uses Kerberos v5 authentication, or a client authentication certificate, the result is the same: a user token.

Authorization in Windows 2000 Server

Authorization is determined by using DACLs. These can be set on any Windows object, but the most common are DACLS on files, as well as on registry nodes, and Active Directory nodes. For more information about using the Windows DACL Editor, see "Access Control" in the *Distributed Systems Guide*.

To demonstrate how access is determined, it is necessary to look at how DACLs are structured. As mentioned previously, a DACL is a series, or list, of ACEs; each ACE contains the SID of a user or group account and indicates whether that account has access to the object in question. For example, a DACL may contain four ACEs:

- Guests (No access)
- System (Full access)
- Administrator (Full access)
- Everyone (Read access)

> **Note** The deny access ACE is placed first. No matter in what order you place the ACEs when you set permissions on an object, for speed reasons Windows 2000 Server will always move deny access ahead of allow access.

When determining access, Windows 2000 Server looks at the DACL on the object and compares the user SID and group membership SIDs in the calling token to see if this SID is listed in the DACL. If a SID matches the SID in an ACE and if the ACE allows access, then access to the object is granted. Otherwise, it is denied.

Privacy and Integrity Mechanisms in Windows 2000 Server

Windows 2000 Server supports three major privacy and integrity protocols: Secure Sockets Layer (SSL), Transport Layer Security (TLS), and IP Security (IPSec). Data privacy and data integrity are usually provided through encryption.

A Brief Overview of Public and Symmetric Key Cryptography

Cryptography is a set of standards and protocols for encoding data and messages, so that they can be stored and transmitted securely. The following introduces the basic terminology of cryptography and explains some of the common methods used.

- Cryptography allows you to achieve secure communications, even when the transmission medium (for example, the Internet) is untrustworthy. You can also use it to encrypt your sensitive files, so that an intruder cannot understand them.
- Cryptography can be used to ensure data integrity as well as to maintain secrecy.
- Cryptography makes it possible to verify the origin of data and messages, by using digital signatures and certificates.
- When you use cryptographic methods, the cryptographic keys must remain secret. The algorithms, key sizes, and file formats can be made public without compromising security.

The two fundamental operations of cryptography are *encryption* and *decryption*. Encryption involves scrambling the data in such a way that it becomes infeasible to deduce the original information, unless you have access to the appropriate key. Decryption is the reverse process—scrambled data is turned into the original text by using a key.

In order to encrypt and decrypt, you need an *encryption algorithm* and a *key*. Many encryption algorithms exist, including Data Encryption Standard (DES), Rivest-Sharmir-Adleman (RSA) encryption, RC2, and RC5. A key is used in conjunction with the algorithm to convert the plaintext (readable by humans) into ciphertext (scrambled, unreadable by humans).

DES, RC2, and RC5 are known as *symmetric key technology* because the key used to encrypt the data is the same one used to decrypt it. Hence the key must be a shared secret between the party encrypting the data and the party decrypting it. You can use public key technology to pass the key securely to the other party.

RSA is known as *public key*, or *asymmetric, technology*, because two keys are used: a public and a private key. The keys are mathematically related, but it is infeasible to deduce one without knowing the other. The private key is kept private—only the party generating the key pair should have access to it. The public key can be freely shared over an insecure medium such as the Internet.

Continued

A Brief Overview of Public and Symmetric Key Cryptography (*continued*)

With public key systems, there is no shared secret between the two parties. If the public key is used to encrypt the data, then only the private key can decrypt it. Similarly, if the private key is used to encrypt the data, then only the public key can decrypt it. The following scenario provides a simple example of how public keys are used.

A Public Key Scenario

In this scenario, Alice wants to send Bob a message, but she wants to make sure only Bob can read it. To do this, the following steps are performed:

- Alice gets a copy of Bob's public key, possibly from the directory, a Web site, or an e-mail message.
- She uses this key to encrypt the data.
- She sends the encrypted data to Bob. Because the data was encrypted using his public key, only his private key can be used to decrypt it.
- Bob uses his private key to decrypt the data, and reads Alice's message.

Secure Sockets Layer

The SSL protocol provides communications privacy, authentication, and message integrity by using a combination of public-key and symmetric encryption. By using this protocol, clients and servers can communicate in a way that prevents eavesdropping, tampering, or message forgery.

In the case of an SSL connection between a Web browser and Web server, you must enter HTTPS rather than HTTP as the protocol type in the URL. This will instruct the Web browser to use a different port for the communication; the Web server will be listening on this port for SSL requests. By default, Web data (HTTP) uses Transmission Control Protocol (TCP) port 80, while SSL (HTTPS) uses TCP port 443.

Transport Layer Security

TLS is very similar to SSL in that it provides communications privacy, authentication, and message integrity by using a combination of public-key and symmetric encryption. In fact, TLS supports the option of reverting to SSL support, if need be. However, there are some important differences. TLS supports different encryption algorithms than SSL and is an Internet Engineering Task Force (IETF) draft standard. Many people view TLS as the likely successor to SSL in the long term.

Because TLS is designed to be a replacement for SSL, using either protocol should be transparent to the user. However, the software must be TLS- and SSL-aware.

IP Security

IETF developed IPSec as the security protocol for Internet Protocol (IP). In fact, IPSec was designed for the next generation of IP, which is IPv6. However, IPSec can be used with the current implementation of IP, which is IPv4.

> **Note** One of the issues with Transmission Control Protocol/Internet Protocol (TCP/IP) is that it was not designed with security in mind; for example, TCP/IP does not accommodate authentication or privacy.

IP Security and Authentication

It is easy for an unauthorized user to *spoof* IP packets, that is, to make packets appear to have come from another destination. This is achieved by writing applications that build complete, but invalid, IP packets and then sending them to a target computer. The packets are invalid in that the source IP address is incorrect, does not exist, or is unreachable because of router settings. The target computer attempts to set up a connection with the unreachable source but fails, and in so doing (although the target will wait for a while to accommodate latencies in the network), the target must allocate memory for the connection anyway. If the system receives a large number of the invalid packets, the target will eventually run out of memory and probably stop responding. This is a denial-of-service attack.

IPSec can provide strong authentication of packet data in order to help alleviate many of these common spoofing attacks.

IP Security and Privacy

No data is encrypted at the TCP or IP layers; it is up to higher-level protocols such as SSL and TLS to perform this work. Because the data is unencrypted, it is very easy to gather important information like user names and passwords by sniffing the network. IPSec can encrypt data in order to help prevent data sniffing.

For more information about IPSec, see "Internet Protocol Security (IPSec)" in the *Microsoft® Windows® 2000 Server Resource Kit TCP/IP Core Networking Guide*.

When to Use SSL, TLS, or IPSec

At first glance it appears that Windows 2000 Server supports three protocols that do the same thing: provide authentication, data privacy, and data integrity. IPSec, which is lower in the network stack shown in Figure 9.3, secures all data flowing from one computer to another. Applications do not need to be modified to support IPSec, whereas they do need modification in order to support SSL and TLS. Refer to the TCP/IP books listed in "Additional Resources" at the end of this chapter.

In summary, SSL and TLS are easier to use than IPSec because there is no complex user setup, but applications need to be SSL- and TLS-aware. IPSec is more difficult to implement, but is totally transparent to all applications because it is lower in the network stack.

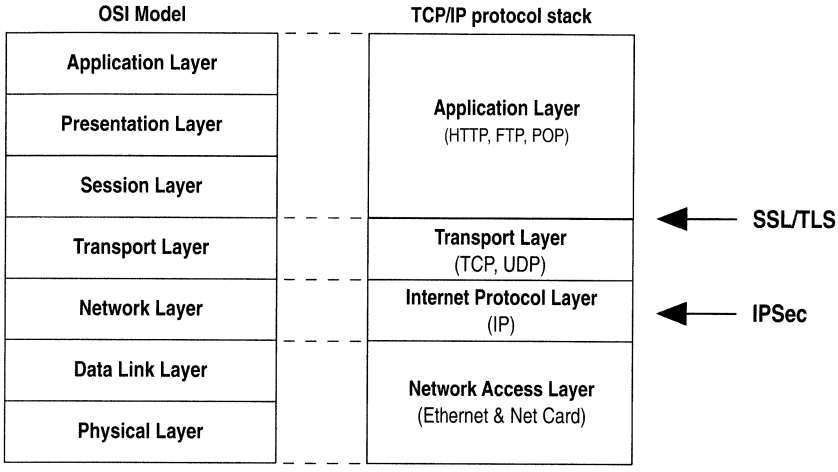


Figure 9.3 SSL, TLS, and IPSec in the TPC/IP Protocol Stack

Availability in Windows 2000 Server

A computer is *available* if its resources are accessible in a timely fashion. Denial-of-service attacks are designed to make a system unavailable by slowing it down, exhausting resources, or stopping it from running.

Windows 2000 Server has a number of features to support availability: most notably, NTFS disk quotas and clustering.

Windows NTFS Disk Quotas

Windows 2000 Server supports the use of disk quotas for NTFS disk volumes. You can use disk quotas to monitor and limit disk space use on a per-user, per-volume basis; users are charged only for the files they own.

If a malicious user could use up all available disk space on a Web server, this would either prevent the server from running correctly or prevent other users from being able to use the server. This malicious consumption of resources is a denial-of-service attack, of which quotas can help reduce the chance.

When you enable NTFS disk quotas, you set two values: a quota limit and a quota-warning threshold. The threshold specifies a warning disk space usage. Events are automatically logged in the Windows Event Log when users exceed these limits.

For example, you can limit a user's quota to 20 megabytes (MB) and the quota threshold to 18 MB. In this case, the user can store no more than 20 MB of files on the volume. If the user stores more than 18 MB of files, the quota system logs a system event as a warning.

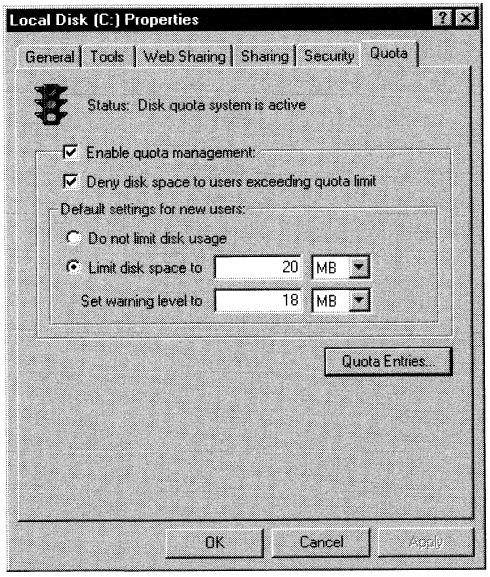


Figure 9.4 Setting Default NTFS Disk Quotas

Using Disk Quotas with IIS 5.0

When using disk quotas in conjunction with IIS 5.0, you need to consider which authentication scheme you will use and how much disk space you will allot to each user.

For example, if your computer has two Web sites—such as reskit.microsoft.com and reskit2.microsoft.com—that map to two different directories, as shown in Table 9.1, you could configure each to have a different anonymous user account and then set quotas on these accounts:

Table 9.1 Examples of Disk Quotas

Site	Directory	Anonymous User Account	Disk Quota
reskit.microsoft.com	C:\InetPub\reskit	IUSR_reskit	Administrators: No Limit
			IUSR_reskit: 20Mb
reskit2.microsoft.com	C:\InetPub\reskit2	IUSR_reskit2	Administrators: No Limit
			IUSR_ reskit2: 10Mb

Clustering Service

A cluster is a group of independent systems that work together as a single system. A client interacts with a cluster as though it were a single server. Cluster configurations increase the availability of a system's resources. If a system or application in the cluster fails, the cluster software responds by restarting the failed application or dispersing the work from the failed system to the remaining systems in the cluster.

Microsoft® Windows® 2000 Advanced Server supports clustering. For more information about Clustering Service, see the documentation included with your Windows 2000 Advanced Server software.

Auditing in Windows 2000 Server

The Event Viewer in Windows 2000 provides auditing information. Windows can be configured to audit certain events (such as when users are logged on), when they access resources (such as files), or when they attempt to use special privileges (such as the ability to debug an application or perform a data backup).

Auditing can be turned on and off by using the Computer Management tool and by selecting System Tools/Group Policy/Computer Configuration/Windows Settings/Local Policies/Auditing Policies.

For a server running IIS 5.0, it is recommended you audit by using the following: "Audit Events," which is the event name you are interested in; "Audit success attempts," which indicates that you are interested in successful events; and "Audit failed attempts," which indicates that you are interested in failures when that event is performed. "On" means the event is being audited, while "off" means it is not. Table 9.2 provides examples for various events:

Table 9.2 Auditing Events in Windows 2000 Server

Audit Event	Audit Success Attempts	Audit Failed Attempts
Account Logon	On	On
Account Management	Off	On
Directory Service Access	Off	On
Logon	On	On
Object Access	Off	Off
Policy Change	On	On
Privilege Use	Off	On
Process Tracking	Off	Off
System	Off	Off

For example, it is quite normal to audit for successful and failed events You might want to see when users are logging on, as well as when people might be attempting to log on by guessing someone else's password.

Note that in Windows 2000 Server there are two types of logon/logoff events: Account Logon/Logoff and Logon/Logoff.

For an excellent summary of how to audit logon events, see "Auditing User Authentication" at http://support.microsoft.com/support/kb/articles/q174/0/73.asp.

Configuring IIS 5.0 Security

This section covers the following topics:

- IIS 5.0 Authentication Modes
- Extending IIS 5.0 Security
- File and Directory Security
- Virtual Directory Security
- Secure Communications with SSL and TLS
- How Access is Determined
- Troubleshooting Permissions

IIS 5.0 Authentication Modes

All users must be authenticated before they can gain access to resources in IIS 5.0. Each HTTP request from a browser runs on IIS 5.0 in the security context of a user account on the Windows operating system. IIS 5.0 executes the request in a thread that impersonates the user's security context. An application, such as IIS 5.0, can have many simultaneous threads of execution internally, each acting on behalf of different users.

The operations that are performed during the execution of the HTTP request are limited by the capabilities granted to that user account in Windows. The user account needs to be created either on the IIS 5.0 server or in a domain of which the server running IIS 5.0 is a member. The latter is more common in intranet applications.

IIS 5.0 supports five Web authentication models:

- Anonymous
- Basic
- Integrated Windows authentication
- Digest
- Client Certificate Mapping

There are also two FTP authentication models:

- Anonymous
- Do Not Allow Anonymous

Before looking at each authentication scheme in detail, an explanation of Web authentication is necessary.

How Web Authentication Works

Web authentication is a communication between the browser and the server, involving a small number of HTTP headers and error messages.

The flow of communication is:

1. The Web browser makes a request (for example, an HTTP GET).
2. The Web server performs an authentication check. If this fails because authentication is required, then the server sends back an error message (usually a 401—Access Denied), along with information so that the Web browser can resubmit the request as an authenticated request.
3. The Web browser uses the server's response to construct a new request that contains authentication information.
4. The Web server performs an authentication check. If the check is successful, the server sends the data that was initially requested back to the Web browser.

Anonymous Web Authentication

The Windows operating system is configured to accept only valid users. Because the Internet is extremely anonymous—in that very few Web sites prompt visitors for a user name and password—IIS 5.0 creates the IUSR_*computername* account so that real Windows accounts can be used in an anonymous Internet. This account is granted to anonymous users who then use a random password, defined when IIS 5.0 is set up, on the local computer. This account gives anonymous users the right to log on locally. Anonymous user access can be reset to use any valid Windows account.

> **Note** With IIS 5.0 you can set up different anonymous accounts for different Web sites, virtual directories, directories, and files. This provides a great deal of flexibility and fine control, telling what accounts will be used where within the site.

If the computer running Windows is a stand-alone server, the IUSR_*computername* account is on the local server. If the server is a Domain Controller, the IUSR_*computername* account is defined for the domain.

Windows uses IUSR_*computername* when a user is authenticated by IIS 5.0 with Anonymous authentication. In other words, a real Windows user account is being used for all nontrusted anonymous access.

Figure 9.5 shows an example of what is entered into the Event Viewer in Windows 2000 when Anonymous authentication is used. Note that the logon process is performed by IIS 5.0 itself and that the logon is a network logon (the logon type is 3 in the audit log entry). This is because the Allow IIS to control password option is enabled. If this option were disabled, the logon would be interactive (the logon type would be 2 in the audit log entry).

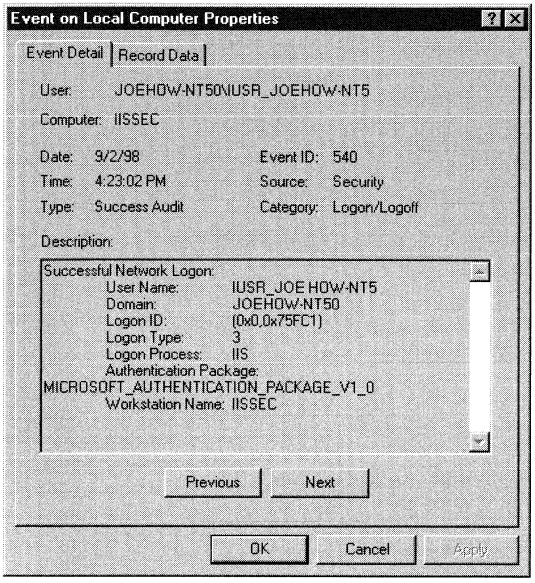


Figure 9.5 Event Log Entry for an Anonymous Logon

Anonymous Authentication Header Flow

Figure 9.6 shows the flow of HTTP headers when using Anonymous authentication.

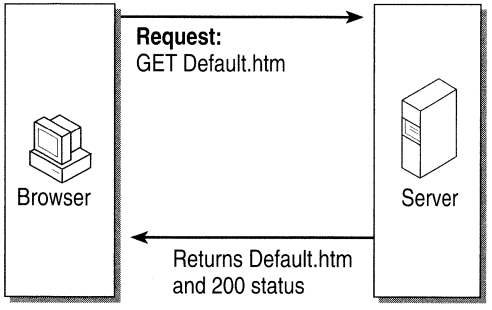


Figure 9.6 Anonymous Access to a Resource

Note If there is a DACL conflict while trying to access the resource (for example, if the IUSR_*computername* account does not have access to Default.htm), the Web server will not return a 200 status. The status will be a 403, which prompts the user to enter a user name and password.

Anonymous Authentication and Allow IIS to Control Password

Prior to IIS 4.0, the administrator had to make sure the anonymous user account passwords were the same in both the Internet Services Manager tool and the Windows User Manager. Failure to do so led to logon failures. To solve this dilemma, Password Synchronization was introduced in IIS 4.0 in order to make administration easier.

Password Synchronization uses a slightly different logon method, which has some side effects that will be explained in detail later. By default, IIS password control is enabled. Figure 9.7 shows the password control option.

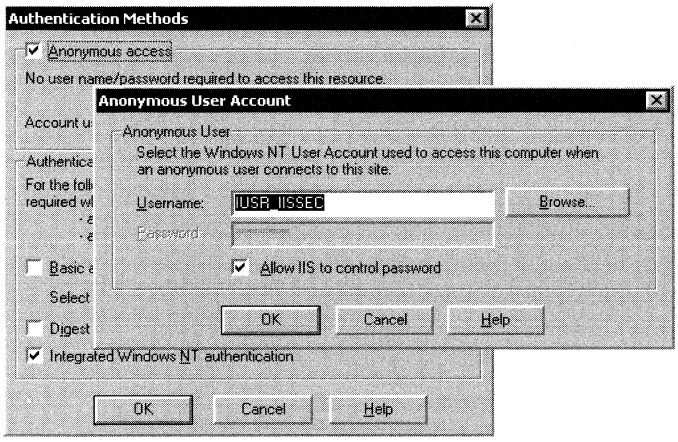


Figure 9.7 The IIS 5.0 Password Control Option

Allowing IIS 5.0 to Control the Anonymous Password

Authentication is performed differently when this option is enabled because IIS 5.0 has to inform Windows that the password is correct. A *subauthenticator* can perform this task. Windows allows subauthenticators—implemented as subauthentication dynamic-link libraries (DLLs)—to be used in conjunction with the normal Windows authentication system.

A subauthentication DLL allows the authentication and validation criteria stored in the Windows user account database to be replaced. For instance, a particular server might supply a subauthentication DLL that validates a user's password via a different algorithm, that uses a different granularity of logon hours, or that specifies workstation restrictions in a different format. All of this can be accomplished using subauthentication DLLs, without sacrificing use of the Windows user account database and thereby losing its administration tools.

IIS 5.0 supplies a subauthentication DLL called Iissuba.dll. The function of this DLL, in terms of Anonymous authentication, is to verify that the password is correct, inform the Windows operating system that the password is valid, and hence log on the user.

You can find more information about Windows subauthentication in the Microsoft® Visual Studio® 6.0 online product documentation. Visual Studio 6.0 ships with a subauthentication sample called SubAuth.

Not Allowing IIS to Control the Anonymous Password

When the password control option is not enabled, IIS 5.0 calls the LogonUser() Application Programming Interface (API) in Windows to log on the account. IIS passes in the user name and password configured by the administrator. If this matches the user name and password set up in Windows User Manager, the account is successfully logged on, the security token is cached by IIS 5.0, and the account is impersonated.

The Side Effect of Allowing IIS to Control the Anonymous Password

Subauthentication DLLs require a network logon, which can produce problems compared to a batch or interactive logon. The most notable is, in some cases, the inability to access resources such as files or Microsoft® Access databases on a network computer. If you see this problem, turn the password control option off; this will force IIS 5.0 to use normal authentication and to log the account on locally.

Basic Authentication

Basic authentication, part of the HTTP 1.0 specification, is the method supported by most Web servers and Web browsers. With Basic authentication, you can restrict access to files on the server that is running IIS 5.0, by using NTFS security. This requires the user to enter credentials, which allows tracking of who has access to what (based on the user ID). You can use this method in order to restrict access to some parts of the Web server when:

- You cannot guarantee that the user's browser supports integrated Windows authentication (Microsoft® Internet Explorer 2.0 or later supports this).
- You need to authenticate through a proxy server.

To use Basic authentication, grant each user account the Log on locally user right on the IIS 5.0 server. These accounts should have file access controlled; place them in a user group that has access only to the required files that are on the server.

When using Basic authentication, the browser prompts the user for a user name and password. This information is then transmitted across HTTP where it is lightly scrambled using Base64 encoding (You can get more information about Base64 encoding from the Internet standard RFC1521.) IIS 5.0 takes this user name and password and authenticates the user as the corresponding Windows user.

Keep the following in mind when using Basic authentication:

- Basic authentication will not succeed if the user doesn't have local logon rights at the Web server. This is the default; it can be changed to a network logon, for example, if you set the **LogonMethod** appropriately. Refer to the IIS 5.0 online product documentation or the Microsoft® Active Directory™ Service Interfaces (ADSI) sample at the end of this chapter for more information about **LogonMethod**.
- A user who can obtain physical access to the host that is running the Web server will be permitted to start an interactive session at the computer. This is because the user has local logon rights. Be sure to set the appropriate NTFS file protections in order to control this.

Basic authentication is inherently insecure. Passwords are encoded but not securely encrypted. As a result, a simple network sniffer can watch for the HTTP authentication headers and Base64 decode this data to obtain the real password. Figure 9.8 shows an example of a network sniff.

```
HTTP: GET Request (from client using port 3840)
  HTTP: Request Method = GET
  HTTP: Uniform Resource Identifier = /data/data.htm
  HTTP: Protocol Version = HTTP/1.1
+HTTP: Undocumented Header = Accept: */*
+HTTP: Undocumented Header = Accept-Language: en-us
+HTTP: Undocumented Header = Accept-Encoding: gzip, deflate
+HTTP: Undocumented Header = User-Agent: Mozilla/4.0 (compatible; MSIE 5.0b1; Windows NT 5.0)
+HTTP: Undocumented Header = Host: iissec
+HTTP: Undocumented Header = Connection: Keep-Alive
+HTTP: Undocumented Header = Authorization: Basic VGVzdFVzZXI6Tm9uZSE=
```

Figure 9.8 Partial Network Trace of HTTP Basic Authentication

In the bottom line, shown in Figure 9.8, the Authorization header indicates the use of Basic authentication followed by some Base64 encoded data. If you were to decode the *VGVzdFVzZXI6Tm9uZSE=* string, you would find this is the *TestUser* account using a password of *None!* to access this Web server.

To make passwords more secure, and hence Basic authentication more secure, you can use SSL support in order to establish a secure session. This way, the password will still be encoded, but the HTTP session carrying the data will be encrypted using cryptographically-secure mechanisms. However, keep in mind that SSL negatively affects performance because all data in the requested page must be encrypted.

Basic Authentication Header Flow

Figure 9.9 shows the flow of HTTP headers when you use Basic authentication.

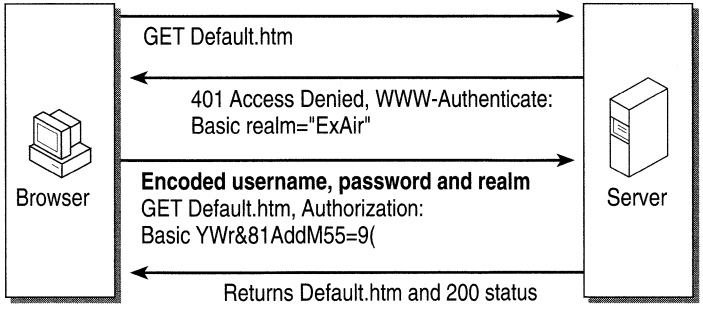


Figure 9.9 Basic Access to a Resource

Integrated Windows Authentication

Integrated Windows authentication—formerly known as both NT LAN Manager (or NTLM) and Windows NT Challenge/Response authentication—is more secure than Basic authentication. This authentication scheme works especially well in an intranet environment where users have Windows domain accounts.

In integrated Windows authentication, the browser attempts to use the current user's credentials from a domain logon. If those credentials are rejected, integrated Windows authentication will prompt the user for a user name and password by means of a dialog box. When integrated Windows authentication is used, the user's password is not passed from the client to the server. If a user has logged on as a domain user on a local computer, the user won't have to be authenticated again when accessing a network computer in that domain.

The user is not prompted for a user name and password for each HTTP request; rather, this will happen only when the cached credentials do not have sufficient permissions to access a specific page or file.

The Role of Negotiation

Prior to Windows 2000 Server, Windows security was limited to just NTLM, but now Windows 2000 Server supports both NTLM and Kerberos v5 authentication. In essence, integrated Windows authentication is NTLM or Kerberos v5. Rather than sending both an NTLM and Kerberos v5 challenge (random data produced by the server) to the client, Windows 2000 Server sends a Negotiate header. This allows the client and server to negotiate a suitable authentication protocol. A response (the challenge modified with user name and password information supplied by the client) is then sent.

Integrated Windows authentication has the following limitations:

- It cannot be performed through a firewall via a proxy.
- Currently, it is supported only by Microsoft® Internet Explorer 2.0 and later.
- It might not support delegation to other servers. In other words, the user's credentials cannot be passed on to another computer. For example, when a request comes in to IIS 5.0, the user account credentials cannot be passed to SQL Server on another Windows computer. However, this is only the case if NTLM is chosen as the authentication protocol during the negotiation phase; if integrated Windows Authentication is chosen, delegation is supported.

Integrated Windows Authentication Header Flow

Figure 9.10 shows the flow of HTTP headers when you use integrated Windows authentication.

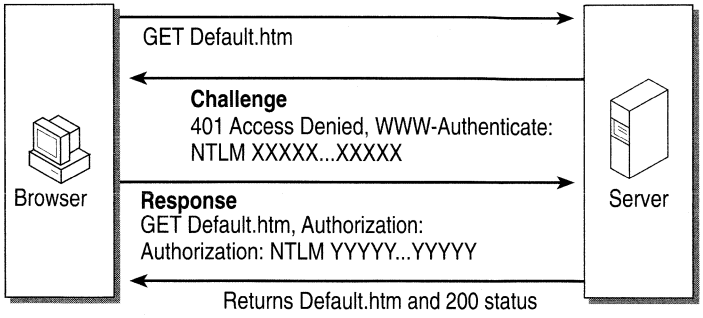


Figure 9.10 Integrated Windows Access to a Resource

Digest Authentication

Digest authentication addresses many of the weaknesses of Basic authentication. Most notably, the password is not in clear text when you use Digest authentication. In addition, Digest authentication can work through proxy servers, unlike integrated Windows authentication.

At the time of this writing, digest authentication is not a completed standard; it is still a draft. The version of digest authentication used in IIS 5.0 follows RFC2069, with some extensions from the IETF draft specification that can be found at http://www.ietf.org/.

Because Digest authentication is a challenge/response mechanism like integrated Windows authentication, passwords are not sent unencrypted, as in Basic authentication.

How to Configure Digest Authentication

In order for you to correctly use Digest authentication, the following must be set up on the server:

- Windows 2000 Server must be in a domain.
- A file called IISUBA.DLL must be installed on the domain controller. This is copied automatically during Windows 2000 Server setup.
- All user accounts must be configured to have the *Save password as encrypted clear text* option enabled, as shown in Figure 9.11. This is an option on each user object in the Active Directory. Setting this option requires the password to be reset or re-entered.

IIS 5.0 must be configured to use Digest authentication. See Figure 9.12.

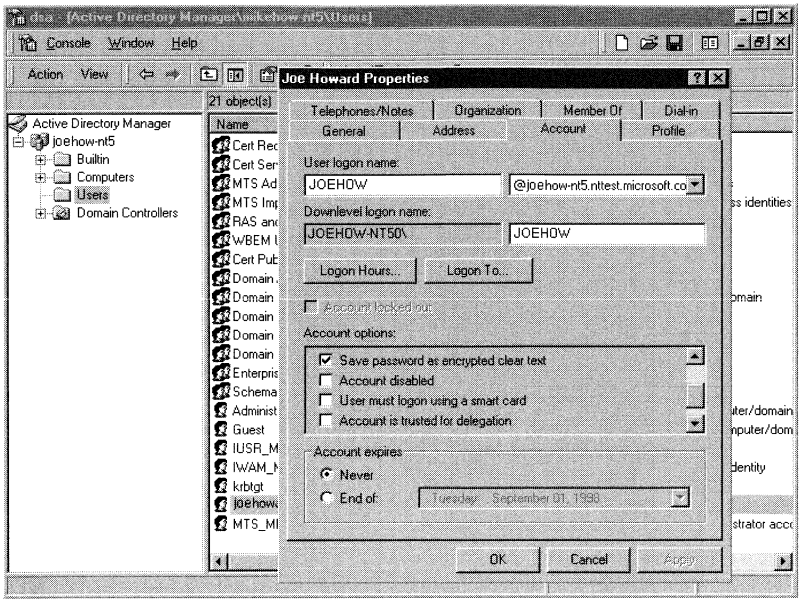


Figure 9.11 Setting the Save password as encrypted clear text Option

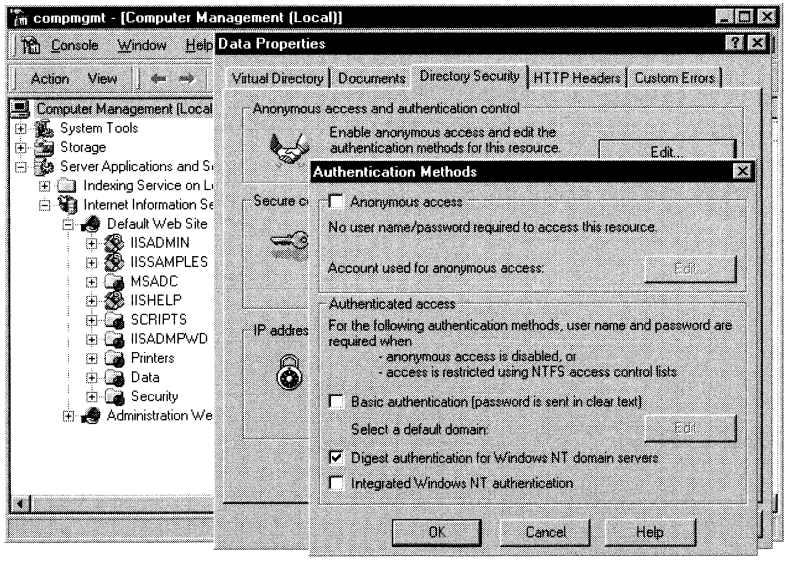


Figure 9.12 Setting IIS 5.0 to Support Digest Authentication

Digest Authentication Header Flow

Figure 9.13 shows the flow of HTTP headers when you use Digest authentication.

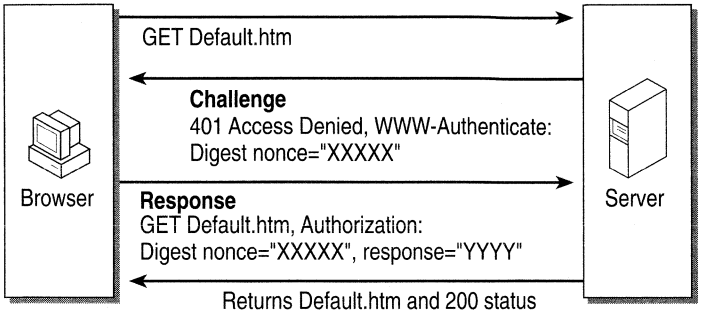


Figure 9.13 Digest Authentication Access to a Resource

Client Certificate Mapping

Traditionally, computer systems have used a centralized accounts database to manage users, their privileges, and their access controls. This technique has worked well and is readily understood. However, as systems become more and more distributed—with hundreds of thousands to even millions of users—this form of centralized control becomes unwieldy. The problem ranges from trying to verify an account against a database located on the other side of the Internet to administering a very large list of users.

Certificates can often simplify these problems. They can be widely distributed, issued by numerous parties, and verified by simply examining the certificate, without having to refer to a centralized database. Sometimes a server might need to refer to a central site in order to gather certificate revocation information, but this data is cached.

Existing operating systems and administration tools can only deal with accounts, not certificates. The simple solution—one that maintains the advantages of both certificates and user accounts—is to create an association (or mapping) between a certificate and a user account. This allows the operating system to continue using accounts, while the larger "system" and the user take advantage of certificates. In this model, a user presents a certificate; the system then looks at the mapping in order to determine which user account should be logged on. For more information about public key certificates, see "Public Key Infrastructure" in the *Distributed Systems Guide*.

Mapping a certificate to a Windows user account can be done in one of two ways:

1. With Microsoft® Active Directory™ directory service.
2. With rules defined in IIS 5.0.

   Note Windows Active Directory Mapping and IIS 5.0 Native Mapping are mutually exclusive service-wide. You cannot use rules defined in one form of mapping in the other mapping form.

What Is a Client Certificate?

Client certificates contain information that identifies the user, as well as information about the organization that issued the certificate. For example, a standard X.509 certificate contains at least the following:

- Version
- Serial number
- Signature algorithm ID
- Issuer name
- Validity period
- Subject (user) name
- Subject public key information
- Signature on the fields just listed

Figure 9.14 shows an example of a client authentication certificate in Microsoft® Internet Explorer 5:

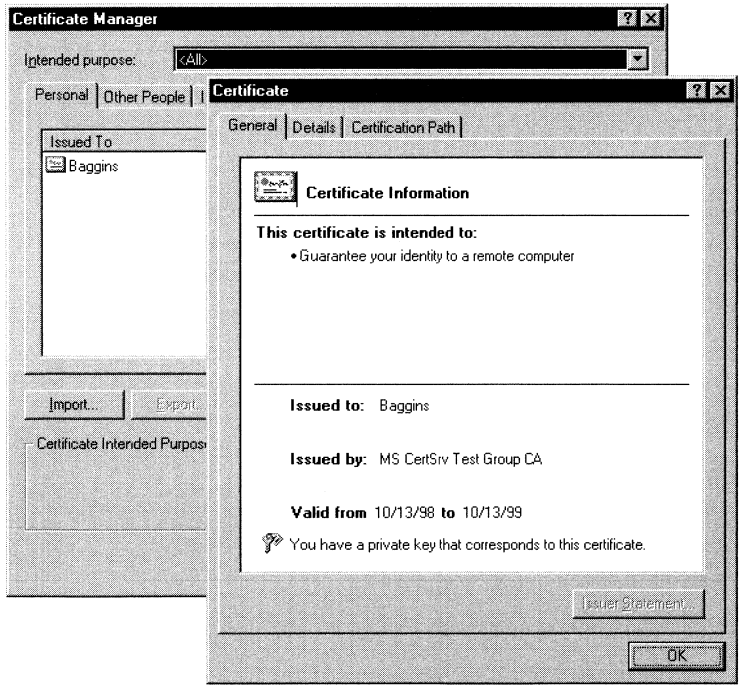


Figure 9.14 A Client Authentication Certificate

A user obtains a client certificate from a trusted third-party organization such as VeriSign (http://www.verisign.com) or Thawte Consulting (http://www.thawte.com). These organizations are usually referred to as certification authorities or CAs. You can find a more complete list of CAs at http://www.microsoft.com/security/ca/ca.htm.

If you have Microsoft® Windows® 2000 Certificate Services installed, your site can issue its own certificates to users on the intranet or to business partners.

A Brief Overview of Certificate Services

The role of Certificate Services is to issue and manage certificates, which are used in software security systems that employ public key technologies.

The role of Certificate Services is to create a CA that receives certificate requests from clients and servers, verifies the information in the request, and issues a corresponding X.509 certificate. The Certificate Services Manager administration tool enables you to administer Certificate Services.

The CA receives requests for new certificates over transports such as HTTP or e-mail. It then checks each request against predefined policies, sets optional properties for the certificate, and issues the certificate. With Certificate Services, administrators can add certificates to a Certificate Revocation List (CRL), as well as publish a signed CRL to a published file share or to the Active Directory on a regular basis.

Certificate Services supports issuing certificates for Secure/Multipurpose Internet Mail Extensions (S/MIME), and digital signatures for use in SSL and TLS.

Programmable interfaces are included, so developers can create support for additional transports, policies, and certificate properties and formats.

A Quick Detour: X.509, DER, PKCS Explained

You'll notice references to X.509, Distinguished Encoding Rules (DER) encoding, (Public Key Cryptography Standards) PKCS #7, and Base64 when you use or administer certificate-based solutions.

X.509 is the industry-standard certificate type. You can find more information at http://www.rsa.com/rsalabs/faq/html/5-3-2.html.

DER is a binary encoding format for certificates. You can find more information at ftp://ftp.rsa.com/pub/pkcs/doc/layman.doc.

PKCS #7, developed by RSA Data Security, is a binary format for defining encrypted and signed data, such as certificates. You can find more information at http://www.rsa.com/rsalabs/faq/html/5-3-3.html.

Base64 encoding is a text-based encoding system for binary data. It is the same coding scheme used in Basic authentication, and is defined in the Internet standard RFC1521.

IIS 5.0 Native Mapping

IIS 5.0 mapping is similar to the mapping capability that shipped with IIS 4.0. It allows the administrator to set up a rule or group of rules that determine how a client authentication certificate is mapped to a Windows user account, without requiring the use of Basic or any other form of authentication. There are two possible IIS 5.0 native mapping modes: one-to-one and many-to-one.

One-to-One Mapping

In one-to-one mapping, the administrator selects the client authentication certificate for mapping and enters the user name and password associated with the certificate. The form displayed on the client certificate in Figure 9.15 is Base64 encoded X.509. This is a standard client authentication certificate that is encoded using Base64, rather than binary. Base64 encoding enables older e-mail servers to handle the data easily, since many older e-mail servers cannot relay binary data.

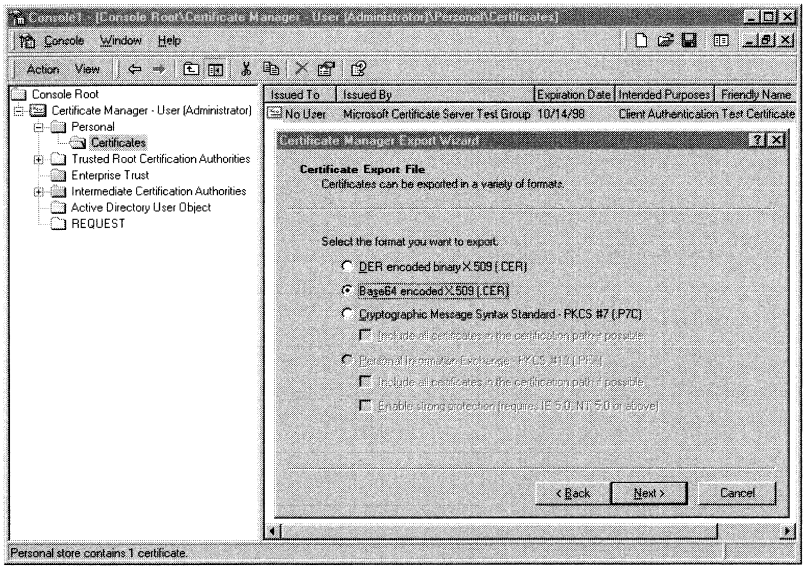


Figure 9.15 Using the Certificate Manager Export Wizard to Export a Client Authentication Certificate

You can also access a user's client authentication certificate from the Active Directory, as shown in Figure 9.16. A client authentication certificate is an optional **UserCertificate** property on the **User** object. The following Visual Basic code will access the **UserCertificate** object:

```
Dim oUser, vCert
Dim strName, strDN
StrName = "CN=Baggins"
StrDN = "CN=Users,DC=iis,DC=nttest,DC=microsoft,DC=com"
Set oUser = GetObject("LDAP://" & strName & "," & strDN)
vCert = oUser.userCertificate
Set oUser = Nothing
```

You can set a user's client authentication certificate by using the Directory Management administration tool. If Certificate Services is installed in the enterprise, the certificate is automatically added to the user's list of certificates in the Active Directory, when the client requests a certificate.

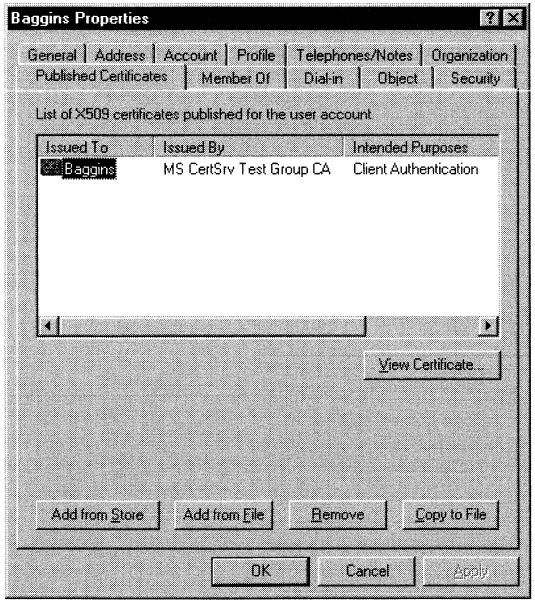


Figure 9.16 A User's Client Authentication Certificate in the Active Directory

Note To view or add a user's client authentication certificate in the Active Directory user interface, you must have Advanced Features enabled. To do this, load the Directory Management administration tool, select **View** and check the **Advanced Features** option.

Many-to-One Mapping

Many-to-one mapping is the mapping of many certificates to a single user account. For example, assume you have a partnership with an agency that provides temporary workers for your job openings. You would like to allow the agency personnel to view Web pages describing current job openings that are otherwise accessible only to company employees. The agency has its own trusted CA that it uses to issue certificates to its employees. After installing the agency CA's *root certificate* (a certificate at the top of the certification hierarchy) that is trusted in your enterprise, you can set a rule that maps all certificates issued by that CA to a single Windows account. You then set NTFS access rights so that the account can access the Web pages. Typically, you will choose a Windows account name that reflects the role or company name of the trusted company, for example TempAgency.

Now, when employees from the agency connect to the agency's Web server and provide their certificates, they are mapped to the same account and can access those pages and no others. This is nice from an administrative viewpoint because the agency can now issue certificates and manage its users, without requiring you to do any more work.

Windows Active Directory Mapping

A Windows 2000 Active Directory installation automatically sets up one-to-one mappings for the certificates it issues. Any logon certificate should automatically be mapped to the account it represents.

If you wish to perform many-to-one mapping or use a CA other than Certificate Services, you will need to use the Name Mappings option in the Active Directory management tool. For more information, refer to the Windows documentation.

Certificate Trust Lists

When using certificate mapping you will need to configure IIS 5.0 to only trust a limited number of CAs. In Windows 2000 Server and IIS 5.0, this is performed through certificate trust lists (CTLs).

A *CTL* is a set of certificates determined as trustworthy by an administrator. For a client authentication certificate to be used successfully, it must be signed (issued) by a trusted CA listed in the CTL.

For example, if you only trusted certain certificates issued by two CAs, such as ExplorationAir Corp. CA and VeriSign, then you could define a CTL that only lists these CAs. If a user attempts to connect to your Web server using a client authentication certificate issued by any other CA, access will be denied.

Caching and the Windows Audit Log

By default, IIS 5.0 caches security tokens for accounts that are logged on locally, such as IUSR_*computername*. You might not see multiple logon entries in the audit logs because the token is already accessible to IIS 5.0. The default cache time is 15 minutes.

The following ADSI code, callable from Visual Basic, Microsoft® Windows® Script Host (WSH), or Active Server Pages (ASP), will change the cache time for the default Web server (Web server 1), on a computer called Baggins, to two minutes.

```
Dim oIIS
Set oIIS = GetObject("IIS://Baggins/W3SVC/1")
oIIS.PasswordCacheTTL = 2
oIIS.SetInfo
Set oIIS = Nothing
```

Web Authentication Summary

Table 9.3 shows a brief summary of all the possible Web authentication schemes built into IIS 5.0:

Table 9.3 Web Authentication Schemes in IIS 5.0

Scheme	Requires SSL	Works with Internet Explorer 4	Works with Internet Explorer 5	Works with Other Browsers	Comment
Anonymous	No	Yes	Yes	Yes	No authentication
Basic	No	Yes	Yes	Yes	Insecure: consider using SSL to protect user name/password
Integrated Windows	No	Yes	Yes	No	Does not work through proxy servers
Digest	No	No	Yes	Varies	Requires Active Directory
Certificate Mapping	Yes	Yes	Yes	Varies	Very scalable and secure, but a little cumbersome to configure

FTP Authentication

Users must log on in order to gain access to the FTP server. The IIS 5.0 FTP services can use the Windows account database to authenticate users logging on. However, all FTP transmissions are in clear text, thus exposing user names and passwords, as seen in Figure 9.17.

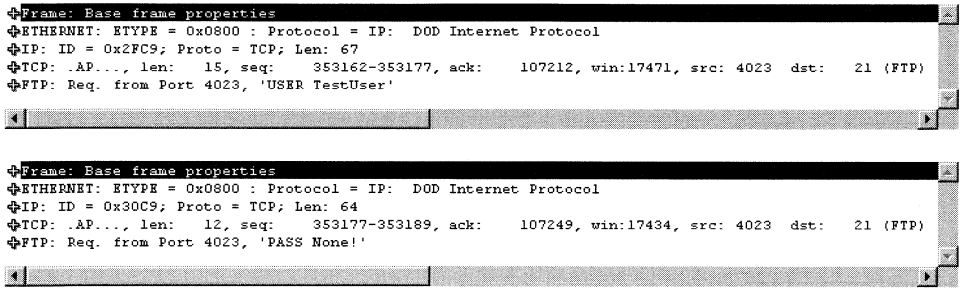


Figure 9.17 Two Packets of FTP Data, Showing the User Name and Password in Clear Text

In order to eliminate exposed passwords, you can configure your FTP server to permit anonymous logons. This type of logon requires the user to type "anonymous" as the user name and the user's Internet e-mail address as the password. Anonymous users can gain access to files under the IUSR_*computername* account.

You can also allow anonymous-only logons to the FTP service. Anonymous-only logons are useful because they prevent real passwords from being revealed on a public network. In IIS 5.0, the FTP service is configured for anonymous-only access by default.

Extending IIS 5.0 Security

Although it is possible to extend IIS 5.0 security schemes by using Internet Server Application Programming Interface (ISAPI) filters, ASP pages, or Microsoft® Component Object Model (COM) components, you should seriously consider the tradeoffs.

The advantage of using alternate authentication mechanisms is flexibility; if you write your own authentication code, you have complete control of those mechanisms. For example, you might decide to check the existence of a user account in an online database.

The disadvantage is that you now have security policy and enforcement in multiple places. If you update your corporate security policy, such as who is allowed access to what resources, then you may need to update the Web server as well. Eventually, this will become difficult to maintain and could lead to security vulnerabilities.

Extending IIS 5.0 Security with ISAPI Filters

The most common way to extend IIS 5.0 security is to use ISAPI filters. These allow you to receive notifications of various events, such as authentication, during the processing of HTTP requests. Filters, implemented as DLLs, are loaded when you start IIS 5.0, and are kept in memory until you shut down IIS 5.0. Once a filter is loaded, it indicates the events that IIS 5.0 should send notification about. Subsequently, IIS 5.0 will notify the filter each time one of the registered events occurs.

If you want to write an ISAPI filter in order to perform authentication, you will need to review the **SF_Notify_Authentication** event in the IIS 5.0 online product documentation.

Extending IIS 5.0 Security with ASP Code

You can also create custom security schemes by using ASP code. For example, you could programmatically deny access to a page with the following code used as part of a POST. In this example, the original page prompted the user to enter a user name and password in a FORM:

```
<%
   Dim strName, strPassword
   strName = Request.Form("Name")
   strPassword = Request.Form("Password")
   'Do a database lookup and set fAllowAccess based on the query.
   If fAllowAccess = False Then
      Response.AddHeader("401 - Access Denied")
   End If
%>
```

The assumption in this case is that there is a database with user names and passwords, which are both used as the basis to perform the security check.

Please note that this mechanism is very insecure. First, you must POST the data over a secure channel by using SSL; otherwise, intruders might be able to gather the user name and password as it goes in plaintext from the browser to the Web server. Also, if this page is circumvented, no access check can be performed whatsoever.

File and Directory Security

IIS 5.0 uses Windows 2000 Server security throughout, including DACLs in the Windows File System, NTFS. It is recommended that you place your data files on an NTFS partition because NTFS provides security and access control for your data files.

DACLs grant or deny access to the associated file or folder by specific Windows user accounts, or groups of users. When an Internet service attempts to read or execute a file on behalf of a client request, the user account offered by the service must have permission.

You can use IIS 5.0 virtual directory access control combined with Windows accounts and NTFS file ACLs in order to configure access to specific files within a Web site. After a user is authenticated for the IIS 5.0 virtual directory, IIS uses the context of the requesting user (Anonymous or specific) to gain access to the NTFS file based on the user account, user rights policy, and file permissions.

It is possible to use the Windows Interactive user and Network user accounts in order to provide broad access control for files available to IIS 5.0. The Interactive user is a special user account representing any user who is logged on interactively. In other words, this is someone who has the Log on locally privilege and has been logged on locally. The Network user account is similar, but applies to users with the Network logon privilege.

- If the Interactive user is given Special (Read) permission for a file, the anonymous user (if Allow IIS to control password is disabled) or any client authenticated by Basic authentication will be able to access the file. This is because these accounts are being logged on locally (in other words, they are interactive).
- If the Network user is given Special (Read) permission for a file, the anonymous user (if Allow IIS to control password is enabled) or any client authenticated by Digest authentication or integrated Windows authentication will be able to gain access to the file. This is because these accounts are logged on as network accounts.

Directories Containing DLLs Needed by the System

Since programs are usually made up of an executable and many DLLs, you must be able to execute the DLLs as well as the main executable. Set directories and files to Read (RX) access for the IUSR_*computername* account and for all accounts that have specific authentication for the following directories:

- %WinDir%\System32 (and all subdirectories)
- \Program Files\Common Files (and all subdirectories)

Directories Corresponding to Web Virtual Directories

- **Special (R):** If the directory contains only static .htm or .asp files.
- **Special (X):** If the directory contains only executable files (for example, .dll).
- **Read (RX):** If the directory is a mix of readable and executable files.
- **Special (RW):** If the directory contains a file database (for example, .mdb).
- **Change (RWXD):** If developers need to be able to modify or delete the files. (This is not recommended due to the security risk.)

You will need to set different permissions on files for individual users, depending on the authentication schemes you use:

- **IUSR_*computername*:** If Anonymous access is being used.
- **Interactive:** If you are allowing access to all users authenticated by Basic authentication and if the default logon method is Log on locally.
- **Network:** If you are allowing access to all users authenticated by integrated Windows authentication or Digest authentication.
- **Specific users and groups:** Ensure that they have the correct user rights profiles.

> **Note** If you are not using the server for any other file access, restrict access to all files to only Administrators, System, and—in less secure environments—Power Users.

Virtual Directory Security

In order to publish from any directory not contained within your home directory, create a *virtual directory*. A virtual directory is a directory that is not contained in the home directory, but which appears to client browsers as though it were. Virtual directories have a small number of options for controlling access and content control:

- Read
- Write
- Log Access
- Directory Browsing
- Index This Directory
- Microsoft® FrontPage® Web (Web site only)

The default settings are Read, Log Access, Index This Directory, and FrontPage Web (for a Web site only).

> **Note** Write access can be performed only with a browser that supports the PUT feature of the HTTP 1.1 protocol standard.

IIS also supports application settings that determine how executable content such as ASP pages operate. The three options are:

- **None:** Doesn't allow any programs or scripts to run in this directory.
- **Script:** Enables applications mapped to a script engine to run in this directory, without having the Execute permission set. Use Script permission for directories that contain scripts in ASP pages, Internet Database Connector (IDC) scripts, or other scripts. Script permission is safer than Execute permission because you can limit the applications that can be run in the directory.
- **Execute:** Allows any application to run in this directory, including applications mapped to script engines and Windows binaries (.dll and .exe files). It is suggested that you use this option with care.

If a virtual directory is on an NTFS drive, the settings for the directory must match these application settings. If they don't, the most restrictive settings take effect. For example, if you give a directory Write permission but only give a particular user group Read access permissions in NTFS, the user group will not be able to write files to the directory because the Read permission is more restrictive.

Using the Permissions Wizard

The combination of NTFS DACLs and Web Permissions can be somewhat daunting at first. The Permissions Wizard in IIS 5.0 allows you to set the root of a virtual server to a known state, thus alleviating you of most of the burden. The wizard does this by asking questions and setting all the appropriate permissions based on your answers. If you still have problems accessing your site, you might need to troubleshoot your permissions.

Accessing Resources on Network Servers

Windows NT 4.0 does not allow delegation of security credentials. That is, it does not allow a server to pass your security information onto another server. Windows 2000 Server, in contrast, allows credential delegation in certain situations.

A common scenario is when IIS 5.0 is trying to access a resource on a network server—such as an Access database—while using integrated Windows authentication, Anonymous, or Basic authentication, regardless of particular DACL issues. See Figure 9.18.

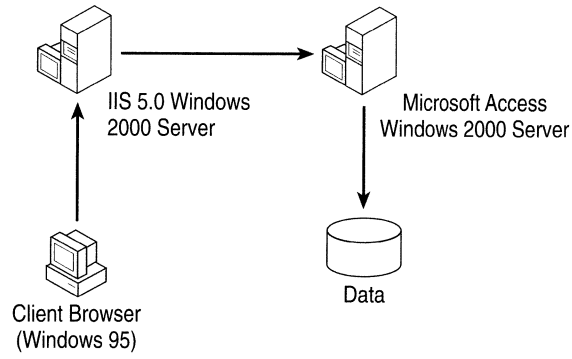


Figure 9.18 Accessing a Remote Access Database from a Browser

Table 9.4 shows which authentication protocols allow access to remote resources:

Table 9.4 Authentication Protocols That Can Access Remote Resources

Protocol	Access Remote Resource?	Comment
Anonymous (IIS control password enabled)	No	Windows subauthentication DLL does not allow this.
Anonymous (IIS control password disabled)	Yes	Can make one "hop" onto a remote server.
Basic	Yes	Can make one "hop" onto a remote server.
Integrated Windows	Depends	No, if NTLM is used; yes, if the Kerberos v5 authentication protocol is used. If Kerberos v5 is used throughout, then the request can be fully delegated to many remote servers.
Digest	No	Windows subauthentication DLL does not allow this.
Certificate Mapping (IIS 5.0 mapper)	Yes	Can make one "hop" onto a remote server.
Certificate Mapping (Windows Mapper)	No	Windows mapping does not allow access to the user's password, so the password cannot be delegated.

Trusted for Delegation

To allow delegation of security credentials, you must also set the Computer is trusted for delegation option on each computer that will be used by the Web application.

To set the Computer is trusted for delegation option

1. On the **Tools** menu, click **Active Directory Users and Computers**.
2. Click the **domain name** node to expand it.
3. Click the **Computers** node to expand it.
4. Right-click on the computer in question.
5. Select **Properties**.
6. Select **Computer is trusted for delegation**.

Perform these steps on the Domain Controllers node also.

You can override this option on a per-user account basis. For example, it is considered bad security practice to enable account delegation for the Administrator account, because this account is a highly trusted account.

To override the Computer is trusted for delegation option

1. On the Tools menu, click **Active Directory Manager**.
2. Click the **Users** node to expand it.
3. Right-click on the user account in question.
4. Select **Properties**.
5. Click the **Account** tab.
6. Deselect **Account is sensitive, do not delegate** in the **Account** options.

Pass-through to UNC-Shared Web Content

In IIS 5.0 you can set the location of a virtual directory to be on a network computer by using a UNC (Uniform Naming Convention) name such as \\MyServer\WebDocs. See Figure 9.19. (Due to delegation issues in IIS 4.0, you had to provide a user name and password in order to connect to the remote share.)

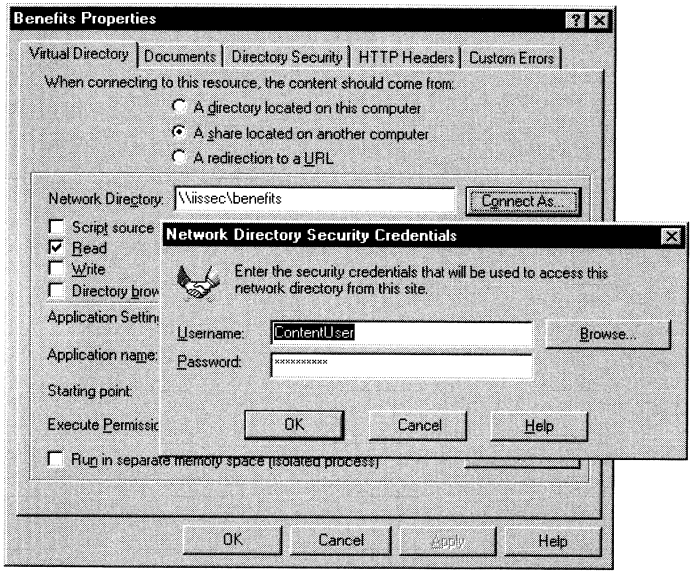


Figure 9.19 Setting the User Name/Password to Access Web Content on a UNC resource

This method has some limitations; most notably you cannot use DACLs correctly on the network share. This is because you are essentially using the user account provided in the IIS 5.0 administration tools and not the real user account in order to access the network resource.

IIS 5.0 introduces the ability to pass user information through user accounts, if the underlying authentication scheme supports credential delegation. However, this ability is not in the user interface. The following script will allow user credentials to be passed through to the remote resource on a virtual directory called *Content* located on the default Web server:

```
Dim oVdir
Set oVdir = GetObject("IIS://localhost/W3SVC/1/Root/Content")
oVdir.UNCAuthenticationPassThrough = True
oVdir.SetInfo
Set oVdir = Nothing
```

See Table 9.3 for a list of authentication schemes and whether they will support delegation. If this pass-through option is set and if the authentication scheme you use does not support delegation, the account defined in the Network Directory Security Credentials is used instead.

Secure Communications with SSL and TLS

As previously mentioned, SSL and TLS are ways means to authenticate, detect tampering, and encrypt communication between a server and a client. IIS 5.0 will attempt to use TLS before attempting to use SSL, because it is more secure.

In order to use SSL or TLS, the Web server must have a server authentication certificate issued by a trusted CA.

Authentication and Trust

Checking that the server's certificate is valid and was issued by a trusted CA gives a high degree of confidence that the server you are communicating with is the correct one. Trusted in this case means trusted by both the server and the client.

You trust the third-party CA if you have the issuer's root certificate in your client software, such as your browser. For example, if the server you wish to communicate with has a certificate issued by VeriSign, then you must have the appropriate VeriSign root certificate in your browser.

To check for root certificates

1. Open Internet Explorer.
2. On the **Tools** menu, select **Internet Options**.
3. Click the **Content** tab.
4. Click the **Certificate** button.
5. Select the **Trusted Root Certification Authorities** tab.

You will then see a list of company root certificates. This list determines which CAs you trust.

> **Note** If you do not trust or do not recognize a company named in this list, you should remove the certificate.

Encryption

Various cryptographic algorithms provide tamper detection and encryption; SSL and TLS can use protocols such as RC2, RC4, DES, 3DES, SkipJack and IDEA for encryption, and MD5 or SHA1 for tamper detection. For more information about these encryption protocols, see the books mentioned in "Additional Resources" at the end of this chapter. For more information about SSL and TLS, see http://www.ietf.org/ids.by.wg/tls.html.

Enabling SSL

In the context of a Web server, SSL is most effectively used when encrypting only communications that contain private data, such as credit card numbers, phone numbers, or company records. Because SSL uses complex encryption, and because encryption requires considerable processor resources, it takes much longer to retrieve and send data from SSL-enabled directories. Therefore, you should place only those pages that will contain or receive sensitive information into your SSL-enabled directory. Also, keep the pages free of elements that consume resources, such as images.

To use SSL on IIS 5.0

1. Request and install a server certificate. You can acquire a server certificate from a trusted third party such as VeriSign, or from Certificate Services by using the Web Server Certificate Wizard in IIS 5.0 to request a server certificate. (Web Server Certificate Wizard is the replacement for Key Manager in previous versions of IIS.)
2. Enable the appropriate settings in the **Secure Communications** dialog box. This dialog box will not appear unless you have a server certificate installed.

   Note When requesting a certificate from Certificate Services, you must decide whether you want the private key to be exportable or not. (The Web Server Certificate Wizard in IIS marks the keys as exportable.)

   If you want to be able to back up your key, you must have an exportable private key. However, an exportable key is sometimes viewed as a security risk because the key could be compromised, and having access to the private key means an attacker can pose as the real user.

A Web server can only have one server certificate assigned to it. For example, reskit.microsoft.com can have a certificate labeled reskit.microsoft.com; it cannot have another certificate labeled reskit-10.microsoft.com. This is because a certificate is an identity and a Web server, just as a real person cannot have more than one identity.

IIS 5.0, SSL, and Fortezza

Fortezza® is a registered trademark of the National Security Agency (NSA). It describes a family of computer security products, most notably PCMCIA-based cryptography cards, designed to meet the needs of U.S. federal agencies.

Fortezza is a piece of a broader Microsoft® Federal Security Initiative. Through this initiative, Microsoft is demonstrating its commitment to meeting the ongoing security needs of its federal customers. The official Web site for technical Fortezza information is http://www.armadillo.huntsville.al.us.

Like many other forms of encryption technology in the Windows operating system, Fortezza is implemented as a Cryptography API Security Provider (CSP). However, this CSP does not ship with Windows 2000 Server or IIS 5.0, but it is available from http://www.spyrus.com.

Configuring IIS 5.0 and Fortezza

SSL has a Fortezza mode that is of benefit to IIS 5.0. All Fortezza cards (PCMCIA cards) contain user certificates that authenticate the card user in much the same way that server or client certificates work in IIS 5.0. These user certificates must be copied over to a secure store on the computer where the card logs on. This makes the certificates available to IIS 5.0. In order to configure IIS 5.0 to use Fortezza, you must be using a domestic version of Windows 2000 Server.

To configure IIS 5.0 for Fortezza

1. Install the card reading equipment and its drivers. For information, see the card reader documentation.
2. Install the Cryptographic API Service Provider (CSP) provided by the equipment supplier. For information, see the card reader documentation.
3. Run the command line utility %windir%\system32\inetsrv\Fortutil.exe.

The Fortutil.exe utility provides functions that can install, confirm, and delete the card certificate and other associated information. To enable these features, type the appropriate commands at the command line, as shown in Table 9.5:

Table 9.5 Commands for Enabling Fortutil.exe Features

Action	Command	Parameters
Add Certificate	**Fortutil /a**	Web site name; card serial number; PIN; card personality
Confirm Certificate	**Fortutil /q**	Web site name
Delete Certificate	**Fortutil /d**	Web site name
Get Help	**Fortutil /?**	None

Certificate Revocation Lists

A CRL (often pronounced "KRILL") is a list of certificates that have been revoked before their scheduled expiration date. A certificate might need to be revoked if its associated private key has been compromised, or if the user no longer works for the company in question.

You can configure Internet Explorer 5 and IIS 5.0 to check CRLs.

To enable CRL-checking in Internet Explorer 5

1. On the **Tools** menu, select the **Internet Options** menu option.
2. Select the **Advanced** tab.
3. Click the **Security** node to expand it.
4. Check the **Check for server certificate revocation** option.

CRL-Checking in IIS 5.0

CRL-checking is enabled in IIS 5.0 by default. The following Visual Basic code shows how to enable or disable CRL-checking on the default Web server.

```
Set oIIS = GetObject("IIS://localhost/W3SVC/1")
oIIS.CertCheckCRL = True ' False to disable CRL checking
oIIS.SetInfo
Set oIIS = Nothing
```

Certificates and CryptoAPI

Unlike previous versions of IIS, IIS 5.0 uses Microsoft® CryptoAPI (CAPI) 2.0 for all encryption and certificate functions. You can use all the certificate tools that are built into Windows 2000 Server in order to perform routine certificate work.

For normal SSL/TLS work, the server certificates are stored in the Local Computer certificate store. For Fortezza, the certificate is held on the PCMCIA card (and then copied to the local computer using Fortutil.exe).

For more information about CryptoAPI, see http://www.microsoft.com/security/default.asp.

How to Back Up Your Web Server Certificate

Because IIS 5.0 leverages Windows security and security tools, you can use the Certificate Manager tool in Windows 2000 Server to export your IIS 5.0 certificates. Historically, the Key Manager tool performed a backup; however, Key Manager is no longer used by IIS 5.0.

First you need to make sure you are looking at the correct Local Computer certificate store. If you are not, you will have to set this up before you can export certificates.

To view the correct certificate store

1. **Open** Microsoft® Management Console.
2. Select the **Add/Remove** Snap-in option on the **Console** menu.
3. Click **Add**.
4. Select the **Certificate Manager** tool.
5. Click **Add**.
6. Select the **Computer account** option.
7. Select the **Local computer** option.
8. Click **Finish**, **Close**, then **OK**.

You are now looking at the correct certificate store.

To export a certificate

1. Click the **Certificate Manager** (Local Computer) node to expand it.
2. Click the **Personal** node to expand it, and then expand the **Certificates** node.
3. Right-click the certificate in question.
4. Select **Export** from the **All tasks** menu option.
5. Click **Next**.
6. Select **Yes** to export the private key.
7. Choose the **PKCS #12** format and enable strong encryption.
8. Type and confirm your password.
9. Enter a file name.
10. Click **Next**, then click **Finish**.

How to Restore Your Web Server Certificate

You can restore certificates by simply double-clicking the PKCS #12 certificate file just created.

How Access Is Determined

When a user attempts to access a resource through IIS 5.0, a number of technologies come into play. Figure 9.20 provides an overview of this process:

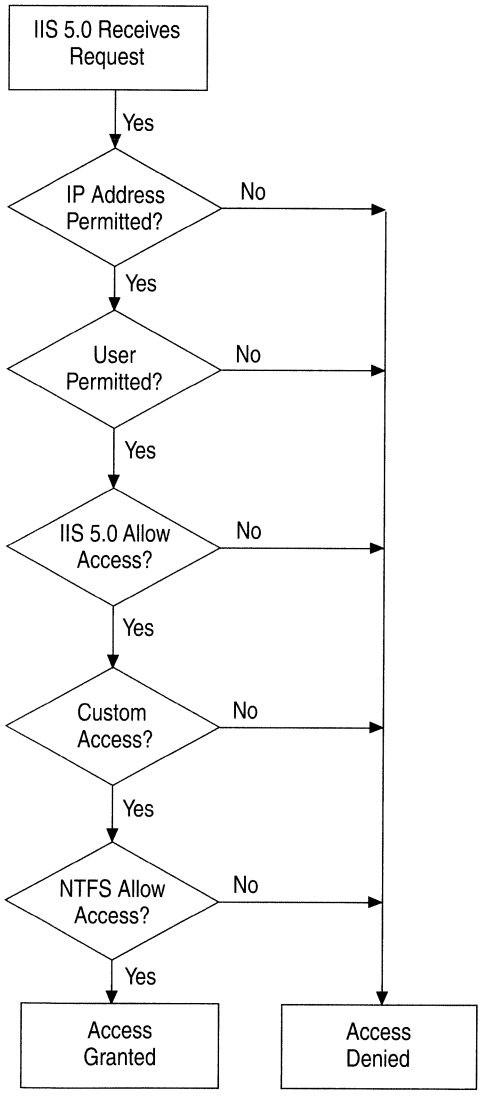


Figure 9.20 IIS 5.0 Access Logic

In the interest of simplicity, Figure 9.20 shows many paths leading to Access Denied. In fact, some paths yield a 401 or 403 HTTP error. A real Access Denied message can prompt the user to re-enter the user name and password; a 401 or 403 cannot.

IP Address Access Control

IIS 5.0 can be configured to grant or deny access to specific IP addresses. For example, you can prevent a company or individual from accessing your site by listing the appropriate IP address, IP address range, or Domain Name System (DNS) name (IIS 5.0 checks the IP address first, then the DNS name). You can even prevent entire networks from gaining access to your server. Conversely, you can allow only specific sites or IP addresses to have access to your service.

If IIS 5.0 is configured to allow access by all IP addresses except those listed as exceptions to that rule, then access is denied to any computer with an IP address included in that list. Conversely, if IIS 5.0 is configured to deny all IP addresses, access is denied to all remote users except those whose IP addresses have been specifically granted access. IP address access restrictions are available for each of the IIS 5.0 services.

When controlling access by IP address, be aware that many Web users will be passing through a proxy server or a firewall. The incoming connection to your Web server will appear to have originated from the proxy server or firewall itself (in other words, the IP address of the originator will be that of the proxy server or firewall). This might be useful in a corporate network as an added security measure, in order to prevent access by anyone from outside your IP address domain.

User Access Control

There are five criteria for a successful logon:

- Valid user name and password supplied
- Windows account restrictions, such as time of day allowed to log on
- Account is not disabled or locked out
- Account password has not expired
- The applicable logon policy (for example, Log on locally or Access this computer from the Network)

By default, on a server running Windows, ordinary users (in other words, those who don't belong to the Administrator group) do not have the Log on locally user right. If FTP or WWW Basic Authentication is used, IIS 5.0 attempts to log on the user as a local user by default.

IIS 5.0 Access Control

Once a user has been granted access, the server examines both the URL and the type of request. It then checks the permissions and the SSL Client Authentication Certificate.

For WWW service, the request can indicate a Read, Write, Execute, or Script action. The applicable WWW virtual directory must have the appropriate permission enabled. Otherwise, the WWW service returns a "403.x: Access Forbidden" error, where "x" represents the type of access attempted.

IIS 5.0 might require a valid client authentication certificate before access to a resource is permitted. If such a certificate is not passed to the server, IIS 5.0 will return a "403.7: Forbidden—client certificate required" error.

Also note that the certificate might not be valid. For example, it could have expired, or the CA that issued the certificate might not be trusted.

Custom Authentication

Custom authentication means that you must create your own authentication mechanisms such as ISAPI authentication filters, ASP pages, or Common Gateway Interface (CGI) applications. The techniques involved with creating custom authentication schemes are many, varied, and thus beyond the scope of this chapter.

NTFS File System Permissions

IIS 5.0 attempts to gain access to the specified resource (usually a file) by using the security context of the authenticated user. For anonymous access, this is typically the IUSR_*computername* account. If authentication has been performed, it will be a valid Windows user account.

Troubleshooting Permissions

As you can see, access to resources such as files and databases can be a complex issue when you are securing your Web server. Site administrators often work blindly when trying to troubleshoot access problems. Monitoring how the server is being used is a good place to start, which requires you to set up auditing.

The following outlines how to set up auditing and logging for troubleshooting.

To set up auditing for user logons and file access

1. Open the **Global Policy Editor** for the computer or domain in question.
2. Go to **Computer Configuration**, **Windows Settings**, **Security Settings**, **Local Policies** and **then Audit Policy**.
3. Set **Audit successful attempts** and **Audit failed attempts** to **Yes** under the **Audit Logon Events** category.
4. Set **Audit failed attempts** to **Yes** under the **Audit Object Access** category.

To choose the file to audit

1. Open Windows Explorer.
2. Select the file or folder that you want to audit.
3. Right-click the file or folder and select **Properties**.
4. Click the **Security** tab and the **Advanced** button, and then click the **Auditing** button.
5. Click the **Add** button.
6. Select **Everyone**, click **Add**, and then click **OK**.
7. Select **Successful** and **Failed** for the following:
   - Traverse folder/Execute file
   - List folder/Read data
   - Create files/Write data
8. Click **OK**.

Much of this process is shown in Figure 9.21.

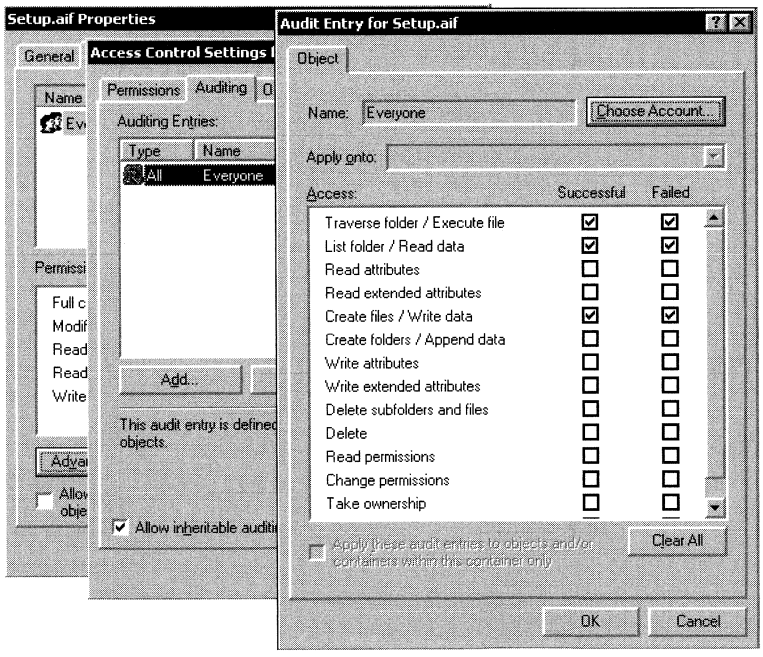


Figure 9.21 Auditing Access to a Single File

To enable Web Logging

1. **Open** the **Computer Management** administration tool.
2. Click **Server Applications and Services** to expand it and click IIS 5.0 to do the same.
3. Select the Web server in question.
4. Right-click on the Web site and select **Properties**.
5. Select **Enable Logging**.
6. Click the **Properties** button.
7. Click the **Extended Properties** tab.
8. Select (at least) the following: **Date**, **Time**, **Client IP Address**, **User Name**, **Method**, **HTTP status**, and **Win32 status**.
9. Click **OK** to exit the logging properties.
10. Click **OK** to exit the Web site properties.
11. Now that basic auditing is in place, clear any cached logon information (remember, for performance purposes IIS 5.0 can cache logon information) by typing the following at the command prompt:

    NET STOP IISADMIN /Y This stops all IIS 5.0 services.

    NET START W3SVC This starts the Web service, if installed.

    NET START MSFTPSVC This starts the File Transfer Protocol (FTP) service, if installed.

    NET START NNTPSVC This starts the Network News Transfer Protocol (NNTP) service, if installed.

    NET START SMTPSVC This starts the Simple Mail Transfer Protocol (SMTP) service, if installed.
12. Open the **Event Viewer**.
13. Clear the Security log by right-clicking the **Security** log and selecting **Clear all events**.

Troubleshooting the Web Server

After all these steps have been performed, try to access the resource from a browser. If you refresh the security log in the Event Viewer, a series of audited events will be listed. Examine the security log entries and answer these questions:

- Are there any access-denied errors in the audit log?
- What account is being logged on? Is it what you expected?
- Does this account have access to the file in question? You can check this by looking at the file access entries in the audit log.
- Does this account have the correct privilege to log on (Log on locally or network logon)?
- Does this account have a deny-logon privilege (for example deny logon locally or deny network logon)?
- Are there any 401s or 403s in the W3C log? If so, why are they there?

The following are some tips regarding permissions and IIS 5.0:

- If Anonymous authentication is turned on, this will usually be used before other authentication schemes.
- If Anonymous authentication is turned on and the IUSR_*computername* account is not allowed access to the resource in question, you might see a user name/password dialog box appear in the browser.
- If you are using Anonymous authentication, check whether the anonymous account is valid (correct password and logon times) and enabled. Right-click the virtual server in question, and select **Properties**, **Directory Security**, **Anonymous Access** and **Authentication Control**, **Edit**, and **Edit**.
- If you are accessing data from within an ASP file, the account being used must have access to the Microsoft® ActiveX® Date Objects (ADO) and ODBC directories.
- If you are using a page count component or a component that accesses files or the registry, the account logging on must have write and possibly delete access to the file or registry entry that it uses to persist its count.
- Any account used for Anonymous authentication when Allow IIS to control password is disabled must have the Log on locally privilege.
- Any account used for Anonymous authentication when Allow IIS to control password is enabled must have the Network logon privilege.

- Any account used for Basic authentication must have the Log on locally privilege.
- Any account used for integrated Windows authentication or Digest authentication must have the Network logon privilege.
- You can change the privilege type in the items just mentioned to Batch or Network, by using the **LogonMethod** ADSI property. The following ADSI code snippet shows how to do this:

```
Dim oIIS
Const LOGON_LOCAL = 0x0
Const LOGON_BATCH = 0x1
Const LOGON_NETWORK = 0x2
Set oIIS = GetObject("IIS://Baggins/W3SVC/1")
oIIS. LogonMethod = LOGON_BATCH
oIIS.SetInfo
Set oIIS=Nothing
```

- Accessing a network computer from an IIS 5.0 computer might fail, because Windows 2000 Server might not be configured to pass the user's security information to the other computer. To verify the configuration, perform the auditing steps (listed earlier) on the network computer in question, as well as on the IIS 5.0 computer.
- Microsoft® Internet Explorer 4.0 or later can be configured to support only certain authentication protocols. You can set these protocols by going to the **Edit** (Internet Explorer 4.0) or **Tools** (Internet Explorer 5) menu and selecting **Internet Properties**, **Security**, **Custom**, **Settings**, and **User Authentication**. If you select **Anonymous Logon**, the browser will not support any authentication schemes other than Anonymous.
- Check the IP and Domain Restrictions. Right-click the virtual server in question, and select **Properties**, **Directory Security**, **IP Address and Domain Name Restrictions**, and **Edit**.
- If you are using <!-- #include --!> statements, errors might be caused, since access is denied on the included file, but not on the main file. For example, if Default.asp includes Tools.asp, but Tools.asp has a restrictive DACL, an error may be reported on Default.asp, even though the logged-on account has access to it. To check for this, you might wish to temporarily comment out #include statements until you have successfully resolved the situation.

An End-to-End Troubleshooting Example

In the following scenario, a travel site called Exploration Air has developed a Web-based employee benefits application that uses IIS 5.0. The application can be accessed from the Internet and corporate intranet. Because the application is open to the Internet, security must be high.

All employees are validated before gaining access to the server and all data is encrypted to prevent people from "sniffing" the data as it moves across the Internet. The benefits application in this scenario deals with medical, dental, and legal assistance, as well as stock options, stock purchase plans, investment, and relocation benefits.

To access these resources, the Web application requires data from several sources:

- A Microsoft® SQL Server 7.0 database that contains all the corporate online benefits data. This is a complex database schema including 181 tables, 27 stored procedures, and 12 triggers. The database is approximately 6.5 gigabytes (GB) in size and grows about 250 MB per month.
- A legacy Oracle database on UNIX that contains the original Human Resources information, including payroll. For the purposes of this application, it is read-only and is used to verify the user.
- The IIS Configuration Store that is used to gather configuration details about the server, which it then displays on the benefits homepage.
- A "hit count" file that lists the number of times the home page has been accessed.

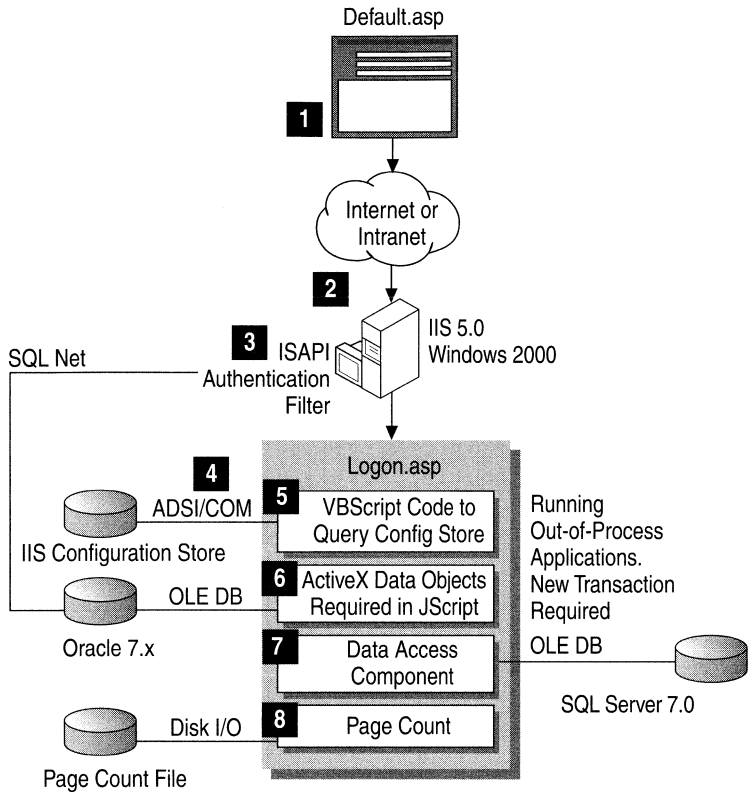


Figure 9.22 Anatomy of Web Application Benefits at Exploration Air

This scenario describes eight main steps, which are depicted in Figure 9.22:

1. The user logs on.
2. An ISAPI authentication filter is called.
3. IIS 5.0 attempts to authenticate the user.
4. IIS 5.0 loads the logon page Logon.asp.
5. Logon.asp attempts to read data from the configuration store.
6. ADO performs a lookup on the Human Resources page.
7. A data access component written in Microsoft® Visual Basic® 5.0 performs a complex update.
8. The page count is updated using a Page Count component.

In each step, issues that could prevent you from proceeding to the next step are outlined.

Step 1: The User Logs On

The user is prompted to enter some sort of user-authentication information on a page called Default.asp. The information is collected in an HTML form and posted to the server using an SSL session. Security problems at this stage are unlikely.

Issues to consider (see Figure 9.23):

- With SSL, use the HTTPS protocol rather than HTTP. Failure to do so will yield a 403.4 error (SSL required).
- Make sure that the server certificate is not out of date, or the browser could reject it.
- If the server certificate is from a CA unknown to the browser, the browser might fail to connect to the server.
- If the name in the server certificate is not the same as the name of the server you wish to communicate with, the browser might fail to connect to the server.

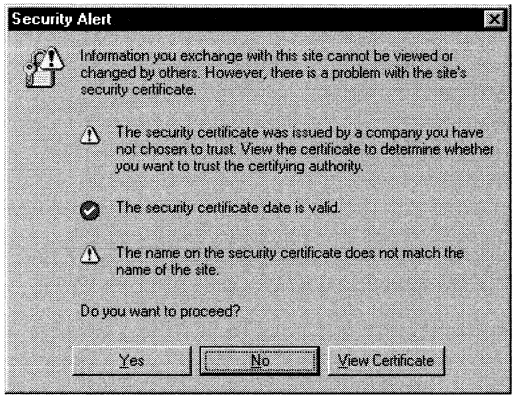


Figure 9.23 Internet Explorer 5 Message Warning That a Server Certificate Has an Incorrect Name and That It Is Issued by an Unknown CA

- Problems can occur if the administrator has set up .asp files so that they won't process HTTP POST methods. Although this is unlikely, with IIS 5.0 you can restrict HTTP method types if necessary, as shown in Figure 9.24.

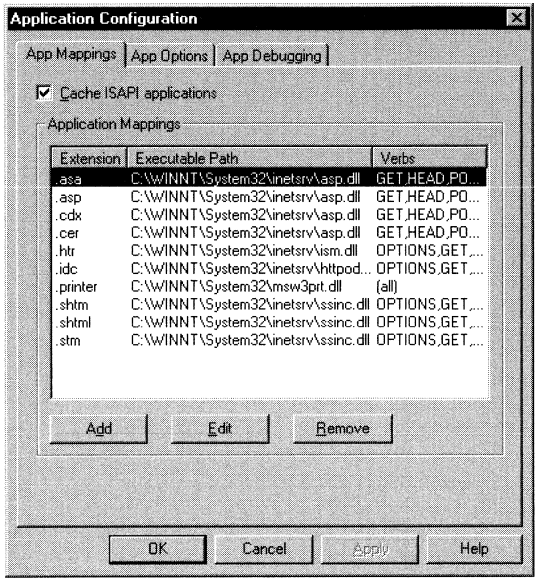


Figure 9.24 Setting HTTP Method Exclusion

> **Note** This functionality has changed from IIS 4.0, in which you select the HTTP verbs you do not want to allow. In IIS 5.0, you select the HTTP verbs you *want* to allow.

Step 2: An ISAPI Authentication Filter Is Called

Implemented as DLLs, ISAPI filters are quite common for custom logging and authentication.

You can use an ISAPI filter to receive notification of various events during the processing of HTTP requests. Filters are loaded when you start IIS 5.0, and are kept in memory until you shut down IIS 5.0. Once a filter is loaded, it indicates the events that IIS 5.0 should send notification about. Subsequently, IIS 5.0 will notify the filter each time one of the registered events occurs.

One type of notification is the Authentication notification. The ISAPI filter is notified when the client makes an initial request, and can then make decisions about whether to allow or deny access to the server. In the case of Exploration Air, this filter takes the user name and password entered in step 1 and uses this to perform a database query. If the name and password combination is correct, the user is allowed access to the Benefits Web application.

Problems can occur if:

- The ISAPI filter rejects the user for some reason, probably based on some data in the Human Resources database.
- The user types in a name or password incorrectly, or the database is out of synch.
- The UNIX server has crashed or the SQL*Net connection is failing. Check the Oracle database logs for any further clues, and use Oracle Trace for low-level information.

Step 3: IIS 5.0 Attempts to Authenticate the User

There are a number of steps that take place here, including IP and domain name restrictions.

Issue to consider:

- At this point, the user will be logged on as either an anonymous user or as a normal Windows account. You can verify this by looking in the Windows Audit Log.

Step 4: IIS 5.0 Loads the Logon Page Logon.asp

Logon.asp is loaded by IIS 5.0 in a separate memory space. This is done for reliability reasons. Eventually, this option will be turned off, once the Web administrators have deemed the code trustworthy and safe. You may wish to run newly written code that has not been thoroughly tested in a separate process. If this code should crash, it will only crash its own process and not the IIS 5.0 process.

Also, Logon.asp includes the following at the top of the page:

```
<%@ Transaction = Requires_New %>
```

This tells IIS 5.0 that it must start a new transaction for any component or data access that supports transactions. In the case of Logon.asp, this includes the Microsoft® Visual Basic® 6.0 data access component and the ADO code performing the Oracle query. Component Services is used to control the transaction.

Issues to consider:

- Any Web application that is marked to run in a separate memory space (out-of-process) runs in the security context of a special user account called IWAM_*computername*. However, the process will impersonate the calling user (for example, IUSR_*computername*) before accessing any resource. You can verify this in the Windows Audit Log.
- While not a security limitation per se, keep in mind the transaction semantics that are now in place. If a transactional component fails or explicitly rolls back its part of the transaction, other transactional components will lose their changes.

Step 5: Logon.asp Attempts to Read Data from Configuration Store

Using ADSI, the page tries to read data from the configuration store. The code looks like this:

```
<%
  Dim oWebApp
  Dim strAppName
  Set oWebApp = GetObject("IIS://LocalHost/W3SVC/1/Benefits")
  StrAppName = oWebApp.AppFriendlyName
  Set oWebApp = Nothing
%>
```

This code has the friendly name "Benefits application." If an administrator decides to change a comment about the application by using the IIS 5.0 administration tools, it will be reflected automatically in the Logon.asp page. Once again, you can verify this in the Windows Audit Log.

This can fail because:

- Accessing the Configuration Store requires Administration privileges on the Web site in question.

Step 6: ADO Performs a Lookup on the Human Resources Page

This is a straightforward database lookup, similar to that performed by the ISAPI filter in step 2.

Issues to consider:

- Problems can occur if the user does not have access to the database being queried.
- If the database connection is being performed with ODBC, use the ODBC Trace tool to check for any access problems. Check the Oracle database logs for further clues. Use Oracle Trace for low-level information.

Step 7: Data Access Component Written in Visual Basic 6.0 Performs a Complex Update

This section assumes knowledge of a relational database management system (RDBMS) such as SQL Server.

The Visual Basic 6.0 component is a Microsoft® ActiveX® DLL that is transaction-aware. The component contains data-access code that was written using ADO, in order to access a SQL Server 7.0 stored procedure.

The stored procedure performs a SELECT from two tables and then an INSERT into another. The last table also has an INSERT trigger.

Issues to consider:

- If you are attempting to access SQL Server using Windows-only security (this was called Integrated security in Microsoft® SQL Server 6.5) from an ASP application, you can have security problems. This is because the account that was used to log you on to IIS 5.0 might not be valid on SQL Server.
- The user account might not have access to the appropriate SQL Server objects (tables, stored procedures, and so on). Check the **Object Properties** dialog box in SQL Server 7.0 Enterprise Manager, shown in Figure 9.25.

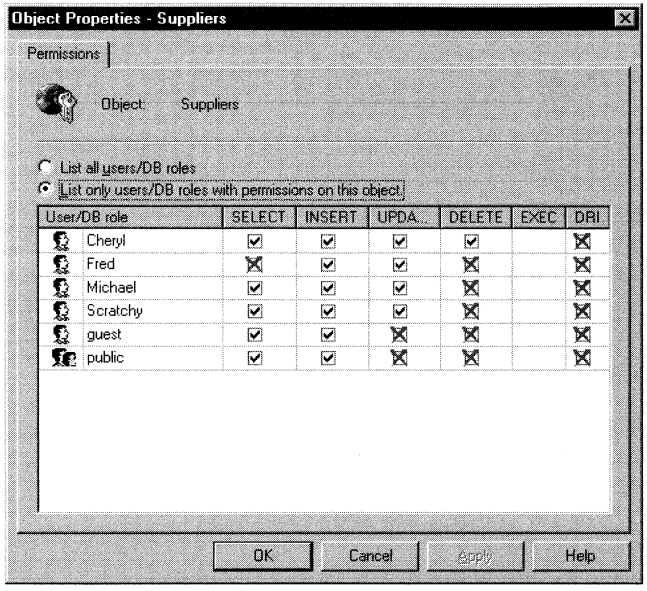


Figure 9.25 Object Permissions in SQL Server 7.0

- Enable Auditing in the SQL Enterprise Manager for successful and failed SQL Server logon.
- If the database connection is being performed with ODBC, then use the ODBC Trace tool to check for any access problems.
- Check the SQL Server Error Log.
- Use the SQL Trace tool to check low-level traffic.
- The INSERT trigger could be failing. For example, it could be performing some business-rule check that is failing, so the insert is rolled back. Check the trigger by right-clicking on the table in question and selecting **Manage Triggers**.

Step 8: Page Count is Updated Using a Page Count Component

Web pages are often counted. Normally when a user accesses a page, some server code is executed to open a file, increment a count in that file, and then resave it.

Issues to consider:

- This can be a cause of failure because the user account performing the request might not have access to the file on the server where the page-hit information is stored. Remedy this by allowing Everyone (both Read and Write) access. You can verify this by auditing for file/object access.

Defending Against Malicious Attacks

The need to safeguard corporate data is great, and while most Web users are benign, many are not. This section addresses some of the low-level security settings required in a Windows 2000 Server and IIS 5.0 environment.

To secure any computer system such as Windows requires a good deal of underlying system knowledge, as well as knowledge of the latest security fixes.

Historically, securing a Windows server has required setting security options in many areas, such as:

- The file system
- The registry
- Networking protocols
- Application settings
- User and group accounts
- Account policies
- System services

Setting all these policies by hand is both tedious and error-prone. Windows 2000 Server includes a tool called the Security Configuration and Analysis in order to alleviate this problem. Its primary goal is to provide a single point of administration for Windows system security.

Security Configuration and Analysis allows an administrator to:

- Define one or more security policies based on the role of the computer.
- Configure a server to match a security policy.
- Audit against an existing policy and report differences.

Using the Security Templates

The Resource Kit companion CD includes two IIS 5.0–specific security templates to be used with Security Configuration and Analysis in Windows 2000:

- Secure Internet Web Server
- Secure Intranet Web Server

Once the policy is updated to meet your requirements, you can import the appropriate template into Security Configuration and Analysis and audit against it or apply the template. It is preferable to audit first, in order to see how misconfigured your servers are.

It is important to note that these templates are starting guidelines. Each will require a little updating in order to reflect your corporate security policy.

Table 9.6 lists some areas to which you should pay close attention:

Table 9.6 Updating Policies to Meet Corporate Requirements

Policy	Comment
Allow storage of passwords under reversible encryption.	Enable only if you are using digest authentication.
User Rights Assignment.	Update the following to include the IUSR_ and IWAM_ accounts: • Log on locally. • Access this computer from the network.
Change Administrator account name to...	Consider changing the administrator account to some other name.
Change Guest account name to...	Consider changing the Guest account to some other name.
Enable digital signing of secure channel network traffic.	Enable, if you are in an intranet.
Enable encryption of secure channel network traffic.	Enable, if you are in an intranet.
Require secure channel traffic to be signed or encrypted.	Enable, if you are in an intranet.
Send downlevel LanMan compatible password.	If you are using Windows throughout, set this to **Not Compatible**.

You should also keep abreast of security issues as they arise by regularly checking http://www.microsoft.com/security/default.asp.

Auditing Access with IIS 5.0 Logs

You can use IIS 5.0 logs in order to track access to your server. Logging is very flexible and can be used in conjunction with a log file analysis tool (such as WebTrends from http://www.webtrends.com) to detect suspicious activity such as:

- Multiple failed commands, especially to the /Scripts directory, or to another directory configured for executable files.
- Attempts to upload files to the /Script directory, or to another directory configured for executable files.
- Attempts to access .bat, .exe or .cmd files and subvert their purpose.
- Attempts to send .bat or .cmd commands to the /Scripts directory, or to another directory configured for executable files.
- Excessive requests from a single IP address attempting to overload or cause a denial-of-service attack.

For more information about logging, see "Monitoring and Tuning Your Server" in this book.

Useful IIS Admin Objects/ADSI Security Settings

Configuring Web services can be time consuming, due to the number of parameters that often need to be set. The IIS Admin Objects (IISAO) can help reduce this work by allowing you to write scripts that automate the process.

The IIS Admin Objects are an ADSI implementation for accessing and setting IIS 5.0 settings. They are COM Automation-based, and can be used with any language that supports Automation, such as Microsoft® Visual Basic® Scripting Edition (VBScript) or Microsoft® JScript®, Visual Basic, Java, or C++.

Table 9.7 lists some useful IISAO security-related settings:

Table 9.7 Security Settiings with IISAO

IISAO Property/Object	Comment
AccessFlags	Sets access permissions such as Read, Write, and Script.
AccessSSLFlags	Sets SSL properties such as SSL required, requires 128-bit SSL, and requires client authentication certificate.
AnonymousPasswordSync	Supports Allow IIS to control password option.
AnonymousUserName & AnonymousUserPass	Allows anonymous user name and password.
AuthFlags	Specifies what sort of authentication scheme will be used.
IIsCertMapper	Manages mapping of client certificates to Windows user accounts.
LogonMethod	Specifies the logon method for Basic and Anonymous authentication.
PasswordCacheTTL	Specifies the amount of time in seconds that an expired password will be held in the memory cache.
UNCAuthenticationPassThrough	Enables user authentication pass-through for UNC virtual root access. This applies to authentication schemes that support delegation.

For more information about these settings, see the IIS 5.0 online product documentation.

IIS 5.0 Security Checklist

This section outlines some of the steps you should take to secure a Windows 2000 Server that is running IIS 5.0 on the Internet. Note that this document does not take into consideration firewalls or proxy servers. It also assumes the company has a security policy in place.

You will notice that this list is quite small, because most of the work is performed by Security Configuration and Analysis.

Step 1: General Information

Server Name		Manufacturer	
Asset #		Location	
Setup Date		Set up by	

Step 2: Background Work

	Read your corporate security policy.		Configure hardware to meet security policy.
	Read the *IIS 5.0 Resource Guide* "Security" chapter.		Check http://www.microsoft.com/security.

Step 3: Windows 2000 Server Settings

	Apply latest Service Pack and hot-fixes from ftp.microsoft.com.		Format hard disk(s) to NTFS.
	Review appropriate Secure Configuration Template settings.		Apply appropriate Secure Configuration Template settings.
	Turn off NTFS 8.3 name generation.		Set appropriate NTFS DACLs.
	Set system start time to zero seconds.		Set domain controller type to: _____.
	Remove OS/2 subsystem.		Remove POSIX subsystem.
	Remove all net shares.		Disable Guest account.
	Check user accounts, group membership, and privileges.		Set a *very* strong password for Admin account (at least nine characters).
	Unbind NetBIOS from TCP/IP.		Disable IP routing.

Step 4: IIS 5.0 Settings

	Install minimal Internet services required.		Set appropriate authentication methods.
	Set appropriate virtual directory permissions and partition Web application space.		Place executable content in Execute (X)-only location.
	Set IP address/DNS address restrictions.		Validate executable content for trustworthiness.
	Set up SSL if appropriate.		Enable Logging.
	Map Client Auth Certificates to Windows accounts if appropriate.		Set Indexing Service to *only* index documentation.
	Lock down Microsoft Certificate Services ASP enrollment pages with DACLs.		Disable or remove all sample applications.

Step 5: Install Scanner/Intrusion Software

Regularly run a security scanner on your Web server, by using software from one of the companies listed at http://backoffice.microsoft.com/securitypartners/.

Step 6: Update the Emergency Repair Disk (ERD)

You should regularly update the ERD.

To update the ERD

1. Run the Backup tool by clicking **Start**, **Run**, and then typing **ntbackup**.
2. Select the **Tools** menu.
3. Select **Create an Emergency Repair Disk**.

Additional Resources

The following books provide additional information about IIS 5.0 and about other features of Windows 2000 Server.

Books

Security Theory

Fundamentals of Computer Security Technology by E. Amoroso, 1994, Upper Saddle River: Prentice Hall.

Computer Security by D. Gollman, 1999, New York: Wiley.

Firewalls and Proxy Servers

PCWeek Intranet & Internet Firewall Strategies by E. Amoroso and R. Sharp, 1996, Indianapolis: ZD Press.

Firewalls & Internet Security: Repelling the Wily Hacker by W.R. Cheswick and S.M. Bellovin, 1994, Reading: Addison Wesley.

Internet Security—Risk Analysis, Strategies & Firewalls by O. Kyas, 1996, Washington, D.C.: Thomson.

Web Proxy Servers by A. Luotonen, 1997, Upper Saddle River: Prentice Hall.

System Intrusion

Maximum Security: A Hacker's Guide to Protecting your Internet Site and Network Second Edition, anonymous, 1998, Indianapolis: Sams.

Intrusion Detection by T. Escamilla, 1998, New York: Wily.

The Cuckoo's Egg by C. Stoll, 1995, Sydney: Pan MacMillan.

General Security

Database Security by S. Castano, M. Fugini, G. Martella, et al., 1994, Reading: Addison Wesley.

Security Protocols by B. Christianson, et al., 1997, Berlin: Springer.

Davis, P.T., ed. *Securing Client/Server Networks* by P.T. Davis, ed., 1996, Maidenhead: McGraw-Hill.

Protecting Yourself Online by Electronic Frontier Foundation, 1998, New York: HarperCollins.

Digital Certificates by J. Feghhi, et al., 1998, Reading: Addison Wesley.

Secure Electronic Commerce by W. Ford and M.S. Baum, 1997, Upper Saddle River: Prentice Hall.

Computer Communications Security by W. Ford, 1994, Upper Saddle River: Prentice Hall.

Practical Unix & Internet Security by S. Garfinkel and G. Spafford, 1996, Sebastopol: O'Reilly & Associates.

Web Security & Commerce, 1997, Sebastopol: O'Reilly & Associates.

Actually Useful Internet Security Techniques by L. Hughes, 1995, Indianapolis: New Riders.

Computer Security Reference Book by K.M. Jackson and J. Hruska, 1992, Perth: CRC.

Java Security, Hostile Applets, Holes & Antidotes by G. McGraw and E. Felten, 1996, New York: Wiley.

Computer Related Risks by P. Neumann, 1995, Reading: Addison Wesley.

Internet & TCP/IP Network Security by U.O. Pabrai and V.K. Gurbani, 1996, Maidenhead: McGraw Hill.

Web Security Sourcebook by A.D. Rubin, D. Geer, and M.J. Ranum, 1997, New York: Wiley.

Computer Security Basics by D. Russell and G.T. Gangemi, 1991, Sebastopol: O'Reilly & Associates.

Digital Money by D.C. Lynch and L. Lundquist, 1995, New York: Wiley.

Web Security—A Matter of Trust, by a collection of authors, Summer 1997, *World Wide Web Journal*, Vol. No. 3, Sebastopol: O'Reilly & Associates.

Information Security Policies Made Easy by C.C. Wood, 1998, Sausalito: Baseline Software.

Secure Computing Threats and Safeguards by R.C. Summers, 1997, Maidenhead: McGraw-Hill.

Encryption

Cracking DES, Electronic Frontier Foundation, 1998, Sebastopol: O'Reilly & Assoc.

Applied Cryptography, Second Edition by B. Schneier, 1996, New York: Wiley.

Cryptography and Network Security Principles and Practice by W. Stallings, 1998, Upper Saddle River: Prentice Hall.

Protect Your Privacy by R.B. Gelman and S. McCandish, *1995,* Upper Saddle River: Prentice Hall.

Windows NT Security

Internet Security with Windows NT by M.J. Edwards, 1998, Loveland: Duke Press.

Microsoft Windows NT 4.0 Security, Audit, and Control by J.G. Jumes, et al, 1998, Redmond: Microsoft Press.

Windows NT Security—Programming Easy-to-Use Security Options by N. Okuntseff, 1997, Gilroy: R&D Books.

NCSA Guide to Windows NT Security by C.B. Rutstein, 1997, Maidenhead: McGraw-Hill.

General Networking

Internetworking with TCP/IP, Volume I, Second Edition by E.C. Comer and D.L. Stevens, 1994, Upper Saddle Hill: Prentice Hall.

Internetworking with TCP/IP, Volume II, Second Edition by E.C. Comer and D.L. Stevens, 1994, Upper Saddle Hill: Prentice Hall.

Internetworking with TCP/IP, Volume III, Second Edition by E.C. Comer and D.L. Stevens, 1994, Upper Saddle Hill: Prentice Hall.

Internetworking with TCP/IP, Volume I, Second Edition, Windows Sockets Version by E.C. Comer and D.L. Stevens, 1997, Upper Saddle Hill: Prentice Hall.

Windows NT TCP/IP by K.S. Siyan, 1998, Indianapolis: New Riders.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

CHAPTER 10

Access to Legacy Applications and Data

By connecting your intranet to the Microsoft® Systems Network Architecture (SNA) network environment, you can gain access to the wealth of applications and data available on IBM platforms such as Multiple Virtual Storage (MVS) and OS/400 systems. Using Microsoft® SNA Server 4.0 SP2 as a data access server, you can readily develop scripts in Active Server Pages (ASP) that interoperate with legacy applications. This will allow you to process legacy data and distribute it across your intranet and over the Internet.

This chapter presents strategies for using Internet Information Services (IIS) 5.0 Web applications to access applications and data on IBM systems running in SNA environments—mainframes running MVS and AS/400 computers running OS/400. The strategies and scenarios in this chapter will help you use Microsoft development tools and production software packages in order to integrate legacy applications and data into IIS 5.0 Web applications.

In This Chapter

Identifying Strategies

Integrating IIS and Legacy Applications

Gaining Access to Legacy File Data

Replicating Legacy Databases

Migrating Transaction Processing

Additional Resources

Identifying Strategies

To leverage Web technology, an enterprise must make its business applications and data easily accessible to its employees, key business partners and customers. This goal is sometimes difficult to achieve because so much mission-critical data (up to 80 percent of vital business information for many large corporations and government agencies) is stored in host-based file systems and databases on IBM mainframes or AS/400 computers. This information—product specifications, customer profiles, position descriptions, and much more—is often unavailable to people who need it most, at the time they could make best use of it.

Delivering large amounts of data from legacy systems to many widely dispersed users has always been difficult because:

- Legacy hardware and system software is very expensive, often prohibiting expansion on an Internet-wide scale.
- IBM network protocols are not widely supported outside the legacy environment—on the Internet, for example.
- Application development costs are high, discouraging the development of modifications needed to make legacy data more widely available.

By using IIS 5.0 Web applications to access legacy applications and data on IBM systems, you can:

- Integrate host applications running in legacy environments into IIS ASP applications. Connect host transaction processors to Microsoft® Windows® 2000 Server, by using SNA Server 4.0 SP2 and Microsoft® COM Transaction Integrator (COMTI) for Customer Information Control System (CICS) and Information Management System (IMS) databases.
- Access legacy files at the record level using SNA Server 4.0 SP2 and Microsoft® OLE DB Provider for AS/400 and Virtual Sequential Access Method (VSAM) Data Provider. Send the data to a Web application running on Windows 2000 Server.
- Acquire host database structures using Microsoft® Host Data Replicator (HDR). *Replicate them* for Microsoft® SQL Server and IIS 5.0.
- Move the automated processes from the legacy environment to the open, more cost-effective Windows 2000 Server environment. To accomplish this, use IIS 5.0 and Microsoft® Component Services (formerly Microsoft® Transaction Server).

Connecting to SNA

Each of the legacy access strategies discussed in this chapter requires connections to IBM host computers through SNA. To understand how each strategy is implemented, you need a basic understanding of how the SNA environment is constructed, how to connect to SNA resources, and how to exploit them. For more detailed information about the SNA environment, see the SNA Server 4.0 SP2 product documentation.

The SNA Environment

To transmit data peer-to-peer over SNA, a session must be established between two Logical Units (LUs) in accordance with the LU 6.2 protocol, with one LU on the host system and the other on the client system. Because LU 6.2 is peer-to-peer, either the mainframe host or the client can initiate a session.

> **Note** Older SNA networks are hierarchical and do not support peer-to-peer communications. In addition, they do not use LU 6.2; instead, other LU types are used that conform to a hierarchical networking scheme.

As shown in Figure 10.1, any computer connected to an SNA network that uses this protocol, including computers running Windows 2000 Server or Microsoft® Windows NT® Server 4.0, can participate in the SNA environment and gain access to:

- Legacy host environments, including transaction processing monitors.
- VSAM files.
- AS/400 files (both flat and unstructured).
- Database data structures, such as IBM DB2 data tables.

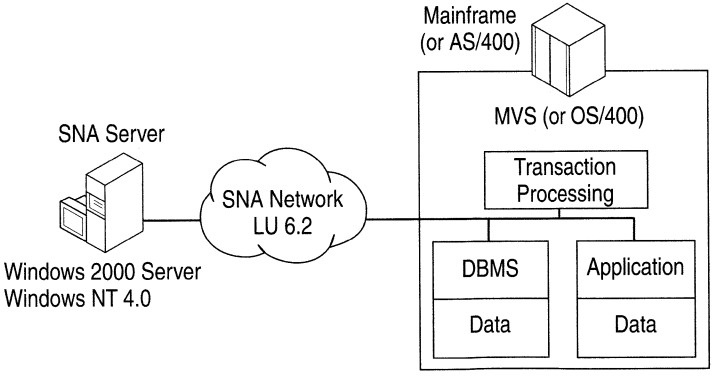


Figure 10.1 The SNA Environment

Connecting Through Microsoft SNA Server

SNA Server 4.0 SP2 provides access to applications and data in the SNA environment from Windows 2000 Server. It translates Windows 2000 Server Transmission Control Protocol/Internet Protocol (TCP/IP) communications to the LU 6.2 protocol, providing access to the wealth of resources in the legacy environment. See Figure 10.2.

SNA Server 4.0 SP2 runs on Windows NT Server 4.0 and Windows 2000 Server.

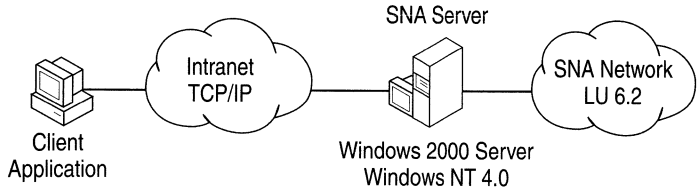


Figure 10.2 SNA Server Connecting Windows 2000 to SNA

Developing and Deploying Applications on Windows

With SNA Server translating between TCP/IP and LU 6.2, you can develop and deploy applications that access the legacy environment from the Windows 2000 Server side of the connection.

Software tools used to gain access to SNA applications and data reside on Windows 2000 Server platforms and take advantage of the unified administrative tools and lower-cost resources in the Windows 2000 Server environment.

Application development and modification is accomplished in the Windows 2000 Server environment as well. This means that you can avoid the high overhead associated with development and modification of legacy host-side resources.

Integrating IIS and Legacy Applications

For years, IBM has encouraged its customers to gain better control of their software development and maintenance by coding their business logic into programs separate from their terminal access logic. Many Information Services (IS) organizations have responded by coding their business rules into transaction processing programs that execute on CICS or IMS. Gaining access to these programs on the host side from the Windows 2000 Server environment can open up the business rules for an entire application, such as inventory control or budgeting, thus creating new opportunities for distributed applications.

Using Web applications to access business logic offers significant advantages over traditional methods such as using a "screen scraper" to gather data from terminal emulation programs because:

- All the data and processes that the business logic allows are accessible, rather than only the limited data and processes accessible in individual terminal access programs.
- There is no requirement for a terminal emulator on the Windows 2000 Server platform because the processing involves no terminal access software.
- Using native protocols, such as Advanced Program-to-Program Communications (APPC) and LU 6.2, requires fewer data translations, thus reducing the likelihood of errors.
- The integration of legacy processes with IIS 5.0–based applications using tools such as COMTI is easier and less costly to accomplish than extensive enhancement of applications in the legacy environment.

COM Transaction Integrator

COMTI for CICS and IMS is a technology that integrates legacy transaction processing programs (that operate on mainframes) with Web application and transaction processes (that run in the Windows 2000 Server environment). See Figure 10.3.

COMTI reduces the effort required to develop applications, by integrating COmmon Business Oriented Language (COBOL) programs on mainframes with Automation clients running Windows 2000 Server, Microsoft® Windows® 2000 Professional, Microsoft® Windows® 95, Microsoft® Windows® 98, or any other computer that supports Automation. Specifically:

- COMTI can automatically create a recordset of the data returned from a mainframe transaction processing program. The recordset data, formatted in a tabular array, can then be accessed by scripts in ASP pages.
- COMTI coordinates legacy transaction processing programs on the mainframe, which are managed by Component Services. This extends the transaction environment to include transactions managed by CICS or IMS on an IBM mainframe computer.
- COMTI development tools map COBOL data declarations to Automation data types.

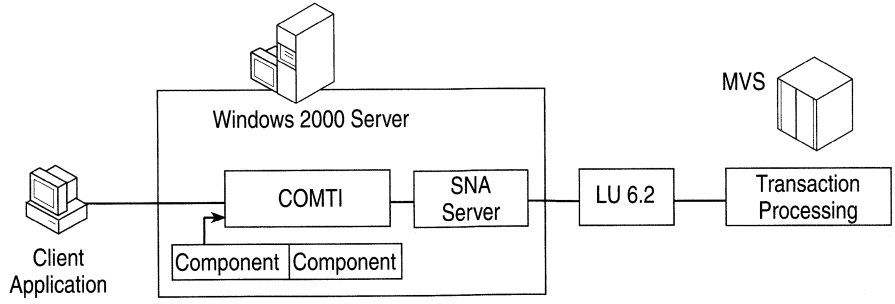


Figure 10.3 COMTI as a Proxy for the Mainframe

Functional Overview of the COM Transaction Integrator

The following list summarizes how COMTI gains access to CICS applications. It also shows how COMTI integrates data returned from CICS transaction processing programs with IIS 5.0, by using COM components and Component Services. Specifically, COMTI can:

- **Gain Access to CICS Transaction Processing Programs** COMTI directly supports any transaction processing program that executes in CICS or IMS. Because COMTI can access CICS programs, developers can issue application calls to the legacy environment, by using CICS to gain access to any program under its control. This includes DB2 databases, VSAM files, or IMS databases.
- **Redirect Method Calls** COMTI is a generic proxy for the mainframe. It intercepts method calls from the client application and redirects them to transaction processing programs running on the mainframe. For example, when an Internet browser sends data that ASP interprets as requiring COMTI, IIS forwards the data to COMTI.
- **Reformat Method Calls** When COMTI intercepts a method call, it converts and formats the method's parameters from Automation data types into IBM System 390 mainframe data types.
- **Handle Return Values** COMTI handles the return of all output parameters and values from the mainframe, converting and reformatting them for IIS as needed.

COMTI processing runs on computers using Windows 2000 Server, not on the SNA host. It does not require any new executable code to be installed on the mainframe, or on the desktop computer that is running the Internet browser. COMTI communicates through SNA Server 4.0 SP2. It uses standard protocols (LU 6.2 or TCP/IP, each of which is supported on SNA Server 4.0 SP2) in order to communicate between the computer running Windows 2000 Server and the mainframe transaction processing program.

COMTI Development Scenarios

The following two scenarios illustrate how the COMTI environment can be used to develop applications that integrate transaction processing programs with ASP pages.

Scenario One: Integrating Legacy Transaction Processing Data Using COM Transaction Integrator

This scenario illustrates how you can connect a Windows 2000 Server–based Web site to an existing COBOL program with transaction processing. Suppose you want to dynamically add content from a legacy database running on CICS on an IBM mainframe computer to a Web application running on IIS 5.0. You can begin by using ASP pages in order to interpret user requests and format the data returned by the mainframe application. Next, you can use COMTI to develop a component that will process the method calls from the IIS environment and the mainframe environment.

This scenario involves six main steps:

- Step 1 (setup time): Configuring COMTI
- Step 2 (design time): Defining required methods and parameters
- Step 3 (design time): Writing the application
- Step 4 (design time): Testing the application
- Step 5 (deployment): Deploying the application components
- Step 6 (post-deployment): Maintaining the application

Step 1: Configuring COMTI

To develop a COMTI component, you must have the following installed:

- Windows 2000 Server or Windows 2000 Professional
- IIS 5.0 with Component Services 2.0
- Microsoft® Windows Client for SNA Server
- Microsoft® Data Access Components (MDAC) 2.0

Additionally, the following COMTI components must be installed:

- The administration component, which collects information about the user's SNA environment.
- The run-time component, which intercepts the method calls to the mainframe and uses the COMTI-created component library to perform the actual conversion and formatting of the method parameters. In addition, the run-time component interacts with SNA Server and builds packets, which are sent to the mainframe using LU 6.2 or TCP/IP.
- Development tool components, featuring the Component Builder, a GUI used to create component libraries from mainframe COBOL programs.

The Component Builder can be installed as an add-in to Microsoft® Visual Basic® 5.0, and does not need to be installed on the same system as the other components. Developers who are not using Visual Basic 5.0 can use the Component Builder as a stand-alone tool.

Step 2: Defining Required Methods and Parameters

1. Acquire the COBOL source code from the mainframe by using a file transfer mechanism, such as the FTP-AFTP gateway service that is included with SNA Server 4.0 SP2.
2. Use the COBOL Import Wizard to:
   - Select the COBOL source code.
   - Specify the methods and mainframe transaction processing names.
   - Select input, output, and return value parameters.
3. When necessary, change the mappings between the COBOL and Automation data types.
4. Use the Component Builder to make a COMTI component type library (.tlb). This is a standard library that can be used by client software and Component Services.

If you have changed the data type mapping in the COBOL code, there are two more actions required in this step:

- Use the Component Builder to generate new COBOL declarations.
- Update the mainframe program with the new COBOL data declarations. This is the only instance requiring modifications to the mainframe environment.

Step 3: Writing the Application

1. Write the client in a language that supports referencing of Automation objects, such as Visual Basic, Microsoft® Visual C++®, or Microsoft® Visual J++®.
2. Add the appropriate COMTI component library to the references list in the project and add the references of the component in the program.
3. Invoke methods as appropriate throughout the application.

If the existing mainframe transaction processing program needs to be modified, do one of the following:

- Perform the modification on the mainframe.
- Use a Windows-based COBOL development environment, such as Microfocus COBOL. Then move the code to the mainframe.

Step 4: Testing the Application

If the mainframe transaction processing program is unchanged, it does not require testing. If the transaction processing program has been modified, then the COBOL program should be tested independently to ensure that it runs correctly in its own environment.

To test the new application

1. Ensure that the COMTI component library is registered in Component Services.
2. Test the mainframe transaction processing independently, if it has been modified in any way.
3. Test the newly developed COMTI component independently to ensure that it is working correctly.
4. Test the application, running the mainframe transaction processing program.

Step 5: Deploying Application Components

Each of the following must be installed on the production computer before you deploy the client side of the application:

- Windows 2000 Server or Windows 2000 Professional
- Windows Client for SNA Server
- Component Services
- COMTI administration and run-time components
- COMTI component libraries registered in Component Services
- Client applications that access COMTI components

Step 6: Maintaining the Application

As changes are made to the mainframe transaction processing program, do one or more of the following, as appropriate:

- Acquire the COBOL source code from the mainframe.
- Use the COBOL Import Wizard in order to respecify the method names and host transaction processing names, and reselect the input, output, and return parameter values.

What If the Required Mainframe Transaction Processing Does Not Exist?

In this case, you must modify steps 2 and 3 of this scenario by developing a transaction processing program that runs on CICS (on the mainframe host).

In step 2, use the COMTI Component Builder to:

- Enter the methods and parameters for the application.
- Add information about the name and location of the new transaction processing program.
- Change the default mappings produced by the Component Builder, if necessary.
- Create the COMTI component library.

In step 3:

- Write the mainframe transaction processing program, either on the mainframe or in the Windows environment using a product such as Microfocus COBOL; then move the program to the mainframe for testing.

Scenario Two: Extending Transactions with COMTI

When deployed in the Windows 2000 Server–based environment, COMTI can extend transactions to include mainframe transaction processing programs running on CICS and IMS.

A developer can use the following steps in order to connect a Windows 2000 Server–based Web site to an existing COBOL transaction processing program. Doing so will make legacy data available to scripts in ASP pages. In this scenario, additional tasks are needed in order to extend transactions that are under the control of CICS.

Like the previous scenario, this scenario also involves six main steps:

Step 1: Configuring COMTI

To develop the COMTI object, you must have the following installed:

- Windows 2000 Server or Windows 2000 Professional
- IIS 5.0
- Windows 2000 Client for SNA Server
- Component Services

Additionally, the following COMTI components must be installed (for descriptions of each of the components, see Scenario One):

- The administration component
- The run-time component
- The Component Builder

Step 2: Defining Required Methods and Parameters

To make the mainframe transaction processing program data available to IIS, perform the following tasks:

1. Acquire the COBOL source code from the mainframe by using a file transfer mechanism such as the FTP-AFTP gateway that is delivered by SNA Server 4.0 SP2.
2. Use the COBOL Import Wizard to:
   - Select the COBOL source code.
   - Specify the methods and mainframe transaction processing program names.
   - Select input, output, and return value parameters.
3. When necessary, change the mappings between the COBOL and Automation data types.
4. Use the Component Builder to make a COMTI component library (.tlb). This is a standard library that can be used by client software and Component Services.

If you have changed the data type mapping in the COBOL code, there are two more actions required in this step:

- Use the Component Builder to generate new COBOL declarations.
- Update the mainframe program with the new COBOL data declarations. This is the only instance requiring modifications to the mainframe environment.

Step 3: Writing the Application

1. Write the client in a language that supports referencing of Automation objects, such as Visual Basic, Visual C++, or Visual J++.
2. Add the appropriate COMTI component library to the references list in the project and add the references of the component in the program.
3. Invoke methods as appropriate throughout the application.
4. Define any transaction-related attributes in the COMTI component. These attributes will handle transactions in a manner transparent to the client application (for example, an IIS 5.0 application using ASP). The COMTI component will call both the Component Services/Distributed Transaction Coordinator (DTC) and the transaction processing program running on CICS.

Step 4: Testing the Application

If the mainframe transaction processing program is unchanged, it does not require testing. If the transaction processing program has been modified, then the COBOL program should be tested independently to ensure that it runs correctly in its own environment.

To test the new application

1. Test the mainframe transaction processing program independently, if it has been modified in any way.
2. Test the newly developed COMTI component independently to ensure that it is working correctly.
3. Test the application completely, by driving the COMTI object with the client application and running the mainframe transaction processing program.
4. Carry out a transaction test. To do this, test the COMTI object with the transactions made available. Check operation between COMTI and Component Services, in conjunction with COMTI and the transaction processing program running on CICS.

Step 5: Deploying Application Components

Each of the following must be installed on the production computer before you deploy the client side of the application:

- Windows 2000 Server or Windows 2000 Professional
- Windows Client for SNA Server
- Component Services
- COMTI administration and run-time components
- COMTI component libraries registered in Component Services
- Client applications that access COMTI components

Step 6: Maintaining the Application

As changes are made to the mainframe transaction processing program, do one or more of the following, as appropriate:

- Acquire the COBOL source from the mainframe.
- Use the COBOL Import Wizard in order to respecify the method names and host transaction processing program names, and reselect the input, output, and return parameter values.

Using COMTI with IMS

Current versions of COMTI do not support transactional semantics (also known as a two-phase commit) on the IMS subsystem. However, you can access an IMS/DB database transaction through a CICS subsystem, front-end transaction processing program. That is, if the mainframe environment supports CICS transaction processing against IMS/DB, you can extend Component Services transactional semantics to the IMS/DB database. In this case, COMTI provides the same services as any other transaction processing program running on CICS. If you do not require transactional semantics, and just want to gain access to your data, you can access IMS directly.

Gaining Access to Legacy Data

OLE DB data providers make it easy to access diverse legacy data sources from IIS 5.0. Data providers targeting legacy host environments include:

- OLE DB provider for DB2
- OLE DB provider for AS/400 and VSAM

This section describes how you can incorporate legacy file systems into your Web applications, by using a data provider to access AS/400 and VSAM files at the record level and move the data to the IIS 5.0 environment. For information about OLE DB provider for DB2, see the SNA Server product documentation.

Legacy File Data and IIS 5.0

In order to develop Web applications that deliver data stored in VSAM and AS/400 files, you need to gain access to VSAM and AS/400 files from the Windows 2000 Server environment. You can do this by making the data available to data consumer applications that are running ASP:

- Access legacy file systems running on MVS and OS/400 to retrieve the business data stored in them.
- Integrate legacy data with applications and data in the IIS 5.0 environment using the OLE DB provider for AS/400 and VSAM.

Access to VSAM and AS/400 files with OLE DB and ActiveX Data Objects

The OLE DB provider for AS/400 and VSAM is the first application to make record-level mainframe VSAM and AS/400 file systems available to ASP applications. The data provider allows consumer ASP applications to gain access to the mission-critical data available in those file systems. The data provider ships with SNA Server 4.0 SP2, Windows Client for SNA Server, and the SNA Server SDK.

For more information about developing ASP applications, see "Developing Web Applications" in this book.

Functional Overview of the Data Provider

The OLE DB provider for AS/400 and VSAM comprises two core components. The first is an OLE DB–compatible data provider that insulates the complexities of LU 6.2 programming from the OLE DB or Microsoft® ActiveX® Data Objects (ADO) programmer. The second is an SNA Distributed Data Management (DDM) transaction management program that runs as a service on Windows 2000 Server, or as an application on Windows 95 or Windows 98.

The following list summarizes the uses of the data provider:

- From Windows 2000 Server, you can gain access to VSAM and AS/400 file systems through the IBM DDM protocol server components (OLE DB/DDM Driver) installed on many IBM host systems. There is no need to install Microsoft software on the host system.
- You can use customizable applications in order to read and write to VSAM and AS/400 files that are in place on IBM host computers. There is no need to migrate the files to the Windows 2000 Server environment.
- You can gain access to fixed and variable logical-record-length classes, as well as file and record locking, while preserving file and record attributes.
- You can gain access to most AS/400 file types (both physical and logical) and most popular mainframe dataset types: sequential (SAM); VSAM key-sequenced (KSDS), VSAM entry-sequenced (ESDS), VSAM relative-record (RRDS), and partitioned (PDS/PDSE).

Leverage Existing Systems with Data Provider

The data provider makes it possible to integrate unstructured legacy file data with data in the Windows 2000 Server environment. The DDM protocol provides program-to-program communications through SNA Server 4.0 SP2 and native host protocols (such as LU 6.2 and TCP/IP). No custom development is required on the host system.

IBM DDM servers are available on host systems supporting record-level access to files. For example, Distribute File Manager, a component of the IBM Data Facility Storage Management Subsystem (DFSMS) version 1R2 or later, is a target DDM server installed on many mainframes running on MVS or OS/390. On AS/400 computers, OS/400 (version 2R2 or later) runs as a DDM server. The data provider communicates with Data File Manager and OS/400 through LU 6.2 and TCP/IP.

The data provider makes it easy for developers to gain access to high-level component interfaces such as OLE DB or ADO. It supports development in Visual Basic, Visual C++, VBScript, and Microsoft® JScript®. Web developers don't need to know how to run SNA or LU 6.2.

Scenario: Using Data Provider to Gain Access to Host Files

With the data provider, you can gain access to file data on an IBM host from a Windows 2000 Server–based Web application. Suppose that you want to add content from a legacy file that is stored on an IBM mainframe or AS/400 computer to an ASP application running on IIS 5.0. ASP can interpret user requests and format the return of data to the user through Web pages. The data provider can process calls from the IIS 5.0 environment and pass data returned from the mainframe environment to IIS.

This scenario requires six main steps:

- Step 1 (setup and configuration): Configuring the data provider
- Step 2 (design time): Defining the application requirements
- Step 3 (design time): Writing the application
- Step 4 (design time): Testing the application
- Step 5 (deployment): Deploying the application components
- Step 6 (post-deployment): Maintaining the application

Step 1: Configuring the Data Provider

To develop an application using the data provider, you must utilize the following system requirements:

- Windows 2000 Server or Windows 2000 Professional.
- IIS 5.0 or later. This includes Microsoft® Data Access Components 2.1 (MDAC 2.1) supported by the data provider.
- Microsoft® Windows® Client for Microsoft® SNA Server or SNA Server 4.0. Configure it so that it connects to SNA Server 4.0 SP2.

Additionally, the following packages must be installed and configured:

- Microsoft OLE DB provider for AS/400 and VSAM.
- Microsoft OLE DB provider for AS/400 and VSAM snap-in for Microsoft® Management Console (MMC). Configure the DDM service for the target host and PC locale. Optionally, configure the data sources if you are not passing the information through an ADO consumer application. Configure the mainframe data column description to OLE DB data type mappings.

Step 2: Defining Application Requirements

1. Compile a list of target host files, keys, and alternate index paths. Define the subset of records to be read from the target Web application.
2. Specify the ADO objects, methods, properties, and collections supported by the data provider that will be used in the application.
3. Consider using **Recordset.Filter** to define recordsets based on logical search criteria and to search for records based on the application program and user input.
4. Use the ADO errors collection to produce errors in formats that the program can understand. This will prevent passing unnecessary error conditions to the Web browser.
5. Use either the automatic AS/400-to-OLE DB data transformation, or custom mapping, by using a data column description file from the DDM service host.
6. Decide whether to automatically map logon user IDs obtained from the Web browser.
7. Choose a deployment option and decide whether to run the DDM service on the computer running SNA Server, or on the computer running Windows Client for SNA Server.

Step 3: Writing the Application

1. Write scripts in order to gain access to ADO 2.0 from an ASP page. These should be written in a language that supports referencing of Automation objects, such as VBScript or JScript.
2. Cast data to match the OLE DB and host data types. Refer to the recordset schema in order to determine which host data types are supported. Ensure that the host data is valid before writing to the host files, especially if a host application concurrently gains access to host data files.
3. Check the syntax of supported OLE DB methods and properties. Pay special attention to the connection string (which is constructed using the data link dialog boxes) and the **Recordset.Open** parameters. These are unique to each OLE DB provider.
4. If appropriate, use the MS$NAME placeholder in order to pass the user ID and password to the SNA Server 4.0 SP2 host security feature. The security feature performs security mapping between Windows accounts and mainframe accounts.
5. Program some loops in order to ensure that target recordsets contain data before passing recordset methods. This will allow for delays caused by network conditions and remote target hosts.

Step 4: Testing the Application

Test the new application to make sure that:

- The ASP pages run correctly.
- There is clean communication between ADO and the DDM Service on the Windows 2000 Server side. Consider starting the DDM Service on Windows 2000 Server automatically, in order to ensure the timely availability of a PC-to-host connection with minimal session startup time.
- There are reliable, efficient operations between DDM Service and the host by way of SNA Server 4.0 SP2. Consider keeping the connection between SNA Server and the host connection active, in order to reduce session startup time.
- There is proper data display in the Web applications. Ensure the data integrity of host files, because there can be a loss of precision when you move data from the host to the PC and back again.

Step 5: Deploying Application Components

Each of the following must be installed on each production computer before you deploy the application:

- Windows 2000 Server or Windows 2000 Professional.
- Windows Client for SNA Server.
- IIS 5.0.
- OLE DB provider for AS/400 and VSAM.
- ASP pages requesting data from the legacy files.
- DDM Service running with Windows Client for SNA Server or SNA Server 4.0. To improve responsiveness, consider starting the DDM Service automatically when the system restarts.

Step 6: Maintaining the Application

If you modify the scripts in ASP pages, you need to retest the application by using the following guidelines:

- Test the application fully if you add new scripts or script fragments to existing ASP pages that request data from new data sources.
- If the target host data files are restructured, or new host tables are added, these changes need to be incorporated into the Web application by modifying ADO methods and creating new recordsets as needed.
- If the host connectivity changes, you should verify the Windows Client for SNA Server configuration or the data provider data sources.

Replicating Legacy Databases

Much of the data stored in legacy systems resides in relational databases. One option for gaining access to legacy data that IIS 5.0 will use is to replicate database tables from a legacy application to SQL Server 7.0 running on Windows 2000 Server.

Why Replication?

Legacy database replication is a set of conversion processes that copies, reformats, and migrates database tables for use in relational databases running on Windows 2000 Server. Using data replicated from a legacy database, developers and systems engineers can:

- Integrate the legacy data with data from the Windows 2000 Server side. If the data is replicated for storage in a Windows 2000 Server–side database, IIS 5.0 connects Internet or intranet clients to dynamically created Web pages. The data is then retrieved through OLE DB provider or Open Database Connectivity (ODBC).

- Readily subject the data to new business logic. For new processes involving the database, developing in Windows is less costly than developing in legacy systems.
- Manage the data efficiently. Windows 2000 Server with IIS 5.0 provides a common set of system management tools, database management tools, Web application servers, and transaction services.
- If the application requires it, rereplicate the data from SQL back to the legacy database.

Replicate Data Using Data Transformation Services

Data Transformation Services (DTS) is a standard feature of Microsoft® SQL Server 7.0. With DTS you can import data to SQL Server 7.0 from legacy databases using OLE DB.

An overview of DTS capabilities and features is provided here. For detailed information, see the SQL Server 7.0 product documentation.

If you are still using a pre-SP2 version of SNA Server 4.0, you can replicate DB2 data by using HDR. See "Replicating DB2 Tables by Using Host Data Replicator" later in this chapter.

Import, Export, and Transform Database Data

Either interactively or automatically, DTS simplifies the process of importing and transforming data from multiple, heterogeneous sources. It supports data lineage, making it easy to track the origin of data. In addition, DTS enables you to move and transform data to and from the following sources:

- OLE DB providers for SQL Server 7.0 and others
- ODBC data sources such as Microsoft® Access, Oracle, and DB2, which are using OLE DB provider for ODBC

DTS provides the functionality to import, export, and transform data between Microsoft SQL Server 7.0 and any OLE DB format. Using DTS, it is possible to build data warehouses and data marts in SQL Server 7.0 by importing and transforming data from multiple heterogeneous sources on a regularly scheduled basis (requiring no user intervention).

DTS imports and exports data between applications by reading and writing data in a common format. For example, DTS can import data from an ASCII text file or an Oracle database into SQL Server 7.0. Alternatively, data can be exported from SQL Server 7.0 to an ODBC data source, or a Microsoft® Excel spreadsheet.

> **Note** DTS only moves schema and data between heterogeneous data sources. Triggers, stored procedures, rules, defaults, constraints, and user-defined data types are not converted between heterogeneous data sources.

Transformations are applied to source data before it is stored in its new destination. For example, DTS allows new values from one or more source fields to be calculated. It also permits a single field that has been broken into multiple values to be stored in separate destination columns. Transformations make it easy to implement complex data validation, scrubbing, and enhancement during import and export.

DTS supports multistep packages, where multiple files can be processed separately, then brought together in a single, final step. Single records in a file can be broken up into multiple records in the destination, or multiple records in the source can be aggregated into single records in the destination.

Creating DTS Packages with Wizards

DTS Export Wizard

DTS Export Wizard guides you through the process of creating DTS packages in order to export data from a SQL Server 7.0 database to heterogeneous data sources. Help is included with this wizard.

DTS Import Wizard

DTS Import Wizard guides you through the process of creating DTS packages in order to import heterogeneous data to a SQL Server 7.0 database. Help is included with this wizard.

Replicating DB2 Tables by using Host Data Replicator

If you are still using a pre-SP2 version of SNA Server 4.0, you can replicate DB2 data by using HDR and IIS 5.0.

HDR is a database replication software product that copies predefined data from IBM DB2 database tables to SQL Server 7.0 database tables. It can do so on demand, at a single scheduled time, or according to a recurring schedule. HDR has the capability to reverse the process as well, replicating SQL Server 7.0 tables for use in a DB2 database.

HDR is composed of the data replicator service (a Windows 2000 operating system service) and Data Replicator Manager (a Windows 2000 operating system application for administration). The Data Replicator Manager has a user interface similar to those that appear in SQL Enterprise Manager and in the scheduling portions of SQL Executive.

Two-Way Replication

The HDR performs fully updated two-way replication. A complete "snapshot" of the source table is copied into the target database table, by using either Bulk Copy Program (BCP) when copying to SQL Server 7.0, or ODBC "inserts" when copying to DB2. All of the records of the target table are overwritten each time replication occurs. Optionally, you can append data to the end of the existing table, provided you do not change the table schema.

Flexible Processing and Filtering

HDR flexibly reproduces host data in whole or in part according to selection criteria determined by the user:

- Replication of selected columns ("vertical partitioning")
- Replication of selected rows ("horizontal partitioning")
- Replication of selected columns from selected rows (combined vertical and horizontal partitioning)

HDR can either replace entire tables or merge replicated data with existing tables that are located in the target database. Changes may be made in format or data values in several ways:

- Construction of destination columns calculated ("derived") from source data
- Use of SQL expressions in order to alter data in destination tables before or after replication
- Change in column names, column data types, or column order between source and destination

Scheduling

HDR offers the ability to perform single replication on demand (repeatable at will or through a programmatic interface such as SP_RUNTASK). It also allows single or repeated replication at predetermined times.

Statistics

Statistics gathered by HDR are available through the Data Replicator Manager or through the System Monitor in Windows 2000 Server. Statistics include the following:

- Throughput for each replication operation
- Number of bytes transferred for each replication operation
- Elapsed time for each replication operation

Security

The Data Replicator Manager prompts the administrator to supply a valid SQL Server 7.0 account and password each time it establishes a connection to a Data Replicator Service. If a correct account and password are not provided, the Data Replicator Manager closes the connection, thus preventing administration of the associated service and its subscriptions. (A subscription refers to a replication operation involving one source table and one destination table. A single Data Replicator Service can handle many subscriptions.)

For DB2, during subscription setup, an administrator must supply a valid DB2 account and password. HDR will also support the version 3.0 Single Sign On option of SNA Server 4.0. Database administrators using HDR are not required to keep track of multiple passwords.

SQL Server 7.0 destination table ownership can be defined during subscription setup. Access to replicated data is then controlled through normal SQL Server 7.0 security measures.

Performance

Source table names can be filtered so that they reduce network traffic and improve performance during setup. This allows subscriptions to be updated in environments that have large numbers of possible source tables. Also, connections to source and destination servers can be pooled in order to avoid the performance costs of reestablishing connections unnecessarily. Pool sizes can be adjusted as needed.

The Data Replicator Service caches subscription information. By doing this, it avoids the performance costs of obtaining the information from the data replicator control database at each scheduled replication time.

HDR does not coordinate online database transactions. Thus, it is not a suitable tool for carrying out routine database updates.

Supported Platforms

HDR is supported on the following platforms:

- SNA Server 4.0 or later
- Microsoft® Windows NT Server 3.51 or later (Intel and Alpha-based)
- Microsoft® SQL Server 6.5 or later
- IBM DB2 including DB2 (MVS), DB2/VM (SQL/DS), DB2/400, and the common family (DB2/2, DB2/6000, and DB2/2 Windows 2000 Server using APPC)

Migrating Transaction Processing

Component Services is a transaction management server that provides reliable, secure transaction management for Web applications. The following section provides information that can help you plan a migration of transaction-driven applications from any legacy environment to the Windows 2000 Server environment, where transactions requested by IIS 5.0 are managed by Component Services.

Why Use Transactions?

Two changes in the use of information technology make the increased deployment of transaction management systems compelling for many organizations:

- The growing demand to use the Internet and intranets for exchanging secure information, including financial exchanges through online commerce
- The increasing trend of running multiple reusable software components within one application, including components used to access databases

As stated, the explosive growth of the Internet and organizational intranets has presented new opportunities for doing business over data networks. The elemental expression of doing business—the exchange of money for goods or services—requires making updates to more than one database for each exchange.

In addition, software design is increasingly shifting toward a component model, in which applications are made up of many code segments operating independently of each other. An application often allows more than one component to concurrently update one or more databases. Concurrent updates require a transaction manager, in order to ensure transaction integrity while still optimizing performance.

For more information, see "Data Access and Transactions" in this book.

Migrating to Component Services

Since there is a growing need for transaction management on the Web, and since many transaction systems that exist on legacy networks process critical business data, more and more organizations need to manage transactions in both environments. A serious obstacle to achieving this is that legacy transaction systems do not extend across boundaries, such as the boundary between SNA legacy networks and TCP/IP–based intranets using Windows 2000 Server. In other words, a legacy transaction processing program running on CICS or IMS cannot track or verify a database update on the Windows 2000 Server network. Additionally, the costs of development, hosting, and scaling up are higher in the legacy environment than on the Windows 2000 Server network.

The best solution is a Windows 2000 Server–based transaction management system that coordinates Web transactions based on IIS 5.0 with legacy transaction processing programs. Any transaction can then involve updates of databases running on Windows 2000 Server, a mainframe transaction processing program, or both at the same time. Transaction processes can be selectively migrated to Windows 2000 Server–based database management software, such as SQL Server 7.0, and any transaction processes left on the mainframe can be managed from Windows 2000 Server as well.

Features and Capabilities

Component Services expands the capabilities of IIS 5.0 to include Web-based transaction management. Essentially, this is a component-based transaction management solution that provides a programming model, a run-time environment, and graphical server administration tools—everything required to design and develop a transaction application and migrate a legacy process to it. With Component Services, Web developers who use ASP pages can create full transaction management capabilities for deployment on the Web.

By deploying Component Services as your transaction management system, you can profit from the advantages of the Windows 2000 Server environment and migrate selected legacy transactional processes. At the same time, Component Services extends transaction management to include processes left running in the legacy environment.

In addition to providing full transaction monitoring and management, as well as a low cost of scaling up Windows 2000 Server–based systems and software, Component Services offers advantages over its mainframe-based counterparts in design time and at run time. Other advantages involve maintenance and administration.

Component Services Design Time

The Component Services programming model provides the framework to develop components that encapsulate business logic.

Component Services fits perfectly into a three-tier programming model; acting as middleware, it manages the components that make it possible for a Web application to handle many clients. It also provides developers with a great deal of flexibility:

- The model emphasizes a logical architecture for applications, rather than a physical one. Any service can invoke any component.
- Component Services connects requests for transactions (calls from scripts in ASP pages) to business logic and to database applications, so that you are not required to develop these processes.
- The applications are distributed, which means you can run the right components in the right places. This benefits users and optimizes network and computer resources.

Three-Tier Applications and Middleware

A three-tier application divides a networked application into three logical areas. Middleware, such as Component Services, connects the three tiers: presentation, business logic, and data processing.

Tier 1 handles presentation. In a Web application, data is requested by the browser and is sent there from the Web server for display.

Tier 2 processes business logic (the set of rules for processing business information). In an IIS 5.0–based Web application, Tier 2 processing is carried out using components of IIS 5.0.

Tier 3 processes the data (the associated databases and files where the data is stored). In a Web application, Tier 3 consists of a back-end database management system (DBMS) or a file access system with its associated data.

Three-tier systems are easier to modify and maintain than two-tier systems because the programming of presentation, business logic, and data processing are separated by design. This architecture permits redevelopment to proceed in one tier, without affecting the others.

Middleware, such as Component Services, makes efficient use of resources so that Web application programmers can concentrate on business logic. Component Services can connect the browser request (Tier 1) to the business logic (Tier 2). In Tier 3, it can connect business logic to the databases and manage all activities of the transaction.

Development Efficiency

Application programming interfaces (APIs) and resource dispensers make applications scalable and robust. Resource dispensers are services that manage nondurable shared-state architecture, on behalf of the application components within a process. This way, you don't have to undertake traditional programming tasks associated with state maintenance.

Component Services works with any application development tool capable of producing COM-based dynamic-link libraries (DLLs). For example, you can use Visual Basic, Visual C++, Visual J++, or any other Microsoft® ActiveX® tool to develop Component Services applications.

Component Services is designed to work with a wide variety of resource managers, including relational database systems, file systems, and document storage systems. Developers and independent software vendors can select from a broad range of resource managers, and can use two or more within a single application.

The Component Services programming model simplifies migration, by making transaction application development easier and faster than traditional programming models allow. For more information about developing Component Services applications, see "Data Access and Transactions" in this book.

Component Services Run Time

The Component Services run-time environment is a second-tier platform for running COM components. This environment provides a comprehensive set of system services including:

- Distributed transactions.
- Automatic management of processes and threads.
- Object instance and connection pool management, in order to improve the scalability and performance of applications.
- A distributed security service that controls object invocation and use.
- A graphical user interface (GUI) that supports system administration and component management.

This run-time infrastructure makes application development, deployment, and management much easier by making applications scalable and robust.

Overall performance is optimized by managing component instantiation and connection pooling. Component Services instantiates components just in time for transactions. It then purges state information from the instance when a transaction completes, and reuses the instance for the next transaction. For example, users can enter transaction requests from their browsers to an ASP page containing the code that is needed to call a COM component. As Component Services receives these messages, the transactions are managed using components already instantiated. This minimizes object instantiation and the number of connections required, both of which often inhibit the performance of systems that support transactions.

Component Services Administration Tools

Microsoft® Component Services Explorer is a graphical administration tool used to register, deploy, and manage components that execute in the Component Services run-time environment. With Component Services Explorer, you can script administration objects in order to automate component deployment.

Planning a Migration to Component Services

As you migrate processes and databases from a legacy environment (transaction processing programs running on CICS on a mainframe) to Component Services, the transaction processing programs will continue to run on the mainframe for a while. To migrate to Component Services:

- Use Component Services and COMTI in order to extend transaction management so that it includes all the parts of each transaction. This involves updates that take place on databases running on Windows 2000 Server, as well as updates that take place on the mainframe.
- Script the ASP pages so that IIS 5.0 calls the COM components that execute the transaction.

You can migrate parts of the legacy transaction infrastructure to SQL Server 7.0 and use Component Services to manage the parts of the transaction on the legacy host. IIS 5.0 can access all of the data by using scripts in ASP pages.

Mapping Transaction Tasks to Windows 2000 Server–Based Applications

Table 10.1 maps transaction-related functions to the applications used to support them in the Windows 2000 Server environment.

Table 10.1 Transaction-Related Functions and Windows 2000 Server–Based Applications

Transaction-Related Task	Windows 2000 Server–Based Application
Manage transactions	Component Services
Manage data resources	SQL Server 7.0
Call transactions from Web pages	ASP
Connect to a legacy network	SNA Server 4.0 SP2
Extend transactions to legacy transaction processing programs	COMTI

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server. It also provides resources for accessing legacy applications and data.

Web Links

http://www.microsoft.com/data
: The Microsoft Universal Data Access Web site provides product information, technical documents, and application development resources involving OLE DB, ADO/RDS, ODBC, and more.

http://www.microsoft.com/sna/default.asp
: The Microsoft® SNA Web site provides information about current and future releases of Microsoft SNA Server 4.0 SP2, COMTI for CICS and IMS, and the OLE DB provider for VSAM and AS/400. Features include access to evaluation copies of SNA Server 4.0 SP2, as well as related software that can be downloaded. The site also offers planning, deployment, development, and support resources, training options, links to SNA specialists, and popular newsgroups of the SNA community.

http://msdn.microsoft.com/default.asp
: MSDN Online offers up-to-date resources on all aspects of developing Web applications, including multitier ASP applications.

Books

Microsoft SNA Server 4.0 Resource Guide, 1997, Redmond: Microsoft Press.
: This second volume of the *Microsoft® BackOffice® Resource Kit* provides the technical information and resources needed to deploy, support, and integrate SNA Server 4.0.

SNA Server 4.0 Software Product Documentation

For information about SNA Server 4.0, COMTI for CICS and IMS, and OLE DB provider for VSAM and AS/400, see the SDK documentation included with SNA Server 4.0 SP2.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

APPENDIX A

ASP Best Practices

The best practices presented here will help you develop Active Server Pages (ASP) applications with well-organized, secure directories and files, and with scripts that execute efficiently. Follow these guidelines to create ASP pages that are styled for consistency, readability, and ease of maintenance.

For best results, you should be familiar with the material in the IIS 5.0 online product documentation, especially the "Building ASP Pages," and "Developing Web Applications" topics in the "Active Server Pages Guide."

This appendix is presented as a sequence of topics that parallels the planning and development of an ASP authoring project. For information about developing software components, see "Developing Web Applications" in this book and in the IIS 5.0 online product documentation.

In This Appendix

When to Use ASP

Project Directories and Files

Style Guide for Scripts in ASP pages

HTML Standards

Scripting for Performance

Additional Resources

When to Use ASP

ASP is a server-side scripting environment that makes it easy to deliver dynamic Web pages to users as they request them. With ASP, you can customize the user experience in a number of ways:

- Deliver pages that are formatted for the user's browser make and version.
- Gather data submitted on HTML forms, and connect browsers with information resources on the server-side, by using COM components for database update or information retrieval.
- Develop Web applications that connect browsers with data resources, such as databases on Microsoft SQL Server or IBM DB2.

Because ASP is optimized for multiple threads and multiple users, you can expect better performance with ASP compared to CGI.

Customizing Information Returned to Users

ASP makes it easy to collect data from users anywhere on the network. This data can be used to customize information, which is sent back to the users via ASP. For example, users could submit their preferences for a home (number of bedrooms, price range, location, and more) to a real estate site, by using HTML forms. ASP would submit a request to the database for only those homes that fit the customers' preferences.

Updating Records Online

ASP can use data from forms that are submitted by browsers in order to maintain databases. For example, if employees are allowed to access an online personnel database, they can update their own data within the personnel directory.

When Not to Use ASP

Do not use scripts in ASP pages in order to perform tasks that are readily accomplished by browser scripts, since ASP consumes server resources. By scripting the browser to perform tasks such as data validation, calculations, and selection of simple conditional output, you can save server resources for those tasks actually requiring ASP. For example, you could use a client-side script in order to validate the input data and to calculate monthly payments for a user who enters a loan amount and specifies the length of the repayment cycle.

To make the most efficient use of your server resources, use the .asp file name extension for pages that contain scripts or that are likely to need scripting in the future. Use the .htm extension for pages that will contain just HTML statements as well as scripts interpreted by the browser.

Project Directories and Files

Organizing Application Directories and Files

This section provides a model for storage structure and access permissions for ASP applications. Using this model will help you establish consistency, security, and ease of maintenance.

The organization and attributes given to the directories and files in the list below are more important than the names used.

```
/Application_Name
  Default.htm
  Global.asa

/Classes

/Content

  /ASP
    *.asp

  /HTM
    *.htm

  /Images

  /Media

  /Themes
```

`/Data` (not in site directory)

```
/DLL (not in site directory)
  *.dll
```

`/Helper_Files` (not in site directory)

Application Root Directory

The application root directory name should clearly convey the theme of the site. For example, an application for financial research might be named /Financial_Research. Avoid application root names that might be misidentified as standard subdirectories of a site, such as /Media or /Content. Also, avoid names that read like part numbers or codes, such as /FR98346A.

To avoid adversely affecting production sites, develop the application in a development test environment. An easy way to do this is to develop new applications under the IIS HTTP root directory, /InetPub/wwwroot, then move them to the same directory under /InetPub/wwwroot in the production environment when they are ready.

> **Note** /InetPub/wwwroot is the home location for all Microsoft® Visual InterDev® and Microsoft® FrontPage® Web documents. Moving a Web application to a storage location that is not under /InetPub/wwwroot makes it inaccessible to these tools.

The root directory of every application should contain at least these files:

- Default.htm or Default.asp
- Global.asa

Default.htm, Default.asp

Default.htm or Default.asp should be the default home page for the application, and the server default should be set accordingly using Internet Services Manager. This enables users to find sites in your organization consistently, by typing the server address plus the application root directory name. For example, a user can access MSDN Online by entering http://msdn.microsoft.com/default.asp. Entering the name of the home page is not necessary.

Global.asa

The file Global.asa specifies event scripts, declares objects that have application or session scope, and declares type libraries. For example, Global.asa scripts make application- and session-scope variables available at startup. Global.asa must be stored in the application root directory.

/Classes

The /Classes directory holds Java classes used by the application.

/Content

The /Content directory holds all pages (except Default.htm) and media that might be retrieved by a user of the site.

/ASP

The /ASP subdirectory of /Content contains all pages with server-side scripting. This directory must contain execute permissions so that ASP can execute the page scripts; storing all scripted pages here simplifies permissions management and site security.

/HTM

The /HTM subdirectory of /Content holds all pages containing only standard HTML. This directory is read-only and does not have execute permissions. A page containing server-side scripts that is stored here will not execute.

/Images

The /Images subdirectory of /Content contains graphics that are used independently of theme-related images, such as standard buttons and icons.

/Media

The /Media subdirectory of /Content contains subdirectories for audio, images, animation files, .avi files, and similar items used throughout the application.

/Themes

The /Themes subdirectory of /Content is used in order to enable application-wide changes to the look of a site. The subdirectory contains style sheets, bullets, buttons, icons, rules, and similar items, and should be organized so that you can easily change the look of an application by modifying any or all the theme-related items. Each item in the /Themes subdirectory can be linked dynamically by setting an application variable to its virtual path.

/Data

The /Data directory contains all database access information such as SQL scripts, file-based dataset names or similar data needed by the application. Do not place this directory under the site directory, as this could enable an unauthorized user to access business rules and private data.

/DLLs

The /DLLs directory contains Microsoft® Component Object Model (COM) components and Microsoft® Visual Basic® 6.0 run-time DLLs, such as Vbrun500.dll and Msvbvm50.dll. Do not place this directory under the site directory, as this could enable an unauthorized user to access business rules and private data.

Helper Files

Helper files are server-side include files or text files that make HTML-coded information available across the application. For security reasons, the directory containing helper files should not be stored in the published Web space (the Web site directories identifiable to users).

Using File Name Extension Standards

This section presents conventions for standardizing file name extensions, accounting for the types of files containing scripts, or a combination of HTML and scripts, including Microsoft® Visual Basic® Scripting Edition (VBScript) or Microsoft® JScript®.

Extensions for Page Files

Standards:

.asp—for ASP pages containing scripts

.htm—for static HTML pages

You must use the .asp extension for pages that contain server-side scripting, or that are likely to in the future. In order to save time and resources when serving pages, use the .htm extension for files that do not, and will not, require server-side script execution.

Extensions for Included Files

For consistency, use include files in order to make specific information available to more than one referring page (changes to include files are distributed to all the pages that include them).

Standards:

.inc—for large amounts of data with client-side scripting

.txt—for text-formatted data files without scripting

Do not use .inc for pages containing server-side scripts. If a user manages to display them, any business rules in the scripts will be exposed. Use the .asp extension for all pages containing scripting, or for which scripting is planned, in order to avoid displaying proprietary information coded as scripts in ASP pages.

Style Guide for Scripts in ASP Pages

The following conventions, which apply to the development of ASP pages containing scripts written in VBScript or JScript, are designed to enhance consistency, readability, and ease of maintenance.

In this guide:

- Blank Lines in Files
- Comments in Scripts
- Constant Names
- Context Switching
- Delimiting Lines of Script
- Indentation
- Language Default
- Layout Order of Scripts
- Object and Procedure Call Names
- Paths, Using **MapPath**
- Select Case Statement
- Spaces in Scripts
- Statement Styles
- String Concatenation
- String Function
- Variables: Case Values, Declaration, Names, Value Trimming

Blank Lines in Files

In general, use blank lines sparingly. You should remove all blank lines at the beginning and end of each file, but you might want to include one blank line between statements to increase readability.

Comments in Scripts

Comments should help any script author looking at code begin to understand it immediately. In addition, they should explain the intent of the code or summarize what the code *does*, not simply repeat what the code *says*.

General Comments

Write consistent comment blocks near the top of each page. In these blocks, list the file name, the group developing the file (not the person—e-mail should go to a group alias), the date the file was developed, the HTML and scripting standards followed, and dated descriptions of all changes made.

Comments To Explain Obscure Code

Use comments to explain obscure or complex code—any coding that would take a script author more than a few seconds to decipher. Do not leave a phrase such as the following without a comment:

```
If Err = LOCK Then
```

Scripts that are commented out should be deleted unless they are placeholders, in which case they should be labeled as such.

Comment Placement

Insert each comment with its corresponding code.

- Inline comments should appear two spaces after the corresponding code.
- Comments beginning on a new line should be set off with a blank line.

Example:

```
<%
  Dim intVariable  'Explicitly declare variable.

  'Assign the variable an integer value.
  intVariable = 5
%>
```

Blocked Comments in VBScript

If a single comment spans multiple lines, each line must begin with the standard VBScript comment symbol ('). Large, multistatement comment blocks should be formatted as in the example below.

Example:

```
Sub ShowIt()

  '=======================================
  'This procedure is called when the
  'user selects a language.
  '
  'It displays an appropriate select
  'item based on their language choice.
  '
  'The method choices are each contained
  'in a separate div.
  '=======================================

  Dim vntCurrLang
  vntCurrLang = document.all.langselect.value
  Select Case vntCurrLang
  Case "C"
    document.all.cdiv.style.display = ""
  Case "VB"
    document.all.vbdiv.style.display = ""
  Case "J"
    document.all.javadiv.style.display = ""
  End Select
End Sub
```

Blocked Comments in JScript

Multiline comments in JScript begin with /* and end with */. Large, multistatement comment blocks should be formatted as in the example below.

Example:

```
function showIt()

  /*****************************************
  ** This is a large comment block.
  **
  ** This procedure is called when the user
  ** selects a language.
  **
  ** It displays an appropriate select item
  ** based on their language choice.
  **
  ** The method choices are each contained
  ** in a separate div.
  *****************************************/

{
  var vntCurrLang = document.all
  ...
}
```

Constant Names

Use all uppercase when naming constants to distinguish them from other elements. An underscore can be used to separate terms when necessary.

Example:

```
   Const MIN_QUAL = 25
```

Context Switching

For readability, try to minimize switching between HTML and scripts. When possible, use a few large blocks of script on a page instead of several scattered fragments.

Delimiting Lines of Script

Lines of script should be blocked between a pair of delimiters, which are placed on a separate line, rather than written with delimiters on each line.

Instead of this:

```
<% strEmail = Session("Email") %>
<% strFirstName = Request.Form ("FirstName") %>
<% strLastName = Request.Form ("LastName") %>
```

do this:

```
<%
  strEmail = Session("Email")
  strFirstName = Request.Form ("FirstName")
  strLastName = Request.Form ("LastName")
%>
```

For a single stand-alone line of script, keep the delimiters on the same line as the script.

Example:

```
<% strEmail = Session("Email") %>
```

If the line of script is an ASP output directive (consisting of an equal sign and a variable), do not use a space between the equal sign and the delimiter.

Example:

```
<%= strSubscrLName %>
```

If the line of script specifies an ASP language directive, do not leave a space between the @ and the delimiter. Do not leave spaces around the equal sign.

Example:

```
<%@ LANGUAGE=VBScript %>
```

Indentation

Indentation makes the logical structure of the code more clear.

Once again, place a script consisting of more than one line on a line below the script delimiter, blocking and indenting it two spaces. Place a single-line script on the same line as the delimiter.

Indent everything between ASP delimiters (<% ... %>) at least two spaces, except procedures (functions and subroutines).

Also indent two spaces:

- Each break in logic
- Nested statements and HTML elements
- The body of a function
- The body of a loop from its controlling code

The following examples illustrate some of the indentation rules for scripts written in either VBScript or JScript.

Single-Line Script

```
<% Dim strLastName %>
```

Script with Nested Logic

```
<%
  'This example demonstrates a script with
  'a nested block of logic.

  Dim vntOutput
  Set vntExample = Server.CreateObject("MyComponents.Component.1")
  vntOutput = varExample.Text

  If vntOutput = "" Then
    Response.Write "An error has occurred"
  Else
    Response.Write vntOutput
  End If
%>
```

Function in VBScript

```
<%
  Function CalcMortgageRate()
    Statement_1
  End Function
%>
```

Function in JScript

```
<%
  //This is an example of a function.

  function calcMortgageRate()
    {
      statement1
      statement2
    }
%>
```

Language Default

Override the server's default scripting language (VBScript) by using the **@LANGUAGE** directive. For example, to ensure that the default language is JScript, put the following code at the top of each ASP page:

```
<%@ LANGUAGE=JScript %>
```

To use the <SCRIPT> tag for server-side processing, use the RUNAT attribute:

```
<SCRIPT LANGUAGE="Script Language" RUNAT=SERVER>.
```

Note that no spaces are used adjacent the equal signs.

Layout Order of Scripts

The following list summarizes the recommended layout of scripts in an .asp file, *proceeding from top to bottom on the page*. Well-ordered scripts produce more readable pages and, in some cases, cleaner execution. The list applies to both VBScript and JScript unless otherwise noted.

- Specify the language.
- Use the **Option Explicit** statement (VBScript only).
- List function library includes.
- Declare page-scoped variables.
- Assign values to page-scoped variables.
- Write HTML and inline scripting.
- List functions called by inline scripts.

Layout Order

```
<%@ LANGUAGE=VBScript %>
<% Option Explicit %>

  <HTML>
    <HEAD>
      <TITLE>Variable Sample</TITLE>
    </HEAD>

    <BODY BGCOLOR="White" topmargin="10" leftmargin="10">

      <!-- Display Header -->

      <FONT SIZE="4" FACE="Arial, Helvetica"> <B>Variable Sample</B></FONT>
      <BR>

      <HR>

      <H3>Integer Manipulation</H3>

    <%
      'Declare variable.
      Dim intVariable

      'Assign the variable an integer value.
      intVariable = 5
    %>
.
.
.
```

For VBScript, use **Option Explicit** to force explicit variable declaration. This prevents misspelled variables from causing unexpected results when the script executes.

To declare a transactional page, add the **<%@ TRANSACTION=value %>** directive to the first line on the page.

Transactional Page

```
<%@ TRANSACTION=Required LANGUAGE=VBScript %>
<% Option Explicit %>
<HTML>
  <HEAD>
.
.
.
```

Object and Procedure Call Names

To distinguish object names and procedure calls from elements such as variables, begin each object name or procedure call with a verb. Use initial capitalization for each term within an object name or procedure call. Table A.1 suggests some naming conventions that could be used to name objects for some typical activities.

Table A.1 Naming Conventions for Objects and Procedures

Name	Function	Example
AddNew	Adds new records	Customer.AddNew
Update	Edits records	Customer.Update
Remove	Deletes records	Customer.Remove
Get	Returns row from database	Customer.GetNewest
List	Returns multiple rows	Customer.ListNew

To specify returns from methods that return information, use **From** and **For** with a method or function name.

Name	Function	Example
GetFor	Returns criteria-based row	Customer.GetForName
ListFor	Returns criteria-based multiple rows	Customer.ListForPeriod

Object Naming

Use initial capitalization for each term when naming objects, including built-in objects. Use descriptive names for objects, even though this requires more characters per name.

The following example conforms to this naming convention ("cnn" is the prefix for an ADO connection variable):

```
Set cnnSQL = Server.CreateObject("ADODB.Connection")
Set cnnSQLServer = Server.CreateObject("ADODB.Connection")
```

These names do not conform:

```
Set cnnS = Server.CreateObject("ADODB.Connection")
Set cnnsql = Server.CreateObject("ADODB.Connection")
```

Paths, Using MapPath

Consider using the **MapPath** method instead of literal paths in ASP applications. The ASP **Server.MapPath** method allows you to physically relocate an ASP application without recoding scripts. This saves program modification and maintenance effort.

Performance is affected slightly, because every time you use **MapPath** in a script, IIS must retrieve the current server path. Consider placing the result of the method call in an application variable in order to avoid retrieving the server path.

Select Case Statement

For readability and efficiency, use the **Select Case** statement in place of **If...Else** in order to repeatedly check for the same variable for different values. For example:

```
<%
  Select Case intYrsService
    Case 0 to 5
      strMessage = "You have ten days paid leave this year."
    Case 6 to 15
      strMessage = "You have fifteen days paid leave this year."
    Case 16 to 30
      strMessage = "You have twenty days paid leave this year."
    Case 31 to 100
      strMessage = "Will you never leave?"
  End Select
%>
```

Spaces in Scripts

To enhance script readability, use spaces before and after operators, such as plus (+), minus (-), and equal (=).

Example:

```
intYearsService = intYearCurrent - intYearFirst
```

Also use spaces after commas, as when passing parameters or declaring more than one variable.

Example:

```
Dim intYearsService, intYearCurrent, intYearFirst
```

Note The use of spaces between arguments in ADO connection strings is invalid and will result in an error.

Statement Styles

Each scripting language has its own conventions for capitalization, indentation, and other style-related characteristics. Since VBScript is case-insensitive, capitalization conventions can be devised to improve readability, as the following suggestions illustrate.

If...Then...Else...End If statements:

- Capitalize the first letter of **If**, **Then**, **Else**, and **End If**.
- Indent the statements following **If**, **Then**, or **Else** two spaces.
- Put spaces at each end of an equal (=) sign.
- Avoid using unnecessary parentheses.

Correct example:

```
<%
  If Request("FName") = "" Then
    Response.Clear  'Not required if Response is buffered.
    Response.Redirect "test.html"
  Else
    Response.Write Request("FName")
  End If
%>
```

Similarly, capitalize the first letters of function and subroutine statements, and indent their definitions two spaces.

Example:

```
Sub SessionOnStart
  Session("MyId") = Request.ServerVariables(…)
End Sub
```

Avoid underscores.

Example:

```
Dim FirstName, LastName
```

String Concatenation

For the sake of consistency and to achieve the intended interpretation, use the string concatenator (&) instead of a plus (+) sign in VBScript strings.

Instead of this:

```
WholeName = FirstName + " " + LastName
```

do this:

```
WholeName = FirstName & " " & LastName
```

String Function

Use the **String(number,character)** function to create a character string consisting of repeated characters. For example, to create a string of 12 asterisks:

```
Dim strAstString
strAstString = String(12,"*")
```

The **String()** function takes character codes and string expressions as arguments, but is less verbose than the **For...** loop.

Variable Case Values

Keep cases consistent in both variable assignment and logical tests by using **UCase()** or **LCase()**. This is especially important when assigning and logically testing HTML intrinsic form controls, such as check boxes and radio buttons.

Variable Declaration

Explicitly declaring variables helps expose errors, such as misspelled variable names. To make code more reliable and readable, use the **Option Explicit** statement in VBScript.

When you want to use strong variable typing, the logic should be programmed into a component built with a language that supports it, such as Visual Basic 6.0 or Microsoft® Visual J++®. Loosely typed variables (not typed until run time), such as variants in VBScript, can affect performance, especially when mathematical computations are involved.

Variable Names

To make the intended use of a variable clear to others reading your script, use three-character prefixes in lowercase to indicate data type. Even though explicit data typing is not supported in either VBScript or JScript, the use of such prefixes is recommended.

For consistency in naming variables, use initial capital letters in variable names. Do not capitalize prefixes. For example, to denote the data type of the variable named "SwitchOn" as Boolean, use the prefix "bln," as found in Table A.2, to name the variable "blnSwitchOn".

Table A.2 Suggested Prefixes for Indicating the Data Type of a Variable

Data Type	Prefix
ADO command	cmd
ADO connection	cnn
ADO field	fld
ADO parameter	prm
ADO recordset	rst
Boolean	bln
Byte	byt
Collection object	col
Currency	cur
Date-time	dtm
Double	dbl
Error	err
Integer	int
Long	lng
Object	obj
Single	sng
String	str
User-defined type	udt
Variant	vnt

To keep variable name lengths reasonable, use standardized abbreviations. For clarity, keep abbreviations consistent throughout an application or group of related applications.

Instead of:

```
strSocialSecurityNumber
```

use:

```
StrSSN
```

Variable Value Trimming

Be consistent when trimming values. Trim numeric values to the desired length before putting them in state. This will eliminate errors in processing caused by inconsistencies in trimming schemes. For example, a value such as 9.997 used repeatedly as a multiplier would accumulate a different result than the trimmed value 9.9. Trim unneeded leading and trailing spaces from strings by using LTrim, RTrim, or Trim, to eliminate the possibility of a space causing a processing error or a display misalignment.

HTML Standards

ASP applications serve dynamic information in standard HTML format, allowing you to customize information for an audience that uses a wide range of browsers.

Supporting Text-Only Browsing

Many users browse the Web in text-only mode in order to speed up the display of information to their browsers. To ensure that you present as much content as possible to these users, take the following steps to support text-only display:

- Include text-only navigation objects, such as menus.
- Include the alternative (ALT) parameter with each image tag to provide information about images.

For example:

```
<IMG SRC="gravity.jpg" ALT="Picture of apple falling">
```

When providing client-side image maps, add a text-only list of the mapped areas, with links to the information that would have been loaded if the user had clicked on a map link.

Checking HTML Files

You should check and debug your HTML code, using either a dedicated HTML checker or an HTML editor that has code-checking features. Choose an editor that helps enforce your HTML version standard. Check each new HTML file as it is developed. Then debug your files again each time after they are modified.

To debug a page that contains scripts

1. Run the page that contains scripts.
2. View the source.
3. Save the output, which is pure HTML.
4. Run the saved output through an HTML checker.

This process is especially important for scripts in ASP pages that include forms. In these cases, HTML errors might corrupt the browser collection values sent from the browser to the server, causing a run-time error.

Using the 216-Color Palette

Color palette mismatches are ever-present concerns when you are creating images for a multiplatform Web. Even images in formats that require compact palettes, such as GIF (256 colors maximum), often display disparate colors when viewed on different platforms, such as Macintosh operating system, Windows 95, and UNIX.

To ensure that your images display the same colors, regardless of browser or platform, use the browser-safe, 216-color palette (also called the safety palette or the 6x6x6 palette). This palette allows nearly as many colors to be viewed as a GIF image can display, and displays them consistently across different systems.

For more information about the browser-safe palette, see http://webdesign.miningco.com/msubcolor.htm. Alternatively, enter "216 color palette" into the search field of any popular Web search engine.

Designing for Small Screens

Small-screen formats are still the standard for many users. Although larger formats are making progress, even 800 x 600 pixels is too large to fit on millions of displays. For example, there are millions of older Macintosh-compatible displays currently in use, many of which display a maximum of 640 x 480 pixels.

To accommodate a broad spectrum of users, including those using small screens, design for 640 x 480 pixels. For usability with small screens, keep the average line of text to approximately 12 words.

Displaying Standard Error Messages

To ensure consistency and to make error messages informative, use standard **Response.Status** values, such as "401 Unauthorized – Logon failed" and other IIS standard responses in your pages. For more information about how to customize error messages, see the IIS 5.0 online product documentation.

Using Object Statements with Embed Statements

To effectively deliver interactive objects to multiple browser types, write for browsers that do not support the HTML <OBJECT> tag.

To script the use of interactive objects, Microsoft® ActiveX® controls, or Java applets in HTML pages designed for a wide range of browsers:

- Use the <OBJECT> tag to place the object on the page.
- Add the <EMBED> tag for browsers that do not support the <OBJECT> tag.
- Add a display object using the <NOEMBED> tag for browsers that cannot "play" the object.

The following example places a ShockWave control onto a page, and provides for the contingencies just mentioned.

```
<OBJECT ID="ShockedPMDauntless"
  CLASSID="clsid:59CCB4A0-727D-11CF-AC36-00AA00A47DD2"
  CODEBASE="http://fabrikam.microsoft.com/marketing/movers/…"
  WIDTH=180 HEIGHT=120>
  <PARAM NAME="swURL" VALUE="dauntless.dcr">
  <EMBED SRC="dauntless.dcr" ALT="Movie of Fabrikam Dauntless model in action"
    WIDTH=180 HEIGHT=120>
  </EMBED>
  <NOEMBED>
  <IMG SRC="dauntless.gif" ALT="Picture of Fabrikam Dauntless model in action"
    WIDTH=180 HEIGHT=120>
  </NOEMBED>
</OBJECT>
```

Scripting for Performance

Object and Variable Initialization

The following information will help you to initialize and set dimensions for objects and variables, in order to achieve faster execution and efficient use of server resources. Unless stated otherwise, VBScript is assumed.

Scoping Variables

Using Page Scope for Best Performance

Local variables reside within functions and subroutines. Give page scope (also called local scope) to variables unless you have a compelling reason to use a broader scope. For example, you might want to assign session scope to a variable that is used in more than one script in a user session. Local variables are compiled into table entries. At run time, references to local variables are resolved with fast-executing table lookups, giving local variables faster performance than global variables.

Using Global Variables Sparingly and Efficiently

Global variables are resolved at run time, and execute much more slowly than local variables. An undeclared global variable is the slowest, requiring a search of the entire variable list the first time it is used. When you need to give global scope to variables, declare them using the **Dim** statement before using them. This saves valuable time on first use by eliminating a search of the entire variable list.

Avoiding the Use of Public Variables

Do not use variables defined as **Public**. The **Public** keyword is under review to determine future use. Use **Dim** instead.

Using Application Scope for Objects

Information stored in variables with application-wide scope is in memory and is cached for access by any page in the application. Give application scope to variables used often within an application, if the values do not change frequently.

Whatever the potential benefits, use caution in deciding whether or not to give application scope to an object. It can potentially affect performance and decrease reliability (your application could stop responding). To get the best performance using objects with application scope, set threading at BOTH.

Avoiding the Use of Server Variables

It is a best practice to avoid using server variables if your application does not need them. Whenever your ASP application accesses a server variable, your Web site makes a request that retrieves the server's entire variable collection—not just the variable to be used. This causes a performance hit the first time a server variable is used. You can enforce this restriction by setting the **ENABLESESSIONSTATE** directive to FALSE.

Declaring Objects with the <OBJECT> Tag

For objects that may or may not be used in an application, it's often most efficient to declare the objects without creating them until they are referenced. To declare an object without actually creating it, use the <OBJECT> tag on the page instead of **Server.CreateObject()**.

The PROGIDs used by **Server.Create.Object()** do not force unique names, and thus create the possibility of name collisions. The <OBJECT> tag uses Class IDs, which are unique and tend to eliminate name collisions.

Also, the <OBJECT> tag is supported in Global.asa and can be used to define scope using SCOPE=Session.

Minimizing Re-Dimensioning of Arrays

When working with arrays, try to avoid the use of the **Redim** statement. When it is feasible, set the dimension of an array to its maximum size requirement, then leave it at that size. This might not work if memory constraints prevent you from permanently setting a huge array, or several arrays, at maximum. However, every time you **Redim** an array, your application will take a performance hit, especially when using the Preserve keyword.

In the following example, the **Dim** and **Redim** statements are used inefficiently:

```
<%
  Dim SomeArray ( )
  Redim SomeArray (3)
  SomeArray (0) = "yes"
  SomeArray (1) = "no"
  SomeArray (2) = "maybe"
  .
  .
  .
  'Code executes happily with the current array
  'but then the program needs a bigger array.
  Redim Preserve SomeArray (8)
  SomeArray (3) = …
  .
  .
  .
%>
```

If the array had been dimensioned at 8 to begin with, some memory would have been wasted, but there would have been a gain in speed.

Using The Dictionary Object for Lookup

The VBScript **Dictionary** object enables fast lookup of arbitrary key/data pairs. Because the **Dictionary** object gives you access to array items by key, it is fast for finding items that are not in contiguous memory, since you are specifying a key, rather than knowing where in the array the item is located.

Use the **Dictionary** object when you have set a high priority on fast searches of nonlinear data.

Using the ENABLESESSIONSTATE Directive

Using the **ENABLESESSIONSTATE** directive (set in Internet Services Manager) for your site enables the detailed tracking of user requests.

In order to save those resources that IIS uses to process scripts for pages not using session state information, set the **ENABLESESSIONSTATE** directive to FALSE for those pages:

```
<%@ ENABLESESSIONSTATE=False %>
```

Working with Connections

ODBC Connection Pooling

One of the potential bottlenecks in ASP application performance is connection management. If not managed properly, the opening and closing of connections can occur so frequently that reduced server performance will result. Microsoft® Windows® 2000 Server features built-in support for connection pooling; this reuses existing connections optimally in order to achieve faster application performance and graceful timeout management with less coding effort.

IIS 5.0 automatically manages connection pooling for you.

To use ODBC or OLE DB connection pooling

1. Configure the driver for the database to which you are establishing a connection.
2. In the Windows 2000 Server registry, check the **CPTimeout** property to verify that connection pooling is on; let connection pooling handle the connection logic.
3. If connection pooling is off, use RegEdit32 to turn it on in the Windows 2000 Server registry.

   Caution Do not use a registry editor to edit the registry directly unless you have no alternative. The registry editors bypass the standard safeguards provided by administrative tools. These safeguards prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. To configure or customize Windows 2000, use the programs in Control Panel or Microsoft® Management Console (MMC) whenever possible.

4. Open individual connections—in your ADO code—just before you need data access on each individual page.
5. Close connections as soon as data access activities are complete.
6. Disconnect stored recordsets by setting **ActiveConnection** property to **Nothing** before storing in Session state.

Timing Out Connection Requests

Busy data servers cause slowdowns in ASP database applications that request database connections. Use the **ConnectionTimeout** property of the **Connection** object. This limits the time your application has to wait in order for a connection request to complete. For descriptions and examples of how to configure the **ConnectionTimeout** property in the registry, see "Data Access and Transactions" in this book.

Browser Connections and ASP

Browser users often request an ASP page, then move on before the requested page has finished processing. Use the **Response.IsClientConnected** property to determine the connection status of the target browser. If the target browser is no longer connected, cut off ASP page processing to avoid wasting server resources.

The following ASP page demonstrates the use of the **Response.IsClientConnected** property to end execution of an ASP application (the code uses many seconds of CPU time, so test it on a system where it will not impact production performance):

```
<%@ LANGUAGE="VBSCRIPT" %>

<%
  Function IsConnectedAfter(Seconds)
    Dim StartTime 'time the function started
    Dim PauseTime 'time the pause loop started

    'Use PerfMon to monitor the CPU cycles on the Web server. You
    'will notice that if you click STOP in the browser, the CPU
    'will settle down sooner than if the loop had continued.

    IsConnectedAfter = True
    StartTime = Now

    Do While DateDiff("s", StartTime, Now) < Seconds
      PauseTime = Now
      Do While DateDiff("s", PauseTime, Now) < 1
        'Do Nothing.
      Loop
      Response.Write "."
      If Response.IsClientConnected = False Then
       IsConnectedAfter = False
        Exit Function
      End If

    Loop
  End Function
%>
```

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual InterDev 1.0">
<META HTTP-EQUIV="Content-Type" content="text/html; charset=iso-8859-1">
<TITLE>Document Title</TITLE>
</HEAD>
<BODY>

<H1>!!! WARNING !!!</H1>
<P>This page has code in it that may use <B>100% CPU cycles</B> for at least 30 seconds.
Do not run this code on a production server. Restrict its use to a test server.

<P>Use SysMon to monitor the CPU cycles on the Web server. Press STOP in the Web
browser, and you will see that the CPU cycles will settle down sooner than they would
have without checking the IsClientConnected property.

<HR>

<%
  If IsConnectedAfter(30) then
    Response.Write "<P>The client is still connected</P>"
  Else
    'The browser is no longer connected. This would be a
    'good place to abort any transactions, clean up any
    'variables, and so forth.
  End If
%>

</BODY>
</HTML>
```

IsClientConnected will only work if **Response** is buffered.

Visual Basic Applications as DLLs

When you convert Visual Basic applications for use in ASP, they should be run as DLLs (components), rather than being converted to VBScript. Visual Basic DLLs will generally run more efficiently than scripts written in any other scripting language. Encapsulate the Visual Basic code in DLLs and create them in your ASP pages by using <OBJECT> tags or **Server.CreateObject()**.

You should use Visual Basic 6.0 to create an ActiveX DLL that has its project properties set to run in **Unattended Mode**, **Apartment Model Threaded**, and **Multi-instance**.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server, as well as information about best practices for ASP.

Web Links

http://msdn.microsoft.com

The Microsoft Developer Network provides up-to-date resources on ASP as well as associated technologies for Web application developers.

http://www.15seconds.com

The site deals exclusively with ASP, providing references, focus areas, and newsgroups. Focus areas include learning the basics through specialized topics, such as security, performance, and debugging.

Books

Beginning Active Server Pages 2.0 by B. Francis, J. Kauffman, et al, 1998, Chicago: Wrox Press.

This book introduces the beginner to ASP basics and to scripting in ASP with VBScript.

Professional Active Server Pages 2.0 by A. Homer, A. Enfield, et al., 1997, Chicago: Wrox Press.

Professional Active Server Pages 2.0 fully describes the Microsoft server-side environment, in the context of developing Web applications with ASP. The book covers using and developing components, developing three-tier, data-driven Web applications, and integrating ASP with Internet technologies such as e-mail.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

APPENDIX B

Site Security Planning

To safely deploy enterprise Web applications, you will need security policies and practices that defend your intranet against intrusion, that protect the communication of sensitive information across the Internet, and that provide secure access to critical applications for customers and business partners.

This appendix describes how to set policies for securing Web resources, applications, and data in an intranet. It also presents scenarios for using Internet Information Services (IIS) 5.0 in order to secure communications, including e-commerce applications, over the Internet. For more information about security in general, see "Security" in this book.

In This Appendix

Assessing Threats to Security

Where to Spend the Effort

Making Policy

Additional Resources

Assessing Threats to Security

This section provides a framework for assessing threats to the security of a Web site and its assets. A Web site is considered to include one or more server platforms and associated Web services under unified administrative control.

To effectively plan the security of your Web site you must:

- Keep pace with changes in business that might require new security measures. For example, e-commerce will require encryption of private information sent over the Internet.
- Identify and assess threats to the security of online assets. For example, if you open your corporate intranet to access by employees from home, their user IDs and passwords are assets that will be made vulnerable to the threat of exposure on the Internet.
- Prioritize threats according to potential exposure and recovery costs. For example, if you allow customers to purchase services from your Web site, determine what assets would be exposed and what the cost would be to secure them.

In the emerging online business environment, accurate threat assessment is vital to achieving cost-effective security for assets shared over the Web within your organization, as well as among your business partners and customers.

Threat Identification

Identifying threats to your Web site includes creating inventories of assets, evaluating assets and potential losses, and recognizing where potential threats originate (from inside and outside your organization).

When your organization engages in business over the Web, potential threats become more numerous. Security plans must account for communication of information between your site and the intranet sites of your business partners and customers.

Threats increase as assets are deployed to new environments. Many sensitive information assets stored and used in traditional environments—such as corporate databases connected to corporate users by means of a local area network (LAN)—will also be deployed in relatively new environments—such as intranets and the Internet. The increase in the use of Transmission Control Protocol/Internet Protocol (TCP/IP) networks, including the Internet, has created new environments in which employees, business partners, and customers expose information assets to new security threats. Therefore, your network-based information will be potentially more vulnerable than ever.

The following list summarizes the criteria to use when assessing threats and potential damage to information that will be deployed on intranets and over the Internet. For an example, see "Where to Spend the Effort."

For each asset requiring security:

- Create an inventory of online assets requiring security. Identify all systems, applications, and information that will need protection from intruders.
- Specify who needs access to the assets (in-house users, business partners, customers, an anonymous public), and from where (your intranet, or the Internet).
- Estimate the relative value of the assets you are charged to protect. A useful approach is to measure value as the relative damage that would be suffered by the organization, if an asset were corrupted or lost. Consider the importance of each asset to the achievement of business goals, the need to maintain credibility with customers, and the cost of replacing these assets if they are lost.
- Specify the consequences to your organization of a successful intrusion (loss of data, loss of service).
- Estimate the maximum potential damage on a scale of 1 (low) to 10 (high). Take into account that both data loss and denial of service will damage your organization and its customers.
- Estimate the minimum cost of providing adequate security for each category of assets on a scale of 1 (low) to 10 (high).
- State any conditions or assumptions affecting threats to assets or cost of security. For example, on the condition that the organization will use Internet-standard encryption methods, you might assign a low-cost factor to securing private data transmitted over the Internet between browsers and servers.

Attacker Motives and Targets

Individuals and groups seeking to invade your site could be operating from any of a wide range of motives: to aid an organization competing against yours; to embarrass the site owners as a prank; to further a social cause (antitechnology, antibusiness); to get revenge (the attacker was fired, did not receive a bonus, hates the boss); or, to gain status among peers.

Intruders can attack any asset of value to your organization, such as the entire contents of the corporate database. Targets are not limited to trade secrets or information that could be sold illegally; they might also be servers or server-based services, such as e-mail.

Threats on Intranet

Intranets can expose assets to threats from both outside and from within an organization.

Threats from Outside

Intranets use Internet technologies, such as e-mail and Hypertext Transfer Protocol (HTTP), in order to provide corporate resources to employees. Because these technologies are based on the TCP/IP suite of protocols, intranets give employees easy access to corporate information resources via the Internet, from wherever they might happen to be: at home, on a business partner's site, or in a distant city.

One result of this access is that employees send identification and authentication data as well as other sensitive business information over public networks, as they take care of business from remote locations. When the logon and authentication communication is not secured by encryption, amateur programmers working outside the corporation can intercept and use this sensitive information against the company intranet.

The Danger from Within

Outsiders do not pose the only threats to corporate resources on your intranet.

When developing plans for protecting new Web applications, there is a tendency to focus exclusively on securing data that will be transmitted past the firewall and across the Internet. However, evidence continues to show that security threats from inside organizations are most significant. While this may change somewhat with the advent of e-commerce and business-to-business applications, potential threats from inside your organization will require continuous attention if they are to be minimized. See the sidebar, "Information Systems Employee Seeks, Gets Revenge," for an example of how one disgruntled employee caused long-term damage to a company's vital intellectual property.

Information Systems Employee Seeks, Gets Revenge

In 1996, a disgruntled Information Systems employee of a U.S. manufacturer of measurement and control instruments planted a logic bomb on a company LAN. The bomb detonated ten days after the man had been terminated, wiping out the company's research, development, and production software, including its backup systems. The company estimated the cost to recover the damage at $12 million (over the several years that would be required to redevelop the software).

The employee had been notified that he was to be terminated, but his network accounts had not been disabled until his termination date. He had ample time to plant the bomb before leaving the company. Had he wanted to do so, he could have opened a back door to the systems he used, which would have allowed him to commit further damaging acts.

This incident shows the need for creating and enforcing realistic security practices, in order to combat internal threats to assets. There is no way to guarantee that employees will not commit destructive acts. However, some policies and practices that could have helped minimize the risk of incurring damages (such as those just cited) include disabling employee access before termination, running antivirus software on each computer on the network, and monitoring the network to detect intrusive traffic. Because the employee, in this case, had root privileges to the systems on the network, the Information Technology team should have locked down all the systems he administered.

At a minimum, the following security policies should be in place, in order to avoid this kind of catastrophe:

- Disable the accounts of employees upon their notification of termination. Do not wait until their termination date. In this case, the logic bomb had been planted before the employee was actually terminated, but after he knew termination was imminent.
- Reinitialize security for systems that were under the terminated employees' control. Continuously monitor these systems for security violations, such as back doors that are left open. Terminated employees often use these as reentry points.
- Install and run antivirus software on each computer on the intranet.
- Back up systems, applications, and data; deploy off-site storage for archival backups.

Threats over the Internet

Businesses gain a competitive advantage both by using the Internet to share information and resources with key partners, and by transacting business with customers. However, with that advantage comes a security challenge: to protect enterprise data and private customer information as they are communicated over the Internet.

In the Internet business environment, information assets take on new forms and appear in unaccustomed places. You will need to account for such changes in form and placement as you build your inventory of assets. For example, if you plan to offer online purchasing to customers, your company will transmit and receive credit card numbers and other private information across the Internet. Formerly confined to file cabinets and an internal network, this information—now transmitted in data packets on public networks—is an old asset that will be transmitted in a new environment. You will need to account for this development in your inventory of all assets that must be secured.

Opening the corporate network to communication from users outside the firewall presents opportunities for amateur programmers to exploit your organization through:

- Software bugs that compromise security, leaving your site vulnerable if the bugs are not fixed.
- Inadequately secured data that is routed over the Internet, leaving your data exposed to interception and illegitimate use.
- Unsecured executable Web application code with system access, leaving your back-end data exposed to access from unauthorized users.

Identify any threats to your assets. Include potential perpetrators, the ways in which they operate, and the targets your organizational environment presents to them. Evaluate the severity of the threats, and the degree of harm that successful attacks could cause.

Threats Impact All Areas of Security

Doing business over the Internet will impact each area of security at your Web site: authentication, authorization (or access control), privacy and integrity, availability, auditing, and nonrepudiation.

Authentication

Customers will open accounts and connect with your firm over the Internet, using the online authentication scheme you choose. Easily read IDs and passwords could be vulnerable to interception as they are transmitted over the Internet.

Some would-be intruders also possess the tools to decrypt passwords that are encrypted. *Password-cracking programs* intercept encrypted password files, then match the passwords to encrypted known passwords by using a pattern matching method. The matches are stored for later use against the site that sent the original encrypted passwords. For example, intruders sometimes use a program named "L0phtcrack" to decrypt passwords from Server Message Blocks (SMB), which are intercepted as they are being transmitted across the Internet.

Spoofing is an insidious method used for gaining access to user IDs, passwords, and other private information. A spoofing operation uses network communications to fool the user into participating in an illegitimate event. The attacker sends what appears to be a legitimate Web server or network service link (spoofing the server or service) for the purpose of collecting important private information.

For example, an attacker might intercept an Internet communication, then send the user a dialog box with the false message that network service has been interrupted. The attacker then requests that the user log on again using his or her user ID and password. Spoofs can be used to hijack Telnet sessions, communications with boards and chat sessions, or e-mail transmissions.

Authorization

You will need to change your authorization (or access-control) policies and procedures in order to meet the new challenges of doing business over the Internet. Employees, customers, and business partners will need access to resources at your site, which probably includes executable content. Recent surveys show that up to 25 percent of successful intrusions into business intranets are perpetrated by employees and users who have no legitimate need for access to the areas they entered. They gain access to these resources mainly through faulty authorization schemes. Both policies and procedures are often inadequate to protect system resources from unwanted intrusion.

Your site will be vulnerable to intrusion from destructive unauthorized users and thieves entering it from the Internet, if your access-control scheme does not protect your resources—such as scripts in ASP pages—from read access by outsiders.

Privacy and Integrity

Customers and business partners will send IDs, passwords, financial account data, and other sensitive information to your site. The privacy and integrity of such information will be vulnerable unless it is properly protected as it is routed over the Internet. Privacy and integrity are closely related and are subject to the same threats:

- Data intercepted and read by a malicious Internet user compromises the privacy of the transmission. The attacker can now use the information for purposes counter to the owner's intentions.
- Once privacy is compromised, the integrity of the information is threatened. The data can be modified before being sent on to its destination.

Availability

Opening your site to Internet commerce can threaten the availability of the services it provides. Heavy use of server resources can slow service; deliberate acts by intruders can crash systems, resulting in denial of service. A lack of redundant systems and proper backups leaves Web sites vulnerable to prolonged outages.

Auditing

A recent U.S. government computer site attack exercise found that fewer than 5 percent of the successful attacks were detected by network and systems administrators, and that even fewer were reported to authorities.

As you begin to do business with relative strangers, it becomes increasingly important to know who has connected to your site, when, and what actions they took. Law enforcement officials might need information about clandestine users coming in from anywhere in the world. You will need continuous auditing of the events occurring on your site.

Nonrepudiation

There will always be customers who try to repudiate actions they have taken, such as claiming they did not make an online purchase. To discourage nonrepudiation, provide not only strong authentication and authorization for your site, but sufficient auditing for it as well. Without these measures, in combination with a strong liaison with law enforcement authorities, those who want to engage in nonrepudiation of business transactions will have ample opportunity.

Where to Spend the Effort

Why Is Security Difficult?

When developing priorities for site security, keep in mind the organization, as well as its assets that must be secured. You must provide extensive access to the same assets you are trying to secure. Balancing these two requirements makes creating effective security measures difficult, particularly on a limited budget. Use the following general rules for deploying and securing the assets on your site:

1. Make each asset accessible to everyone who should have access to it, whenever they need it.
2. Protect each asset from intrusion by anyone who should not have access to it.

The first rule promotes awareness that the organization's mission is something other than security. The second rule suggests an effective approach to developing security policy: a *least-access* approach.

A Least-Access Approach

A least-access approach to security means that you should lock down, turn off, or remove online assets that do not require online access. Furthermore, you should only allow access to resources to those who truly require it.

This approach tends to greatly reduce such calamities as loss of data and denial of service that are due to the unwitting actions of users who wandered into areas in which they did not belong. It also minimizes the number of potential easy entry points for unauthorized users. For example, you might want to open only Transmission Control Protocol (TCP) ports 80 (HTTP) and 443 (HTTPS) for access to your Web services, and turn off the others. Other examples include disabling guest user accounts, as well as restricting anonymous users to read-only access in well-defined areas of the site.

Most of your effort should be spent securing assets that are potentially under threat, and to which Information Technology staff or users need access. This requires that you prioritize threats, assigning the highest security needs to those assets whose loss could most damage the organization.

The Case of Exploration Air

For a more in-depth view of the thread-assessment process, consider a hypothetical example. A network administrator planned to secure a Web site at his company, Exploration Air. He began by analyzing the security requirements of a Web site that maintains customer information—including frequent flyer miles—for members of its Flyers Club.

Assigning Threats and Potential Damage

First the administrator listed the assets needing protection, the network environments from which users would gain access, and the threats to those assets. Linking assets to access (see Table B.1) leads to an important security issue: the need to protect assets outside the corporate firewall. An asset only available from the company's intranet is protected from outside intruders by the Exploration Airlines firewall. However, assets sent over the Internet are vulnerable to interception while being routed across a public network. Making this distinction helps identify security requirements and cost issues that can be resolved in later steps. Notice that for item 1 (in Table B.1), access is from the intranet *and* the Internet. This is assuming that Exploration Air employees are allowed to access their accounts from home using dial-up access and to logon via HTTP forms. Thus there is potential danger, due to employees who access the intranet from the Internet.

Next, the administrator considered how each asset might be compromised, and categorized the potential loss as either a loss of data or denial of service. Based on the high value the company attributes to its customers and to its own good name, the administrator placed a high damage score on losses of customer data, especially when customers' money might be involved. For clarity and simplicity, the administrator did not include information considered public, needing little if any protection.

Table B.1 Threats and Potential Damage

Web Site Threat Assessment for Site:_____ as of (Date):_____

	Asset/Access from	Threat Loss	Expected Maximum Damage
1	Employee user IDs	Acquired by malicious user	10
	Intranet, the Internet	Data loss, denial of service	"10" for net administrator IDs
2	Back-end SQL Server	Corrupted by malicious user	10
	Intranet	Data loss	"10" for corrupt current and backup data
3	Flyers Club IDs and passwords	Acquired by potential thief	9
	Intranet, the Internet	Data loss	Customers lose money if thief uses frequent flyer miles
4	Other backend customer data	Acquired by malicious user	9
	Intranet, the Internet	Data loss	
5	Web services	Brought down by malicious user	6
	Intranet	Denial of service	
7	Network operating system	Brought down by malicious user	6
	Intranet	Denial of service	
8	ASP page source	Acquired by malicious user	5
	Intranet, the Internet	Data loss	

Assigning Relative Costs to Security

Next, the Exploration Air administrator pulled together a team, in order to determine the relative cost of securing the online assets needing protection (see Table B.2). The team looked for security solutions that were cost-effective, assigning a factor of "1" for the lowest relative cost and "10" for the highest. The team favored security solutions that were integrated into the network operating systems, or that were based on Internet community standards. They assigned the solutions relative cost factors of "1" or "2." Assumptions were spelled out in the appropriate fields of Table B.2. These assumptions pointed to requirements that would help Exploration Air assess network operating systems, Web services software, and add-in security software that might be used to secure the site.

Finally, the team listed the areas of security that were directly involved in securing each asset. For example, securing asset number 3—Flyers Club IDs and passwords—would require effective authentication, authorization, integrity, privacy, availability, and nonrepudiation.

At this stage, the team faced an issue new to Exploration Air's business model: how to secure user logon and transmission of sensitive, private information over the Internet. Customers would be allowed to submit forms that would trigger the creation and maintenance of their own accounts at the Flyers Club site. These accounts would receive and hold personal information about the club members, as well as keep track of their frequent flyer credits (as credits were added and subtracted by the reservation system). Members would be able to download all the information contained within their own club records over the Internet.

Table B.2 The Relative Costs of Security

Web Site Threat Assessment for Site:_____ **as of (Date):**_____

	Asset/Access from	Threat Loss	Expected Maximum Damage	Expected Minimum Security Cost	Security Areas
1	Intranet user IDs	Acquired by malicious user	10	1 Assumes adequate firewall and operating system-integrated security	Authentication Authorization Auditing
	Intranet, the Internet	Data loss, denial of service	"10" for net admin IDs		
2	Back-end SQL Server	Corrupted by malicious user	10	1	Authentication Authorization Integrity Availability Auditing
	Intranet	Data loss	"10" for corrupt current and backup data		
3	Flyers Club IDs and passwords	Acquired by potential thief	9	2 Assumes end-to-end Internet standard encryption and authentication	Authentication Authorization Integrity and privacy Availability Nonrepudiation
	Intranet, the Internet	Data loss	Customers lose money if thief uses frequent flyer miles		

Continued

Table B.2 The Relative Costs of Security *(continued)*

Web Site Threat Assessment for Site:_____ **as of (Date):**_____

	Asset/Access from	Threat Loss	Expected Maximum Damage	Expected Minimum Security Cost	Security Areas
4	Other back-end customer data	Acquired by malicious user	9	2 Assumes Internet standard encryption and authentication	Authentication Authorization Integrity and privacy Availability Nonrepudiation
	Intranet, the Internet	Data loss			
5	Web services	Brought down by malicious user	6	4	Authentication Authorization Availability
	Intranet	Denial of service			
7	Network operating system	Brought down by malicious user	6	4	Authentication Authorization Availability
	Intranet	Denial of service			
8	ASP page source	Acquired by malicious user	5	2	Authentication Authorization Availability
	Intranet, the Internet	Data loss			

The Exploration Air case is an example of how to assess threats and costs, before establishing security policies and practices for any Web site. Once you know the assets to be secured, the threats to those assets, the relative costs of security solutions, and the areas of security affected, you can begin to outline policy goals and objectives.

Making Policy

Design your Web site security policies to achieve realistic goals at a reasonable cost. Although Web sites will differ from each other, they will share some fundamental goals relating to strength of security, its cost, and the means of achieving a secure site. To ensure this:

- Provide strong security that is consistent with access requirements.
- Certify that all personnel who administer security are fully competent to enforce security policy consistently and accurately. Make sure that all users accept their responsibility to comply with this policy.
- Control security implementation costs that are consistent with the need for strong security. Security must scale up efficiently as sites expand.
- Adopt technologies, standards, and practices that are adaptable to changing conditions and new developments.
- Choose technologies that allow you to fully integrate security monitoring and management into network and user account administration. A single interface for security and administration will enable you to have efficient and timely security monitoring.
- Adopt Internet community standards for communication between your Web site and Internet destinations, including the security of communication. The adoption of Internet standards yields low-cost start-up and good scalability, because the standards are widely supported by your customers and business partners.

Vigilance and Revision

Successful security planning results in policies that mandate constant vigilance and periodic revision.

Constant Vigilance

Effective security planning requires you to monitor and report all significant security-related events. It also requires that you audit the reports from the systems administration in a timely fashion. Planning leads to security policies and standards that support effective monitoring and review.

Develop security plans that, at a minimum, thoroughly require you to monitor the following events and situations inside and outside the site:

- All security-related network events, such as resource access activities and logon attempts
- New users and changes to user network authentication status
- Reports of employees who are to be terminated
- Changes in authorization (access control) to site directories and files
- The addition of, or changes to, organizational firewalls and network-wide authentication systems
- Forums that report on discovered network security holes for the systems in place at your site, as well as fixes for them
- Problem and incident reports from the Internet community

It cannot be overemphasized that the systems and applications you install will contain bugs that will likely be discovered elsewhere in the Internet community, before you know about them. Your vendors and the Internet community security forums will broadcast news as these problems surface, and as solutions are developed. Security policy must include the practice of diligently monitoring the forums that provide this information, as well as the fixes for them.

Here are two examples of forums that effectively track bugs and fixes for major network operating systems:

- For UNIX systems: bugtraq.
  To subscribe: send a message to listserv@netspace.org; no subject; in the message area type **subscribe bugtraq**.
- For Windows: ntbugtraq.
  To subscribe: send a message to listserv@listserv.ntbugtraq.com; no subject; in the message area type **subscribe ntbugtraq**.

Periodic Revision

Security plans and policies are effective only to the extent that they anticipate and counter potential threats. Establish a policy to periodically review your security plan, in light of changes in your organization's business practices.

New ways of using the Web, such as connecting your Web users to your organization's databases, will incur new vulnerabilities to threats—on your intranet and over the Internet. The scripts used to activate Web pages with database information contain code needed to open and query these databases. You will need to revise your security plan, in order to establish policies and practices that prevent unauthorized access to your proprietary Web application scripts (For an example, see the sidebar "Who Is Reading Your Server-Side Scripts?").

Who Is Reading Your Server-Side Scripts?

At some point in the near future you will probably want to provide your users with access to your server data over the Internet. You can accomplish this by writing scripts in ASP pages on the Windows® 2000 platform, utilizing the IIS 5.0 online product documentation as a resource. Browser users cannot view your scripts by viewing the page source, because the server-side scripting was removed before the page was sent to the browser.

However, you will compromise the security of your server-side scripts if you allow browsing of directories containing scripts in ASP pages, or server-side includes containing collections of script fragments. Here are some common mistakes to avoid:

- Files with the .asp extension must be placed in a directory with execute permissions set. It is a security risk to also set read permissions for these directories, because this permits easy pilfering of your original scripts in ASP pages.
- For efficient maintenance and ease of use, commonly used script fragments are often stored in a server-side include file. Include files use .stm, .shtm, or .shtml as standard extensions. Anyone who knows how to look for include files can download them if they are stored in a directory with browsing enabled.

To prevent users from downloading copies of any of these file types, establish and enforce a policy of always keeping them in directories that disallow directory browsing.

Adopting Technologies and Standards

Technologies and Standards for the Server Side

Use your primary security goals (such as those listed above) to select security technologies and standards for each area of site security: authentication, authorization, privacy, integrity, availability, auditing, and nonrepudiation. Build your security plans, policies, and practices around these technologies and standards.

Strong Authentication

Use authentication schemes that are integrated with your network operating systems, and that use Internet standard protocols. Examples:

- Network authentication protocols—such as the Kerberos v5 authentication protocol, a feature of Microsoft® Windows® 2000 Server security—distribute tickets that limit the exposure of passwords, and that authenticate users for network-wide access to resources. The Kerberos v5 protocol is a widely used Internet standard for network-wide authentication.
- Public-key client certificate authentication allows users to communicate across the Internet with your site, without exposing passwords or data that would be vulnerable to easy interception.

You might also need to support special functions such as smart-card authentication, or server certificates with public keys that allow users to authenticate your servers as trusted sources.

System-Integrated Authorization

In order to control access to resources, you will need authorization (access control) standards. Do not rely on application-level access to resources. Instead, use network-wide authorization services such as discretionary access control lists (DACLs) in Windows 2000 Server.

Network-wide authorization will make it easy for authenticated employees and customers to use the resources they need, while allowing you to efficiently control access to valuable resources.

Protected Privacy and Data Integrity

Select technologies that, by using encryption, are able to protect user privacy and data integrity across the network. Set a protocol standard for your site that is supported across the Internet community, such as:

- Secure Sockets Layer (SSL)
- Transport Layer Security (TLS)
- Internet Protocol Security (IPSec)

Assured Availability

Attacks that cause denial of service to users—such as crashing a server system—are difficult to defend against, or even to predict. Develop security policies that mandate clustering and solid backup practices, in order to provide the most availability to your users at the lowest possible cost.

Timely Auditing

A good auditing policy demands that you record events of interest that take place on your system and evaluate them in a timely fashion. Timely audit trails facilitate the pursuit of perpetrators, while delayed audit trails often lead to fixing the security problem when it is too late: after the perpetrator has completed all destructive actions.

Effective Nonrepudiation

Make it difficult for users to take action only to repudiate it later. When users can engage in business transactions and then deny that they have done so, they can cost your organization resources and diminish its good name with customers and business partners.

Requiring the signing of acceptable-use policies is the first step in discouraging employees and customers alike from wrongly repudiating the actions they might take. For example, in the acceptable-use policy for customers, state explicitly that customers are committing themselves to take full responsibility for orders they submit.

Client Security Management

To protect your site and the users who provide content for it, prevent users from downloading unsafe files over the Internet. Set and enforce browser and e-mail security for at least those users who provide content to your site. Establish a policy for maintaining a list of restricted sites, and set browser restrictions accordingly. Use a centralized browser management package to lock in browser security settings (see the sidebar, "Locking in Security Settings for Microsoft Internet Explorer and Outlook" for an example of how to do this).

Locking in Security Settings for Microsoft Internet Explorer and Outlook

Develop and implement an effective browser security policy in your organization in order to prevent browser users from downloading possibly dangerous content. For Windows:

1. Use Microsoft® Internet Explorer 5, or later, as the standard browser in your organization.
2. Establish a browser security policy that protects users against downloading unsafe active content. At a minimum, require that browsers be set on High security for the **Restricted sites** zone and on Medium security for the **Internet** zone. (From the **View** menu, select **Internet Options**, then click the **Security** tab. Then select from the **Zones** drop-down menu.)
3. Implement the policy, using the *Microsoft® Internet Explorer Administration Kit*. This policy will lock in effective security levels on all browsers in your organization.

Effective Internet Explorer security will also protect against downloading unsafe e-mail content when using Microsoft® Outlook® 98 as a mail client, because Outlook 98 uses Internet Explorer components (including the rendering engine) to enable its Web features.

Firewalls

If your Web site is but one of many within your organization, a corporate firewall placed between your intranet and the Internet will partially protect it from intrusion. The firewall protects your intranet or corporate LAN from intrusion, by controlling access from the Internet, or other large network.

Firewalls vary in their approach to providing security. *IP packet filtering* offers weak security, is cumbersome to manage, and is easily defeated. *Application gateways* are more secure than packet filters and easier to manage because they only deal with a few specific applications, such as a particular e-mail system. *Proxy servers* can provide application gateways, safe access for anonymous users, and other services.

Take advantage of the firewall security features that can help you. Your firewall administrator might be able to fine tune the firewall's access control in order to meet your site's needs. The best firewalls feature reports all attempts at unauthorized access. Use these reports in your own monitoring efforts.

Do not place sole reliance for Web site security on your corporate firewall. Above all, do not take the effectiveness of your corporate firewall for granted. Among the reasons to resist this temptation:

- Firewalls are fallible. They are often breached. The viruses designed to breach firewalls and wreak havoc on your site are called Trojan Horses for good reason: they get past the gate (the firewall).
- Firewalls are subject to constant technological change. As your organization upgrades its firewall, the firewall security scheme might change. Do you know what the new scheme entails? Does it meet your needs?
- Firewall security policy changes to meet changing needs. Are the security needs of your site included?

Whatever its security scheme, once the firewall has been breached, you must rely on your own site security measures to defend its resources against intruders.

Security Checklists

Make your Web site security policies complete and explicit. Link them to practices that include recording information in security checklists. Emphasize accountability by requiring signatures of employees who fill out the checklists.

Example: Security Initialization Checklist

Create a checklist for each server platform and the Web services running on it. Record items that impact security (see Table B.3):

- Storage formats used
- Bug fixes and service packages
- TCP port access information

You can use the sample checklist in Table B.3 to record security information for a Windows 2000–based server used as a Web site. The checklist reflects a least-access approach to security.

Table B.3 Sample Windows 2000/IIS 5.0 Security Initialization Checklist

1. **Server Initialization**

Computer name _____
Setup by Name (print): _____
Signature _____
Setup date _____

Computer manufacturer/model _____
CPUs, make, model, speed _____
Memory _____ Network card(s) _____
_____

Hard drive formatted in NTFS Yes __ No __
NTFS 8.3 Name Generation turned off Yes __ No __

Service Packs and hot-fixes applied		Date applied/reference
Windows 2000	_____	_____
	_____	_____
	_____	_____
IIS 5.0	_____	_____
	_____	_____
	_____	_____
SSL	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____
	_____	_____

2. **TCP Ports Access Limits**

Port 80 access by SSL only Yes____ No____
Port 443 access by SSL only Yes____ No____

TCP Notes (other ports and access methods used) _____
_____
_____
_____

Table B.3 Sample Windows 2000/IIS 5.0 Security Initialization Checklist *(continued)*

3. **Unneeded Services Log**

Service	Installed/Enabled		
FTP Publishing	Yes___	No___	Note_____
NNTP Service	Yes___	No___	Note_____
SMTP Service	Yes___	No___	Note_____
Content Index	Yes___	No___	Note_____
Certification Authority	Yes___	No___	Note_____
Plug and Play (recommended)	Yes___	No___	Note_____
RPC Locator (required for remote administration)	Yes___	No___	Note_____
Server Service	Yes___	No___	Note_____
Telephony Service	Yes___	No___	Note_____
Remote Access (required for dialup access)	Yes___	No___	Note_____
Alerter	Yes___	No___	Note_____
ClipBook Server	Yes___	No___	Note_____
Computer Browser	Yes___	No___	Note_____
DHCP Client	Yes___	No___	Note_____
Messenger	Yes___	No___	Note_____
Net Logon	Yes___	No___	Note_____
Network DDE and DSDM	Yes___	No___	Note_____
Network Monitor Agent	Yes___	No___	Note_____
Simple TCP/IP Services	Yes___	No___	Note_____
Spooler	Yes___	No___	Note_____
NetBIOS Interface	Yes___	No___	Note_____
TCP/IP NetBIOS Helper	Yes___	No___	Note_____
WINS Client (TCP/IP)	Yes___	No___	Note_____
NWLink NetBIOS	Yes___	No___	Note_____
NWLink IPX/SPX	Yes___	No___	Note_____

Pursuit

When a security incident occurs that requires a coordinated response from outside your site team, you should be ready to follow up with a plan and a policy. Be prepared to:

- Report incidents to your central authority. If your organization is large, it might have standard incident reporting procedures, and incident response teams to handle them. If that is the case, incorporate these procedures into your security incident reporting policy.
- Report incidents to the appropriate governmental authorities. Incidents involving attacks from outside your state or outside the U.S. should be reported to the FBI.
- Report problems to appropriate vendors and Internet security monitoring and coordination organizations, such as the CERT® Coordination Center (http://www.cert.org), bugtraq, or ntbugtraq.

In addition to a policy, you should have contact information for reporting incidents to all pertinent authorities.

Additional Resources

The following Web sites and books provide additional information about IIS 5.0 and about other features of Windows 2000 Server.

Web Links

http://www.ntbugtraq.com
: The ntbugtraq Web site hosts the leading Internet mailing list for tracking Windows NT and Windows 2000 Server–related bugs.

http://www.cert.org
: The CERT Coordination Center manages responses to security incidents and security vulnerability reports, and provides information about a wide range of security-related issues.

http://info.internet.isi.edu/in-notes/rfc/files/rfc2196.txt
: RFC 2196 is a site security planning document published by the Internet Engineering Task Force (IETF). Coverage includes policy formation and content, many technical network security topics, and incident response.

http://www.iss.net
: The Internet Security Systems Web site is a source of links to FAQs, mailing lists, newsgroups, and other security-related resources. The company describes its security products at this site as well.

http://www.microsoft.com/security/default.asp

This is the most comprehensive and up-to-date site dealing with Microsoft security. Coverage includes Microsoft's security bulletins, information about security for each Microsoft product and technology, and links to other security information resources.

http://www.sans.org/newlook/home.htm

The SANS Institute is an organization dedicated to helping computing and networking professionals share their experiences in solving the problems they face. The Institute provides publications and online information and hosts conferences and seminars on network security. Publications on Windows NT security include: *SANS NT Digest* and *Windows NT Security: Step-by-Step Guide.*

http://www.trustedsystems.com/NSAGuide.htm

Trusted Systems wrote *Windows NT Security Guidelines* through a year-long project for the National Security Agency (NSA). It provides guidelines for configuring Windows NT for optimum security, including government C2-level security. You can download the guidelines from this site.

http://www.w3.org/security

The World Wide Web Consortium (W3C) pages on security cover many important security issues and technologies. The site is also a good portal to other security-related sites.

Books

Extranets by Richard H. Baker, 1997, New York: McGraw-Hill.

A complete sourcebook on how to plan, implement and secure extensions to your intranet that reach out to business partners and customers.

Mastering Internet Security by Chris Benton, 1998, Cybex.

The book is a comprehensive guide to planning security, writing security policies, as well as choosing and using security tools for a multiplatform network.

Hacker Proof, The Ultimate Guide to Network Security by Lars Klander, 1997, Houston: Jamsa Press.

A great resource for understanding network security issues and learning what to do about them. Excellent descriptions of encryption and hash values.

Information in this document, including URL and other Internet Web site references, is subject to change without notice.

Glossary

A

access control
The mechanisms for limiting access to resources based on users' identities and their membership in various predefined groups. Access control is used typically to control user access to network resources such as servers, directories, and files.

access control list (ACL)
A list that indicates which users or groups have permission to access or modify a particular file; the Windows discretionary access control list (DACL) and system access control list (SACL) are examples of access control lists.

ACL
See **access control list**.

Active Directory Service Interfaces (ADSI)
A COM-based directory service model that allows ADSI-compliant client applications to access a wide variety of distinct directory protocols, including Windows directory service and Lightweight Directory Access Protocol (LDAP), while using a single, standard set of interfaces. ADSI shields the client application from the implementation and operational details of the underlying data store or protocol.

Active Group, The
A standards organization, under the auspices of The Open Group, which is an open, customer-driven steering committee responsible for the ongoing development and management of ActiveX technologies and licensing.

active script
A script that can be implemented in various languages, persistent formats, and so on, that can interact with other ActiveX Controls.

active scripting
A Microsoft technology that uses COM to run third-party scripts in Microsoft Internet Explorer without regard to language and other elements of implementation.
See also **Active Server Pages; Automation; Component Object Model component; script; scripting engine**.

Active Server Pages (ASP)
A server-side scripting environment that can be used to create dynamic Web pages or build Web applications. ASP pages are files that contain HTML tags, text, and script commands. ASP pages can call Component Object Model (COM) components to perform tasks, such as connecting to a database or performing a business calculation. With ASP, the user can add interactive content to Web pages or build entire Web applications that use HTML pages as the interface to your customers.

ActiveX
An umbrella term for Microsoft technologies that enable developers to create interactive content for the World Wide Web. A set of language-independent interoperability technologies that enable software components written in different languages to work together in networked environments. The core technology elements of ActiveX are the Component Object Model (COM) and distributed COM. These technologies are licensed to The Open Group standards organization, and are being implemented on multiple platforms. *See also* **Component Object Model; Common Gateway Interface; distributed COM; Java**.

ActiveX Controls
Reusable software components that incorporate ActiveX technology. These components can be used to add specialized functionality, such as animation or pop-up menus, to Web pages, desktop applications, and software development tools. ActiveX Controls can be written in a variety of programming languages including C, C++, Visual Basic, and Java.

ActiveX Data Objects (ADO)
A high-level data access programming interface to an underlying data access technology (such as OLE DB), implemented by using the Component Object Model (COM).

activity
A collection of COM objects that has a single distributed logical thread of execution. Every COM object belongs to one activity.

Address Resolution Protocol (ARP)
A TCP/IP protocol for determining the hardware address (or physical address) of a node on a local area network connected to the Internet, when only the IP address (or logical address) is known. An ARP request is sent to the network, and the node that has the IP address responds with its hardware address. Although ARP technically refers only to finding the hardware address, and Reverse ARP (RARP) refers to the reverse procedure, the acronym ARP is commonly used to describe both. ARP is limited to physical network systems that support broadcast packets. It is defined in RFC 826. *See also* **Reverse Address Resolution Protocol; Transmission Control Protocol/Internet Protocol**.

ADO
See **ActiveX Data Objects**.

ADSI
See **Active Directory Service Interfaces**.

ADSI Provider
An application that makes itself available to ADSI client applications by providing an ADSI implementation.

agent
In client/server applications, a process that mediates between the client and the server. In Simple Network Management Protocol (SNMP), agent information consists of comments about the user, the physical location of the computer, and the types of service to report based on the computer's configuration. *See also* **catalog agent**.

aggregation
A composition technique for implementing component objects whereby a new object can be built by using one or more existing objects that support some or all of the new object's required interfaces.

alias
A name that maps part of a URL to a physical directory on the server. In general, an easily remembered name used in place of an IP address, directory path, or other identifier; also called a friendly name. *See also* **host name; virtual directory; virtual server**.

American National Standards Institute (ANSI)
A voluntary, nonprofit organization of U.S. business and industry groups formed in 1918 for the development of trade and communication standards. It provides area charters for groups that establish standards in specific fields, such as the Institute of Electrical and Electronics Engineers (IEEE). ANSI is the American representative of the International Standards Organization and has developed recommendations for the use of programming languages including FORTRAN, C, and COBOL. Standards approved by ANSI are often called ANSI standards (for example, ANSI C is the version of the C language approved by ANSI). *See also* **ASCII; ASCII character set; ASCII file**.

American Standard Code for Information Interchange (ASCII)
A coding scheme using 7 or 8 bits that assigns numeric values up to 256 characters, including letters, numerals, punctuation marks, control characters, and other symbols. ASCII was developed in 1968 to standardize data transmission among disparate hardware and software systems and is built into most minicomputers and all personal computers.

annotation file
For the FTP service, a summary of the information in a given directory. This summary appears automatically to browsers.

Anonymous File Transfer Protocol (anonymous FTP)
Makes it possible for a user to retrieve documents, files, programs, and other archived data from anywhere on the Internet without having to establish a logon name and password.

anonymous-only logons
Allows remote access by the IUSR_*computername* account. Remote users can connect to that computer without a user name or password, and they have only the permissions assigned to that account. Anonymous access is typically used for Internet sites.

ANSI
See **American National Standards Institute**.

Apartment model multithreading
The Component Object Model (COM) supports a form of multithreading called the Apartment model. The apartment is essentially a way of describing a thread with a message queue that supports COM objects. Apartment model multithreading enables multiple application threads—one for each apartment—to be managed by COM.

Apartment thread
A thread used to execute calls to objects of components configured as "Apartment threaded." Each object "lives in an apartment" (thread) for the life of the object. All calls to that object execute on the Apartment thread.

API
See **application programming interface**.

application
A computer program, such as a word processor or electronic spreadsheet; or a group of Active Server Pages (ASP) scripts and components that perform such tasks.

application programming interface (API)
A set of routines that an application uses to request and carry out lower-level services performed by a computer's operating system. Also, a set of calling conventions in programming that define how a service is invoked through the application.

application root
The root directory for an application; all directories and files contained within the application root are considered part of the application. Also called an application starting-point directory.

application scope
A way of making data available to all users of an application from all pages of a Web application. A variable or an object instance is given application scope by storing it in the Active Server Pages (ASP) application object. Application scope is useful for global data, such as a global counter.

argument
A constant, variable, or expression passed to a procedure.

ARP
See **Address Resolution Protocol**.

array
A list of data values, all of the same type, any element of which can be referenced by an expression consisting of the array name followed by an indexing expression. Arrays are part of the fundamentals of data structures, which, in turn, are a major fundamental of computer programming.

ascii
In an FTP client program, the command that instructs the FTP server to send or receive files as ASCII text. *See also* **ASCII**.

ASCII
See **American Standard Code for Information Interchange**.

ASCII character set
A standard 7-bit code for representing ASCII characters by using binary values; code values range from 0 to 127. Most PC-based systems use an 8-bit extended ASCII code, with an extra 128 characters used to represent special symbols, non-English language characters, and graphic symbols.

ASCII file
Also called a text file, a text-only file, or an ASCII text file. An ASCII file contains characters, spaces, punctuation, carriage returns, and sometimes tabs and an end-of-file marker, but it contains no other formatting information.

ASP
See **Active Server Pages**.

ASP buffering
Functionality of ASP that temporarily stores all output generated by a script until script execution is complete, then sends it to a client.

associating
See **file name extension mapping**.

asynchronous transfer mode (ATM)
A network technology capable of transmitting data, voice, video, and frame relay traffic in real time. Data, including frame relay data, is broken into packets containing 53 bytes each, which are switched between any two nodes in the system at rates ranging from 1.5 to 622 Mbps. ATM is defined in the broadband ISDN protocol at the levels corresponding to levels 1 and 2 of the ISO/OSI model. It is currently used in local area networks involving workstations and personal computers. *See also* **Integrated Services Digital Network; International Organization for Standardization Open Systems Interconnection model**.

asynchronous transmission
In modem communication, a form of data transmission in which data is sent intermittently, one character at a time, rather than in a steady stream with characters separated by fixed time intervals. Each transmitted character consists of a number of data bits (the character itself) preceded by a "begin character" signal called the start bit, and ending in an optional parity bit followed by 1, 1.5, or 2 "end character" signals, called stop bits.

ATM
See **Asynchronous Transfer Mode**.

atomicity
A feature of a transaction considered or guaranteed to be indivisible. Either the transaction is uninterrupted, or, if it fails, a mechanism is provided that ensures the return of the system to its state prior to initiation of the transaction.

attributes
In a database record, the name or structure of a field. The size of a field or the type of information it contains would also be attributes of a database record. In markup languages such as Standard Generalized Markup Language (SGML) and HTML, a name-value pair within a tagged element that modifies certain features of that element.

auditing
The process an operating system uses to detect and record security-related events, such as an attempt to create, access, or delete objects such as files and directories. The records of such events are stored in a file known as a security log, whose contents are available only to those with the proper clearance. *See also* **security log**.

authentication
The process by which the system validates a user's logon information. A user's name and password are compared against an authorized list, and if the system detects a match, access is granted to the extent specified in the permission list for the user.

authentication certificate
See **certificate, digital**.

authorization
In relation to computers, especially to remote computers on a network open to more than one person, the right granted to an individual to use the system and the data stored on it. Authorization is typically set up by a system administrator, Web master, or site owner and checked and cleared by the computer. This requires that the user provide some type of identification, such as a code number or a password, that the computer can verify against its internal records. Also called permission or privilege.

automatic directory listing
Providing a directory listing by default when a URL without a file name is received; also called directory browsing.

Automation
A COM-based technology that enables dynamic binding to COM objects at run time. Automation was previously called OLE Automation and ActiveX Automation.

Automation object
An object that is exposed to other applications or programming tools through Automation interfaces.

B

bandwidth
The capacity of the transmission medium stated in bits per second (bps) or as a frequency (Hz). Generally, a higher bandwidth number indicates faster data-transfer capability. In communications, the difference between the highest and lowest frequencies in a given range. In computer networks, greater bandwidth indicates faster data-transfer capability and is expressed in bits per second (bps).

bandwidth throttling
Setting the maximum portion of total network capacity that a service is allowed to use. An administrator can deliberately limit a server's Internet workload by not allowing it to receive requests at full capacity, thus saving resources for other programs such as e-mail.

Basic authentication
An authentication protocol supported by most browsers, including Internet Explorer. It is a method of authentication that encodes user name and password data transmissions. Basic authentication is sometimes called clear-text authentication because the Base-64 encoding can be decoded by anyone with a freely available decoding utility. Note that encoding is not the same as encryption.
See also **Integrated Windows authentication; encryption**.

baud
A measure of data transmission speed. Commonly used to refer to the data transmission speed of a modem.

BIND
See **Domain Name System**.

binding
The way in which Microsoft Visual Basic code uses Automation to access objects in another application. *See also* **Automation; static binding; dynamic binding**.

Bits per second (bps)
The speed at which data bits are transmitted over a communications medium, such as a transmission wire or a modem.

Boolean
Of, pertaining to, or characteristic of logical (true or false) values. Many languages directly support a Boolean data type, with predefined values for true and false; others use integer data types to implement Boolean values, usually (although not always) with 0 equaling false and "not 0" equaling true. Queries with Boolean operators (AND, OR, NOT, and NEAR) are referred to as Boolean queries.

Boolean expression
An expression that yields a Boolean value.

Both-threaded
A component that supports Free- and Apartment-threading models. *See also* **Apartment thread**.

broken link
A reference to a resource that cannot be located because the URL is not valid, the resource the link points to doesn't exist, or the server containing the resource is busy or is having other technical difficulties.

browser
Also called a Web browser. A client interface that enables a user to view HTML documents on the World Wide Web, another network, or the user's computer; follow hyperlinks among them; and transfer files. One example is Microsoft Internet Explorer.

bulk data encryption
The encryption of all data sent over a network. *See also* **encryption**.

business rules
The laws, regulations, policies, and procedures that are encoded into a computer system. Also known as business logic.

bytecode
The executable form of Java code that executes within the Java virtual machine (VM). Also called interpreted code, pseudo code, and p-code.

C

CA
See **certification authority**.

cache
A special memory subsystem in which frequently used data values are duplicated for quick access. A memory cache stores the contents of frequently accessed RAM locations and the addresses where these data items are stored. When the processor references an address in memory, the cache checks to see whether it holds that address. If it does, the data is returned to the processor; if it does not, a regular memory access occurs. A cache is useful when RAM accesses are slow compared with the microprocessor speed, because cached memory is faster than main RAM memory.

call
To transfer program execution to some section of code (usually a subroutine) while saving the necessary information to allow execution to resume at the calling point when the called section has completed execution. When a subroutine call occurs, one or more values (known as arguments or parameters) are often passed to the subroutine, which can then use and sometimes modify these values.

callback function
A function provided by IIS that allows an ISAPI extension or filter to access IIS services.

caller
A client that invokes a method of an object. An object's caller isn't necessarily the object's creator. For example, client A could create object X and pass this reference to client B, and then client B could use that reference to call a method of object X. In this case, client A is the creator, and client B is the caller.

catalog agent
An automatic software program that periodically opens all files in a designated set of directories and indexes their contents; also called a link crawler.

certificate, client
A digital certificate that functions in a way similar to a driver's license or passport. Client certificates can contain detailed identification information about the user and organization that issued the certificate. *See also* **certificate, digital**.

certificate, digital
An encrypted file, containing user or server identification information, that is used to verify identity; also called an authentication certificate. When issued to users, a digital certificate is called a client certificate. When issued to a server administrator, it is called a server certificate. *See also* **key pair; certificate, client**.

certificate revocation list
A document maintained and published by a certification authority (CA) that lists certificates that have been revoked by the certification authority. *See also* **certification authority**.

certification authority (CA)
An entity that issues, manages, and revokes certificates.

CGI
See **Common Gateway Interface**.

class
In Microsoft Visual Basic Scripting Edition (VBScript), the formal definition of an object. The class acts as the template from which an instance of an object is created at run time. The class defines the properties of the object and the methods used to control the object's behavior. *See also* **Microsoft Visual Basic Scripting Edition**.

class factory
An object that implements the IClassFactory interface, which allows it to create objects of a specific class.

class ID (CLSID)
A universally unique identifier (UUID) that identifies a COM component. Each COM component has its CLSID in the Windows registry so that it can be loaded by other applications.

class restrictions
A general term sometimes used for access control by IP address filtering and hostname filtering.

client
On a local area network or the Internet, a computer that accesses shared network resources provided by another computer, called a server. Also, an application or process that requests a service from some process or component. A client facilitates a connection to server computers, and manages and presents information retrieved from those sources. In a client/server environment, the workstation is usually the client computer. When referring to COM objects, a program that accesses or uses a service provided by another component.

client/server architecture
A model of computing whereby client applications running on a desktop or personal computer access information on remote servers or host computers. The client portion of the application is typically optimized for user interaction, whereas the server portion provides centralized, multiuser functionality.

CLSID
See **class ID**.

clustering
Connecting two or more computers together for the purpose of sharing resources and request load. Each member computer of a cluster is called a node. The nodes in a cluster may either have their own storage devices or share a common device. Typically, clustering will involve support for load balancing, fault tolerance, and failover. *See also* **load balancing; node; fault tolerance; failover**.

colocation
Installing and maintaining a computer at an Internet Service Provider (ISP) that belongs to another company or group. For example, a company might colocate one of their servers at an ISP to save costs, or to make large-scale upgrades easier.

COM
See **Component Object Model**.

commit
The phase in a transaction when all interactions are finalized and the persistent state of the underlying database is changed.

Common Gateway Interface (CGI)
A server-side interface for initiating software services. The specification that defines communications between information services (such as an HTTP service) and resources on the server's host computer, such as databases and other programs. For example, when a user submits a form through a Web browser, the HTTP service executes a program (often called a CGI script) and passes the user's input information to that program through CGI. The program then returns information to the service through CGI. Any software can be a CGI program if it handles input and output according to the CGI standard. CGI applications always run out-of-process. *See also* **server**.

Common Gateway Interface (CGI) bin directory
The directory on a server where CGI script programs are stored. Commonly called CGI-bin or CGI-scripts.

Common Gateway Interface (CGI) script
A program that allows a server to communicate with users on the Internet. For example, when a user enters information in a form on a Web page, a CGI script interprets the information and communicates it to a database program on the server.

Common Object Request Broker Architecture (CORBA)
A specification developed by the Object Management Group in 1992 in which pieces of programs (objects) communicate with other objects in other programs, even if the two programs are written in different programming languages and are running on different platforms. A program makes its request for objects through an object request broker, or ORB, and thus does not need to know the structure of the program from where the object comes. CORBA is designed to work in object-oriented environments.

communications protocol
A set of rules or standards designed to enable computers to connect with one another and to exchange information with as few errors as possible. Some communications protocols contain other protocols, such as hardware protocols and file transfer protocols. Examples include Hypertext Transfer Protocol, Transmission Control Protocol/Internet Protocol (TCP/IP), and Systems Network Architecture (SNA).

compile time
The time during which a program is translated from source language into machine language.

Component Object Model
The object-oriented programming model that defines how objects interact within a single application or between applications. In COM, client software accesses an object through a pointer to an interface—a related set of functions called methods—on the object.

Component Object Model (COM) component
A binary file containing code for one or more class factories, COM classes, registry-entry mechanisms, loading code, and so on. *See also* **Component Object Model; distributed Component Object Model**.

concurrency
The appearance of simultaneous execution of processes or transactions by interleaving the execution of multiple pieces of work.

connected user
A user who is currently accessing one of the services of a Web server.

connection pooling
A performance optimization based on using collections of pre-allocated resources, such as objects or database connections. Pooling results in more efficient resource allocation.

content type
The type of file (such as text, graphic, or sound), usually indicated by the file name extension (such as .txt, .gif, or .wav, respectively).

control
In a graphical user interface (GUI), an object on the screen that can be manipulated by a user to perform an action. Perhaps the most common controls are buttons that a user can click to select an option, and scroll bars that a user employs to move through a document or position text in a window.

cookies
A means by which, under the HTTP protocol, a server or a script can maintain information on the client computer. Cookies are small text files which are stored in the user's browser by the Web server. Cookies contain information about the user such as an identification number, a password, how a user shopped on a Web site, or how many times the user visited that site. A Web site can access cookie information whenever the user connects to the server.

CORBA
See **Common Object Request Broker Architecture**.

crawler
See **spider**.

CryptoAPI
See **Microsoft Cryptographic API**.

cryptography
A field science involving the transmission of information in an encoded form so that only an intended recipient can decode the information and reveal its meaning. Encoded information is commonly said to be encrypted.

cursor
An onscreen indicator, such as a blinking underline or rectangle, that marks the place at which a keystroke will appear when typed. In applications and operating systems that use a mouse, the arrow or other onscreen icon that moves with movements of the mouse. Also, a piece of software that returns rows of data to the application. A cursor on a resultset indicates the current position in the resultset.

cycle
In logging, to close an existing log file and start a new one.

D

daemon
A networking program that performs a housekeeping or maintenance utility function without being called by the user. A daemon sits in the background and is activated only when needed, for example, to correct an error from which another program cannot recover.

Data Encryption Standard (DES)
A specification for encryption of computer data developed by IBM and adopted by the U.S. government as a standard in 1976. DES uses a 56-bit key to protect against password discovery and playback.

datagram
A self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between the source and destination computer and the transporting network. *See also* **frame; packet**.

data provider
Software that implements Object Linking and Embedding Database (OLE DB) methods and interfaces.

data source
The name that applications use to request a connection to an Open Database Connectivity (ODBC) data source. It specifies the computer name and (optionally) database that the data source name (DSN) maps to. A system data source is a data source that is available to anyone using the computer. Data sources that will be used with a Web server need to be system data sources.

Data Source Name (DSN)
The logical name used by Open Database Connectivity (ODBC) to refer to the drive and other information required to access data. The name is used by Internet Information Services for a connection to an ODBC data source, such as a SQL Server database.

data source tier
A logical layer that represents a computer running a Database Management System (DBMS), such as a SQL Server database. *See also* **client tier; middle tier**.

DAV
See **Distributed Authoring and Versioning**.

DCOM
See **distributed COM**.

deadlock
In operating systems or databases, a situation in which two or more processes are prevented from continuing while each waits for resources to be freed by the continuation of the other.

debugger
A software tool used to detect the source of program or script errors, by performing step-by-step execution of application code and viewing the content of code variables.

default document
Sometimes called a default home page. The file sent by a Web server when it receives a request for a URL that does not specify a file name. This document can be generated automatically by the server, or it can be a custom file placed in that directory by the administrator.

default gateway
In TCP/IP, the intermediate network device on the local network that has knowledge of the network IDs of the other networks in the Internet, so it can forward the packets to other gateways until they are delivered to the one connected to the specified destination.

DES
See **Data Encryption Standard**.

design time
The time during which a user builds an application in the development environment by adding controls, setting control or form properties, and so on. In contrast, during run time, a user interacts with the application.

Design-time ActiveX Controls
Visual authoring components that help a developer construct dynamic Web applications by automatically generating standard HTML and scripting code. They are analogous to wizards. Design-time ActiveX Controls exist at design time, and not at run time.

developer isolation
A feature of some software that provides a way for an application developer to create and test scripts without a connecting to a Web server.

DHCP
See **Dynamic Host Configuration Protocol**.

DHTML
See **Dynamic HTML**.

dial-up
Of, pertaining to, or being a connection that uses the public switched telephone network rather than a dedicated circuit or some other type of private network. Also called a slow link.

Network and Dial-Up Connections
A component of the Windows operating system that makes it possible for users to connect to remote networks such as the Internet or a private network.

Digest authentication
An authentication method that sends user name and password information over the network as a hash value. *See also* **authentication; hash value; hash value comparison**.

digital signature
The part of a digital certificate that contains an encryption key that uniquely identifies the holder of the certificate. *See also* **client certificate; key pair**.

directive
An instruction to the Active Server Pages (ASP) script engine that specifies properties, such as script language, for the selection of a script.

directory browsing
A feature that automatically provides a default Web page of available directories and files to browsers that submit a URL that does not specify a particular file.

directory replication
The copying of a master set of directories from a server (called an export server) to specified servers or workstations (called import computers) in the same or other domains. Replication simplifies the task of maintaining identical sets of directories and files on multiple computers, because only a single master copy of the data must be maintained. Files are replicated when they are added to an exported directory, and every time a change is saved to the file.

directory service
Middleware that locates the correct and full network address from a partial name or address typed into a dialog box. *See also* **middleware**.

disconnected recordset
A recordset in a client cache that no longer has a live connection to the server. If something must be done with the original data source, such as updating data, the connection will need to be re-established.

discovery mechanism
A way of finding other servers on the network.

Web Distributed Authoring and Versioning (WebDAV)
An extension to the HTTP 1.1 standard that facilitates access to files and directories through an HTTP connection. Remote authors can add, search, delete, or change directories and documents and their properties.

distributed COM
A wire protocol that enables software components to communicate directly over a network.

Distributed interNet Application Architecture (DNA)
Microsoft's architecture for Web applications.

distributed processing
A form of information processing in which work is performed by separate computers linked through a communications network. Distributed processing is usually categorized as either plain distributed processing or true distributed processing. Plain distributed processing shares the workload among computers that can communicate with one another. True distributed processing has separate computers perform different tasks in such a way that their combined work can contribute to a larger goal. The latter type of processing requires a highly structured environment that allows hardware and software to communicate, share resources, and exchange information freely.

DLL
See **dynamic-link library**.

DNS
See **Domain Name System**.

domain
In Windows, a collection of computers that share a common domain database and security policy. Each domain has a unique name. *See also* **domain, Internet**.

domain controller
For a Windows 2000 Server domain, the server that authenticates domain logons and maintains the security policy and the master database for a domain.

domain, Internet
The highest subdivision of a domain name in a network address, which identifies the type of entity owning the address (for example, .com for commercial users or .edu for educational institutions) or the geographical location of the address (for example, .fr for France or .sg for Singapore). The domain is the last part of the address (for example, www.microsoft.com).

domain name
An address of a network connection that identifies the owner of that address in a hierarchical format. For example, www.whitehouse.gov identifies the Web server at the White House, which is a government agency. *See also* **Domain Name System**.

Domain Name System (DNS)
The system by which hosts on the Internet have domain name addresses (such as microsoft.com) and IP addresses (such as 172.21.13.45). The domain name address is used by human users and is automatically translated into the numerical IP address, which is used by the packet-routing software. DNS is also the acronym for Domain Name Service, the Internet utility that implements the Domain Name System. DNS servers, also called name servers, maintain databases containing the addresses and are accessed transparently by the user.

Domain Name System (DNS) reverse lookup
Finding the IP address that corresponds to a domain name.

Domain Name System (DNS) spoofing
Assuming the DNS name of another system by either corrupting a name-service cache, or by compromising a domain-name server for a valid domain.

download
In communications, the process of transferring a copy of a file from a remote computer to the requesting computer by means of a modem or network.

DSN
See **Data Source Name**.

DWORD
The Win32 API designation for a 32-bit integer.

dynamic binding
Binding (converting symbolic addresses in the program to storage-related addresses) that occurs during program execution. The term often refers to object-oriented applications that determine, during run time, which software routines to call for particular data objects. Also called late binding.

Dynamic Host Configuration Protocol (DHCP)
A TCP/IP protocol that enables a network connected to the Internet to assign a temporary IP address to a host automatically when the host connects to the network.

dynamic HTML (DHTML)
A set of innovative features in Internet Explorer version 4.0 and later that can be used to create HTML documents that dynamically change their content and interact with the user. By using DHTML, authors can provide special effects on a Web page without relying on server-side programs.

dynamic-link library (DLL)
A feature of the Microsoft Windows family of operating systems that supports executable routines—usually serving a specific function or set of functions—to be stored separately as files with the file extension name .dll, and to be loaded only when called by the program that needs them. This saves memory during program execution and enables code reusability.

dynamic page
An HTML document that contains animated GIFs, Java applets, ActiveX Controls, or DHTML. Also, a Web page created automatically based on information provided by the user, or generated "on the fly" with ASP.

E

early binding
See **static binding**.

e-commerce
Electronic commerce. The process of buying and selling over the Web—often based on software products such as Microsoft Commerce Server.

e-mail
A system whereby a computer user can exchange messages with other computer users (or groups of users) through a communications network. E-mail is one of the most popular uses of the Internet.

encapsulate
To treat a collection of structured information as a whole without affecting or taking notice of its internal structure. In communications, a message or packet constructed according to a protocol such as a TCP/IP packet, may be taken with its formatting data as an undifferentiated stream of bits that is then broken up and packaged according to a lower-level protocol (for example, as ATM packets) to be sent over a particular network; at the destination, the lower-level packets are assembled, re-creating the message as formatted for the encapsulated protocol.

encryption
A way of making data indecipherable to protect it from unauthorized viewing or use, especially during network transmission or when it is stored on a transportable magnetic medium while it is being sent from computer to computer. Encryption can be either symmetric or asymmetric. Symmetric encryption involves the use of the same key to both encrypt and decode the data. Asymmetric encryption uses one key to encrypt and another to decode. *See also* **key pair**.

Ethernet
A 10-Mb/s standard for local area networks (LANs) initially developed by Xerox and later refined by Digital, Intel, and Xerox (DIX). All hosts are connected to a coaxial cable where they contend for network access using a CSMA/CD paradigm.

event
Any action, often generated by a user or an ActiveX control, to which a program might respond. Typical events include pressing a keyboard key, choosing a button by using a mouse click, and other mouse actions. Programmers write code to respond to these actions.

event method
A procedure that is invoked only by a particular event, such as On_Click.

exception
In programming, a problem or change in conditions that causes the microprocessor to stop what it is doing and handle the situation in a separate routine. An exception is similar to an interrupt; both refer the microprocessor to a separate set of instructions.

executable program
A program, or collection of programs, forms, data, menus, and other files, that can be run.

expires header
An expiration date or time for a file sent by a server; the expiration information is used by proxy servers and browser caches.

extended partition
Created from free space on a hard disk, an extended partition can be subpartitioned into zero or more logical drives. Only one of the four partitions allowed per physical disk can be an extended partition, and no primary partition needs to be present to create an extended partition. *See also* **logical drive**.

eXtensible Markup Language (XML)
A data format for structured document interchange on the Web. It is called the "extensible markup language" because it is not a fixed format like HTML. XML is designed to enable the use of SGML on the World Wide Web. XML is not a single markup language: It is a metalanguage that allows an author to design a markup language. A regular markup language defines a way to describe information in a certain class of documents (for example, HTML). With XML, authors can define their own customized markup language for many classes of documents.

eXtensible Stylesheet Language (XSL)
A stylesheet mechanism that can be used to specify how to transform XML documents into displayable structures. Although XSL defines a grammar of advanced formatting characteristics, it also can be used to generate displayable HTML, or other well-formed markup languages.

extension control block
A data structure created and used by IIS to communicate with an ISAPI extension.

extranet
An extension of a corporate intranet using World Wide Web technology to facilitate communication with the corporation's suppliers and customers. An extranet allows customers and suppliers to gain limited access to a company's intranet in order to enhance the speed of communications and the efficiency of business relationships.

F

failback
When the failed server node is fully restored to action.

failover
When one individual computer fails, another automatically takes over its request load. The transition is invisible to the user.

FAQ
See **Frequently Asked Questions**.

fat server
In a client/server architecture, a server computer that performs most of the processing, with little or none performed by the client.

fault tolerance
The ability of a computer or an operating system to respond to a catastrophic event or fault, such as a power outage or a hardware failure, in a way that ensures that no data is lost or corrupted. This can be accomplished with a battery-backed power supply, backup hardware, provisions in the operating system, or any combination of these. In a fault-tolerant network, the system has the ability either to continue the system's operation without loss of data; or to shut the system down and restart it, recovering all processing that was in progress when the fault occurred. *See also* **replication; failover**.

file allocation table (FAT) file system
The system used by Mirosoft® MS-DOS to organize and manage files. The FAT is a data structure that MS-DOS creates on the disk when the disk is formatted. When MS-DOS stores a file on a formatted disk, the operating system places information about the stored file in the FAT so that MS-DOS can retrieve the file later when requested. The FAT is the only file system MS-DOS can use. *See also* **NTFS**.

file name extension mapping
Connecting all files with a certain file name extension to a program. For example, by a default setting in Windows Explorer, all .txt files are associated with Notepad.

file space
A term sometimes used for the file-directory tree of a server.

File Transfer Protocol (FTP)
The protocol used for copying files to and from remote computer systems on a network using Transmission Control Protocol/Internet Protocol (TCP/IP), such as the Internet. This protocol also allows users to use FTP commands to work with files, such as listing files and directories on the remote system.

filter
In IIS, a feature of ISAPI that allows pre-processing of requests and post-processing of responses, permitting site-specific handling of HTTP requests and responses.

filtering, host name
Allowing or denying access based on the host name from which the browser is attempting access.

filtering, IP address
Allowing or denying access based on the IP address from which the browser is attempting access.

finger
An Internet utility that enables a user to obtain information on other users who may be at other sites (if those sites permit access by finger). Given an e-mail address, finger returns the user's full name, an indication of whether or not the user is currently logged on, and any information other users have chosen to supply as a profile. Given a first or last name, finger returns the logon names of users whose first names match. It can also show the last time the user logged on, idle time, terminal line, and terminal location (where applicable), and even project files left by the user.

firewall
A security system intended to protect an organization's network against external threats, such as intruders, coming from another network such as the Internet. A firewall prevents computers in the organization's network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the organization's network, and the proxy server decides whether it is safe to let a particular message or file pass through. *See also* **proxy server**.

footer
In Web publishing, a short addition to every Web page sent out by the server. *See* **server-side include**.

form
In Web publishing, a Web page or portion of a Web page that is filled out by the user and sent back to the server for processing.

Fortezza
The U.S. government security standard that satisfies the Defense Messaging System security architecture with a cryptographic mechanism that provides message confidentiality, integrity, authentication, and access control to messages, components, and systems. These features can be implemented both with server and browser software and with Personal Computer Memory Card International Association (PCMCIA) hardware.

frame
In asynchronous serial communications, a unit of transmission that is sometimes measured in elapsed time. It begins with the start bit that precedes a character and ends with the last stop bit that follows the character. In synchronous communications, a package of information transmitted as a single unit. Every frame follows the same basic organization and contains control information, such as synchronizing characters, station address, and an error-checking value, as well as a variable amount of data. *See also* **datagram; encapsulation; packet**.

Frequently Asked Questions (FAQ)
Usually a document containing questions and answers that address basic questions. A visitor can find an FAQ on many Web sites. An FAQ serves to introduce a visitor to the topic or subject of the Web site and to offer general guidelines about how to best use the site.

friendly name
Also called a host name. A name that substitutes for an IP address, for example, www.microsoft.com instead of 172.16.255.255.

FrontPage Server Extensions
A group of files installed on an HTTP service to give that service the ability to provide special Microsoft FrontPage functionality. With FrontPage Server Extensions, administrators can view and manage a Web site in a graphical interface. Also, authors can create, edit, and post Web pages to IIS remotely.

FTP
See **File Transfer Protocol**.

G

gateway
A device that connects networks using different communications protocols so that information can be passed from one to the other. A gateway both transfers information and converts it to a form compatible with the protocols used by the receiving network.

GIF
See **Graphics Interchange Format**.

Global.asa
A file that stores information about an IIS application such as initialization in structures, and objects that have been given application scope.

globally unique identifier (GUID)
In COM, a 16-byte code that identifies an interface to an object across all computers and networks. Such an identifier is unique because it contains a time stamp and a code based on the network address hard-wired on the host computer's LAN interface card. These identifiers are generated by a utility program.

Gopher
An early Internet protocol and software program designed to search for, retrieve, and display text documents from remote computers or sites.

graphical user interface (GUI)
A type of environment that represents programs, files, and options by means of icons, menus, and dialog boxes on the screen. The user can select and activate these options by pointing and clicking with a mouse or, often, with a keyboard.

Graphics Interchange Format (GIF)
A computer graphics file format developed in the mid-1980s by CompuServe for use in photo-quality graphic image display on computer screens. Now commonly used on the Internet.

GUI
See **graphical user interface**.

GUID
See **globally unique identifier**.

H

handshake
A series of signals acknowledging that communication or the transfer of information can take place between computers or other devices. A hardware handshake is an exchange of signals over specific wires (other than the data wires), in which each device indicates its readiness to send or receive data. A software handshake consists of signals transmitted over the same wires used to transfer data, as in modem-to-modem communications over telephone lines.

hash value
A small amount of binary data, typically around 160 bits, derived from a message by using a hashing algorithm. The hashing procedure is one-way. There is no feasible way of deriving the original message, or even any of its properties, from the hash value, even given the hashing algorithm. The same message will always produce the same hash value when passed through the same hashing algorithm. Messages differing by even one character can produce very different hash values.

hash value comparison
When a client or server receives a hash value as part of an authentication scheme it will use a commonly known key value, such as a password, to create a hash value and compare the generated hash value with the one it received. If they are identical, authentication is accepted. *See also* **replication**.

heap (Windows heap)
An area of working memory provided by Windows that applications can use to store data.

hit
A successful retrieval of data from a cache rather than from the slower hard disk or RAM; a successful retrieval of a record matching a query in a database; or the retrieval of a document, such as a home page, from a Web site. *See also* **usage data**.

home directory
The root directory for a Web site, where the content files are stored. Also called a document root or Web root. In Internet Information Services, the home directory and all its subdirectories are available to users by default. Also the root directory for an IIS service. Typically the home directory for a site contains the home page. *See also* **home page**.

home page
The initial page of information for a collection of pages, a Web site or section of a Web site. *See also* **default document**.

host
The main computer in a system of computers or terminals connected by communications links.

host name
The name of a specific server on a specific network within the Internet, leftmost in the complete host specifications. For example, www.microsoft.com indicates the server called "www" within the network at the Microsoft Corporation.

HTML
See **Hypertext Markup Language**.

HTTP
See **Hypertext Transfer Protocol**.

HTTPD
HTTP Daemon; a Web server.

HTTP header
An informational listing at the top of an HTTP request or response.

hyperlink
A connection between an element in a hyptertext document, such as a word, phrase, symbol, or image, and a different element in the document, another hypertext document, a file, or a script. The user activates the link by clicking on the linked element, which is usually underlined or in a color different from the rest of the document. Hyperlinks are indicated in a hypertext document by the use of tags in markup languages such as SGML and HTML. These tags are generally not visible to the user. Also called hot links and hypertext links.

hypertext
Text linked together in a complex, nonsequential web of associations in which the user can browse through related topics. The term hypertext was coined in 1965 to describe documents presented by a computer that express the nonlinear structure of ideas as opposed to the linear format of books, film, and speech.

Hypertext Markup Language (HTML)
A simple markup language used to create hypertext documents that are portable from one platform to another. HTML files are simple ASCII text files with codes embedded (indicated by markup tags) to indicate formatting and hypertext links. The formatting language used for documents on the World Wide Web. *See also* **Dynamic Hypertext Markup Language; eXtensible Markup Language; Standard Generalized Markup Language**.

Hypertext Transfer Protocol (HTTP)
The client/server protocol used to access information on the World Wide Web.

I

ICMP
See **Internet Control Message Protocol**.

identities, multiple
A term sometimes used for multiple Web sites hosted on one computer; also called virtual servers. *See also* **Web site**.

IETF
See **Internet Engineering Task Force**.

IIS Admin Base Object
A DCOM object that implements the IMSAdminBase interface, using methods that enable a Web application to manipulate IIS configuration keys and data in the memory-resident metabase.

image map
An image that contains more than one hyperlink on a Web page. Clicking on different parts of the image links the user to other resources on another part of the Web page, a different Web page, or a file. Often an image map, which can be a photograph, drawing, or a composite of several different drawings or photographs, is used as a map to the resources found on a particular Web site. Image maps are created with CGI scripts. Image maps can be server-side or client-side. Server-side image maps map each URL on the server. Client-side image maps, on the other hand, do not require mediating server-side scripts because the URL mapping is contained in an HTML file. *See also* **Common Gateway Interface (CGI) script; hyperlink**.

index file
See **default document**.

inheritance
Generally, the ability of a newly-created object to automatically have, or inherit, properties of an existing object. For example, a newly created child directory can inherit the access-control settings of the parent directory.

in-process component
A component that runs in a client's process space. This component is typically a dynamic-link library (DLL).

instance
An object of a particular component class. Each instance has its own private data elements or member variables. Component instance is synonymous with object.

instantiate
To create an instance of an object.

Integrated Services Digital Network (ISDN)
Combines voice and digital network services in a single medium, making it possible to offer telephone customers digital data service and voice connection through a single "wire." A dial-up ISDN line can offer speeds of up to 128,000 bps. A type of phone line used to enhance wide area network (WAN) speeds, an ISDN line can transmit at speeds of 64 kilobits or 128 kilobits per second. An ISDN line must be installed by the phone company at both the server site and the remote site.

integrated Windows authentication
A method of authentication in which a server verifies user account information by means of a cryptographic exchange; actual passwords are never transmitted. Formerly known as NTLM and Challenge/Response authentication.

interface
A group of logically related operations or methods that provides access to a component object.

internal Web
An intranet; also sometimes called an internal network, private network, local area network (LAN), or wide area network (WAN). *See also* **intranet; local area network; wide area network**.

International Organization for Standardization (ISO)
A voluntary, nontreaty organization founded in 1946 which is responsible for creating international standards in many areas, including computers and communications. Its members are the national standards organizations of the 89 member countries, including ANSI for the U.S. *See also* **American National Standards Institute**.

International Organization for Standardization Open Systems Interconnection model (ISO/OSI model)
A layered architecture (plan) that standardizes levels of service and types of interaction for computers exchanging information through a communications network. The ISO/OSI model separates computer-to-computer communications into seven layers, or levels, each building upon the standards contained in the levels below it. The lowest of the seven layers deals solely with hardware links; the highest deals with software interactions at the application-program level.

Internet
Abbreviation for internetwork. A set of dissimilar computer networks joined together by means of gateways that handle data transfer and the conversion of messages from the sending network to the protocols used by the receiving networks. These networks and gateways use the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols. Originally part of the Defense Advanced Research Projects Agency (DARPA), operated by the U.S. Department of Defense.

Internet Control Message Protocol (ICMP)
An extension to Internet Protocol (IP), ICMP allows for the generation of error messages, test packets, and informational messages related to IP. *See also* **Packet INternet Groper (PING)**.

Internet Engineering Task Force (IETF)
A protocol engineering and development organization focused on the Internet. The IETF is a large, open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is now under the auspices of the Internet Society, a non-governmental international organization for global cooperation and coordination for the Internet and its internetworking technologies and applications. For more information, see http://www.isoc.org/.

IIS Admin Objects
A set of methods provided by IIS that allow applications to access and modify configuration settings in the metabase.

Internet Network Information Center (InterNIC)
A coordinator for DNS registration of names in the .com, .net, .org, .edu, .gov, and .mil top-level domains. To register domain names and obtain IP addresses, contact InterNIC at http://internic.net/.

Internet Protocol (IP)
The part of Transmission Control Protocol/Internet Protocol (TCP/IP) that routes messages from one Internet location to another. IP is responsible for addressing and sending Transmission Control Protocol (TCP) packets over the network. IP provides a best-effort, connectionless delivery system that does not guarantee that packets arrive at their destination or that they are received in the sequence in which they were sent. *See also* **packet**.

Internet Protocol address (IP address)
A unique address that identifies a host on a network. It identifies a computer as a 32-bit address that is unique across a Transmission Control Protocol/Internet Protocol (TCP/IP) network. An IP address is usually represented in dotted-decimal notation, which depicts each octet (eight bits, or one byte) of an IP address as its decimal value and separates each octet with a period. For example: 172.16.255.255.

Internet Server Application Program Interface (ISAPI)
An application program interface that resides on a server computer for initiating software services tuned for the Microsoft Windows operating system. It is an API for developing extensions to IIS and other HTTP services that support the ISAPI interface. *See also* **Application Programming Interface; Common Gateway Interface**.

Internet service
Any protocol for transferring information over the Internet, except HTTP. The protocol is the first part of the full URL for a resource. Internet service types include Gopher, telnet, WAIS, NNTP, HTTP, and FTP. *See also* **Hypertext Transfer Protocol; protocol**.

Internet service provider (ISP)
Public provider of remote connections to the Internet.
A company or educational institution that enables remote users to access the Internet by providing dial-up connections or installing leased lines.

interoperability
The ability of software and hardware on multiple computers from multiple vendors to communicate meaningfully.

intranet
A network designed for information processing within a company or organization. Its uses include such services as document distribution, software distribution, access to databases, and training. An intranet is so called because it usually employs applications associated with the Internet, such as Web pages, Web browsers, FTP sites, e-mail, newsgroups, and mailing lists, in this case accessible only to those within the company or organization.

IP
See **Internet Protocol**.

ISAPI
See **Internet Server Application Programming Interface (ISAPI)**.

ISDN
See **Integrated Services Digital Network**.

ISO/OSI model
See **International Organization for Standardization Open Systems Interconnection model**.

ISP
See **Internet service provider**.

J

Java
An object-oriented programming language developed by Sun Microsystems, Inc. Currently, the most widespread use of Java is in programming small applications, or applets, for the World Wide Web.

JavaBeans
An object model being developed by SunSoft that is targeted to interoperate with a variety of other object models, including COM and CORBA. *See also* **Common Object Request Broker Architecture; Component Object Model**.

Java Database Connectivity (JDBC)
Data access interfaces based on ODBC for use with the Java language.

JavaScript
A scripting language developed by Netscape Communications that is syntactically similar to Java. JavaScript, however, is not a true object-oriented language, and it is limited in performance compared with Java because it is not compiled. A JavaScript-client Web browser is necessary to run JavaScript code. Now an open standard known as the ECMA 262 language specification. *See also* **JScript**.

Java virtual machine
Software on a computer that runs Java applets.

JDBC
See **Java Database Connectivity**.

JIT
See **just-in-time activation**.

Joint Photographic Experts Group (JPEG)
An ISO/ITO standard for storing images in compressed form using a discrete cosine transform. JPEG trades off compression against loss; it can achieve a compression ratio of 100:1 with significant loss and up to 20:1 with little noticeable loss.

JPEG
See **Joint Photographic Experts Group**.

JScript
The Microsoft open implementation of JavaScript. JScript complies with the ECMA 262 language specification.

just-in-time activation (JIT)
The ability of a COM object to be activated only as needed for executing requests from its client. Objects can be deactivated even while clients hold references to them, allowing otherwise idle server resources to be used more productively.

K

Keep-Alive connection
An HTTP connection that is not closed after an exchange is completed.

Kerberos protocol
The basis of Windows security, for both internal and intranet logon. The Kerberos protocol provides for the secure use of distributed software components. *See also* **cryptography; encryption**.

key
A node in the Windows registry or IIS metabase. A key can contain subkeys and value entries. For example: Environment is a key of HKEY_CURRENT_USER.

key pair
The combination of private and public encryption keys that provides verification of the source of data sent across a network. *See also* **certificate, client; digital signature; session key**.

keyword
In search-engine technology, a significant word, which is used for content indexing; *see also* **noise word**. In programming, a word reserved for a command or other program instruction.

keyword index
A file of significant words appearing in documents; used for keyword searches.

L

LAN
See **local area network**.

late binding
See **dynamic binding**.

LCID
See **Locale Identifier**.

LDAP
See **Lightweight Directory Access Protocol**.

Lightweight Directory Access Protocol (LDAP)
A network protocol designed to work on Transmission Control Protocol/Internet Protocol (TCP/IP) stacks to extract information from a hierarchical directory. This gives users a single tool to search through data to find a particular piece of information, such as a user name, e-mail address, security certificate, or other contact information.

link
See **hyperlink**.

load balancing
When a server cluster shares the information requests equally over all of its active nodes. This can be done either statically, by tying clients directly to different back-end servers, or dynamically by having each client tied to a different back-end server controlled by software or a hardware device. The Network Load Balancing feature of Windows 2000 Advanced Server provides load balancing for HTTP services.

local area network (LAN)
A group of computers and other devices intended to serve an area of only a few square kilometers or less and connected by a communications link that enables any device to interact with any other on the network. Because the network is known to cover only a small area, optimizations can be made in the network signal protocols that permit data rates of up to 100 Mbps. *See also* **Ethernet; token ring; wide area network**.

Locale Identifier (LCID)
A function of Windows that specifies operating system settings based on geographical location (country).

local group
For Windows 2000 Professional, a group that can be granted permissions and rights only for its own workstation. However, it can contain user accounts from its own computer and (if the workstation participates in a domain) user accounts and global groups both from its own domain and from trusted domains.

localhost
A placeholder for the name of the computer on which a program is running; localhost uses the reserved loopback IP address 127.0.0.1.

log file
The file in which logging records are stored. This file can be either a text file or a database file.

logging
Storing information about events that occurred on a firewall or network.

logical drive
A subpartition of an extended partition on a hard disk. *See also* **extended partition**.

M

main thread
A single thread used to run all objects of components marked as "Single-threaded." *See also* **Apartment thread**.

Mail or Messaging Applications Programming Interface (MAPI)
An open and comprehensive messaging interface used by developers to create messaging and workgroup applications—such as e-mail, scheduling, calendaring, and document management. In a distributed client/server environment, MAPI provides enterprise messaging services within Windows Open Services Architecture (WOSA).

Management Information Base (MIB)
Software that describes aspects of a network that can be managed by using the Simple Network Management Protocol (SNMP). The MIB files included in Windows can be used by third-party SNMP monitors to enable SNMP monitoring of the Web and FTP services of IIS.

MAPI
See **Mail or Messaging Applications Programming Interface**.

marshaling
The process of packaging and sending interface method parameters across thread or process boundaries.

master properties
In IIS, properties that are set at the computer level that become default settings for all Web or FTP sites on that computer. *See also* **inheritance**.

MDAC
See **Microsoft Data Access Components**.

Message Digest 5 (MD5)
An encryption method used on the Internet.

message passing
A method for processes running in parallel to interact with one another.

Message Queuing
A server technology that developers can use to build large-scale distributed systems with reliable communications between applications that can continue to operate even when networked systems are unavailable.

meta-authoring environment
A term sometimes used for the process of both authoring Web pages and setting up a Web site.

metabase
A structure for storing IIS configuration settings; the metabase performs some of the same functions as the system registry, but uses less disk space.

metadata
Data used to describe other data. For example, Indexing Service must maintain data that describes the data in the content index.

method
A procedure (function) that acts on an object.

MIB
See **Management Information Base**.

Microsoft Cryptographic API
An application programming interface providing services for authentication, encoding, and encryption in Win32-based applications.

Microsoft Data Access Components (MDAC)
Consists of ActiveX Data Objects (ADO), the Remote Data Service (RDS), Microsoft OLE DB Provider for ODBC, Open Database Connectivity (ODBC), ODBC drivers for Microsoft SQL Server, Microsoft Access and other desktop databases, as well as Oracle databases.

Microsoft Visual Basic for Applications (VBA)
The development environment and language found in Visual Basic that can be hosted by applications.

Microsoft Visual Basic Scripting Edition (VBScript)
A subset of the Microsoft Visual Basic language, VBScript is implemented as a fast, portable, lightweight interpreter for use in World Wide Web browsers and other applications that use ActiveX Controls and Java applets.

middle tier
Also known as application server tier. The logical layer between a user interface or Web client and the database. This is typically where the Web server resides, and where business objects are instantiated. *See also* **client tier; data source tier**.

middleware
The network-aware system software, layered between an application, the operating system, and the network transport layers, whose purpose is to facilitate some aspect of cooperative processing. Examples of middleware include directory services, message-passing mechanisms, distributed transaction processing (TP) monitors, object request brokers, remote procedure call (RPC) services, and database gateways.

mirror set
A fully redundant or shadow copy of data. Mirror sets provide an identical twin for a selected disk; all data written to the primary disk is also written to the shadow or mirror disk. The user can then have instant access to another disk with a redundant copy of the information on the failed disk. Mirror sets provide fault tolerance. *See also* **fault tolerance**.

modem
Modulator/demodulator. A communications device that enables a computer to transmit information over a standard telephone line.

multihomed host
A host which has a connection to more than one physical network. The host may send and receive data over any of the links but will not route traffic for other nodes. *See also* **host; router**.

Multipurpose Internet Mail Extensions mapping (MIME mapping)
A way of configuring browsers to view files that are in multiple formats. An extension of the Internet mail protocol that enables sending 8-bit based e-mail messages, which are used to support extended character sets, voice mail, facsimile images, and so on.

multithreading
Running several processes in rapid sequence within a single program, regardless of which logical method of multitasking is being used by the operating system. Because the user's sense of time is much slower than the processing speed of a computer, multitasking appears to be simultaneous, even though only one task at a time can use a computer processing cycle.

multitier architecture
Also known as three-tier architecture, multitier architecture is a technique for building applications generally split into user, business, and data services tiers. These applications are built of component services that are based on an object model such as COM. *See also* **three-tier architecture**.

N

name resolution
The method of mapping friendly names to IP addresses. *See also* **friendly name**.

natural language query
A query to a database system that is composed in a subset of natural language, such as English or Japanese. The query must conform to some restrictive syntax rules in order to be parsed.

Network News Transfer Protocol (NNTP)
The protocol used to distribute network news messages to NNTP servers and to NNTP clients (news readers) on the Internet. NNTP provides for the distribution, inquiry, retrieval, and posting of news articles by using a reliable stream-based transmission of news on the Internet. NNTP is designed so that news articles are stored on a server in a central database, thus users can select specific items to read. Indexing, cross-referencing, and expiration of aged messages are also provided. Defined in RFC 977.

network sniffer
A hardware and software diagnostic tool that can also be used to decipher passwords, which may result in unauthorized access to network accounts. Clear-text passwords are susceptible to network sniffers.

NNTP
See **Network News Transfer Protocol**.

node
A computer that is attached to a network; also called a host. A node is also a junction of some kind. On a local area network, a node is a device that is connected to the network and is capable of communicating with other network devices.

noise word
An insignificant word, such as *the*, *and*, or *be* which is ignored during indexing; also called an ignored word.

NTFS
A file system designed for use specifically with the Windows operating system. It supports long file names, full security access control, file system recovery, extremely large storage media, and various features for the Windows POSIX subsystem. It also supports object-oriented applications by treating all files as objects with user-defined and system-defined attributes. *See also* **file allocation table (FAT) file system**.

O

object
In object-oriented programming, a variable comprising both routines and data that is treated as a discrete entity. An object is based on a specific model, where a client using an object's services gains access to the object's data through an interface consisting of a set of methods or related functions. The client can then call these methods to perform operations.

object-cache scavenger
The code that periodically scans the cache for objects to be discarded. It deletes from the cache files that have not been used recently and therefore are unlikely to be used again in the near future.

Object Linking and Embedding (OLE)
A set of integration standards to transfer and share information among client applications. A protocol that enables creation of compound documents with embedded links to applications so that a user does not have to switch among applications in order to make revisions. OLE is based on the Component Object Model (COM) and allows for the development of reusable objects that are interoperable across multiple applications. The technology has been broadly used in business, where spreadsheets, word processors, financial packages, and other applications can share and link disparate information across client/server architectures.

Object Linking and Embedding Database (OLE DB)
Data-access interfaces providing consistent access to SQL and non-SQL data sources across the enterprise and the Internet. *See also* **Structured Query Language**.

object orientation
Representing the latest approach to accurately model the real world in computer applications, object orientation is an umbrella concept used to describe a suite of technologies that enable software products that are highly modular and reusable. Applications, data, networks, and computing systems are treated as objects that can be mixed and matched flexibly rather than as components of a system with built-in relationships. As a result, an application need not be tied to a specific system or data to a specific application. The four central object-oriented concepts are encapsulation, message passing, inheritance, and late binding.

Object Management Group (OMG)
A vendor alliance formed to define and promote CORBA object specifications.

Object Request Broker (ORB)
In client/server applications, an interface to which the client makes a request for an object. The ORB directs the request to the server containing the object and then returns the resulting values to the client.

octet
Eight contiguous bits, or a byte. The term was created because some computer systems attached to the Internet used a byte with more than eight bits.

ODBC
See **Open Database Connectivity**.

OLE
See **Object Linking and Embedding**.

OLE DB
See **Object Linking and Embedding Database**.

OMG
See **Object Management Group**.

Open Database Connectivity (ODBC)
An application programming interface that enables applications to access data from a variety of existing data sources. A standard specification for cross-platform database access.

Open Group, The
The parent company of a number of standards organizations including The Active Group. The Open Group now manages the core ActiveX technology, X/Open, and the Open Software Foundation (OSF).

ORB
See **Object Request Broker**.

out-of-process component
A COM component that runs in a separate process space from its client.

P

packet
A transmission unit of fixed maximum size that consists of binary information representing both data and a header containing an ID number, source and destination addresses, and error-control data. A piece of information sent over a network.

Packet INternet Groper (PING)
A command used to verify connections to one or more remote hosts. The ping utility uses the ICMP echo request and echo reply packets to determine whether a particular IP system on a network is functional. The ping utility is useful for diagnosing IP network or router failures. The term is also used as a verb. *See also* **Internet Control Message Protocol; router**.

page
See **Web page**.

parameter
A value passed in a function call.

parity
The quality of sameness or equivalence, in the case of computers usually referring to an error-checking procedure in which the number of ones must always be the same—either even or odd—for each group of bits transmitted without error. If parity is checked on a per-character basis, the method is called vertical redundancy checking, or VRC; if its checked on a block-by-block basis, the method is called longitudinal redundancy checking, or LRC. In typical modem-to-modem communications, parity is one of the parameters that must be agreed upon by sending and receiving parties before transmission can take place. *See also* **fault tolerance; stripe set; stripe sets with parity**.

partition
A portion of a physical disk that functions as though it were a physically separate unit.

password authentication
See **authentication**.

path, physical
A universal naming convention (UNC) directory path. *See also* **path, relative**.

path, relative
A UNC directory path with placeholders, or wildcards, at some levels. The term relative path is also sometimes used to mean the physical path that corresponds to a URL. *See also* **Uniform Resource Locator**.

path, URL
A term sometimes used for the full URL submitted to the server; a URL path may or may not include a specific file name. *See also* **Uniform Resource Locator**.

Perl
Practical Extraction and Report Language. An interpreted language, based on C and several UNIX utilities. Perl has powerful string-handling features for extracting information from text files. Perl can assemble a string and send it to the shell as a command; hence, it is often used for system administration tasks. A program in Perl is known as a script. Perl was devised by Larry Wall at NASA's Jet Propulsion Laboratory. *See also* **script**.

PGP
See **Pretty Good Privacy**.

physical transaction
The actual updating of the data resources that are used to record a logical transaction.

PING
See **Packet INternet Groper**.

Point-to-Point Protocol (PPP)
A set of industry-standard framing and authentication protocols included with Windows remote access to ensure interoperability with third-party remote access software. PPP negotiates configuration parameters for multiple layers of the OSI (Open Systems Interconnection) model. The Internet standard for serial communications, PPP defines how data packets are exchanged with other Internet-based systems using a modem connection.

Point-to-Point Tunneling Protocol (PPTP)
A specification for virtual private networks in which some nodes of a local area network are connected through the Internet. PPTP is an open industry standard that supports the most prevalent networking protocols—IP, IPX, and Microsoft Networking (NetBEUI). Companies can use PPTP to outsource their remote dial-up needs to an Internet service provider or other carrier to reduce cost and complexity.

policies
Conditions set by the system administrator such as how quickly account passwords expire and how many unsuccessful logon attempts are allowed before a user is locked out. These policies manage accounts to prevent exhaustive or random password attacks.

port number
A number identifying a certain Internet application. For example, the default port number for the WWW service is 80.

PPP
See **Point-to-Point Protocol**.

PPTP
See **Point-to-Point Tunneling Protocol**.

Pretty Good Privacy (PGP)
A security application that uses public-key encryption. *See also* **public-key encryption**.

process
In Windows, an object consisting of an executable program, a set of virtual memory addresses, and a thread; in UNIX, a synonym for thread. *See also* **thread**.

process accounting
A feature of IIS that allows administrators to monitor and log resource consumption of CGI scripts and out-of-process applications.

process isolation
Running an application or component out-of-process. *See also* **out-of-process component**.

program file
A file that starts an application or program. A program file has an .exe, .pif, .com, .cmd, or .bat file name extension.

programmatic security
Procedural logic provided by a component to determine if a client is authorized to perform the requested operation.

properties, document
Information about a document and its physical location on a hard disk.

properties, link
Information about an HTML document and the full URL associated with it.

protocol
The method by which computers communicate on the Internet. The most common protocol for the World Wide Web is HTTP. Other Internet protocols include FTP, Gopher, and telnet. The protocol is part of the full URL for a resource.

proxy
A software program that connects a user to a remote destination through an intermediary gateway.

proxy server
A firewall component that manages Internet traffic to and from a local area network and can provide other features, such as document caching and access control. A proxy server can improve performance by caching and directly supplying frequently requested data, such as a popular Web page, and can filter and discard requests that the owner does not consider appropriate, such as requests for unauthorized access to proprietary files. *See also* **firewall**.

public-key encryption
An asymmetric scheme that uses a pair of keys for encryption: The public key encrypts data, and a corresponding secret key decrypts it. For digital signatures, the process is reversed: The sender uses the secret key to create a unique electronic number that can be read by anyone possessing the corresponding public key, which verifies that the message is truly from the sender. *See also* **RSA; session key**.

Q

query form
An online form that the user fills out to search for information by keyword or concept; also called a search interface.

query restriction
What to look for in a search; a query restriction narrows the focus of a search. Also called a search expression or search string.

R

RAID
See **Redundant Array of Independent Disks**.

RAM
See **random access memory**.

random access memory (RAM)
Semiconductor-based memory that can be read and written by the central processing unit (CPU) or other hardware devices. The storage locations can be accessed in any order. Note that various types of ROM memory are capable of random access but cannot be written to. The term RAM is generally understood to refer to volatile memory that can be written to as well as read. Information stored in RAM is lost when the user turns off the computer.

RARP
See **Reverse Address Resolution Protocol**.

RAS
See **remote access**.

realm
A term sometimes used for domain, in this case to refer to user domains established for security reasons, not Internet domains. For password-protected files, the name of the protected resource or area on the server. If the user tries to access the protected resource while browsing, the name of the realm usually appears in the dialog box that asks for a user name and password.

redirection
The process of writing to or reading from a file or device different from the one that would normally be the target or the source. Can be used to automatically send a user from an outdated URL to a new one.

Redundant Array of Independent Disks (RAID)
A data storage method in which data, along with information used for error correction, such as parity bits, is distributed among two or more hard disk drives in order to improve performance and reliability. The hard disk array is governed by array management software and a disk controller, which handles the error correction. RAID is generally used on network servers. Several defined levels of RAID offer differing trade-offs among access speed, reliability, and cost. Windows includes three of the RAID levels: Level 0, Level 1, and Level 5.

registry
A central hierarchical database in Windows used to store information necessary to configure the system for one or more users, applications, and hardware devices. The registry contains information that is constantly referenced during operation, such as profiles for each user, the applications installed on the computer; and the types of documents each can create, property sheet settings for folders and application icons, what hardware exists on the system; and which ports are being used.

remote access
A service that allows remote clients running Microsoft Windows to dial-in to a network. *See also* **dial-up**.

Remote Data Services
A Web-based technology that brings database connectivity and corporate data publishing capabilities to Internet and intranet applications.

remote procedure call (RPC)
In programming, a call by one program to a second program on a remote system. The second program usually performs a task and returns the results of that task to the first program.

replication
Copying from one server node to another of either content or the configuration metabase, or both. This copying can either be done manually or automatically by using replication software. Replication is a necessary function of clustering to ensure fault tolerance. *See also* **fault tolerance; clustering**.

Request for Comments (RFC)
The document series, begun in 1969, which describes the Internet suite of protocols and related experiments. Not all (in fact very few) RFCs describe Internet standards, but all Internet standards are written up as RFCs. The RFC series of documents is unusual in that the proposed protocols are forwarded by the Internet research and development community, acting on their own behalf, as opposed to the formally reviewed and standardized protocols that are promoted by organizations such as ANSI. *See also* **American National Standards Institute**.

resource dispenser
A service that provides the synchronization and management of nondurable resources within a process, providing for simple and efficient sharing by COM objects. For example, the ODBC resource dispenser manages pools of database connections. *See also* **Open Database Connectivity**.

resource manager
A system service that manages durable data. Server applications use resource managers to maintain the durable state of the application, such as the record of inventory on hand, pending orders, and accounts receivable. The resource managers work in cooperation with the transaction manager to provide the application with a guarantee of atomicity and isolation (using the two-phase commit protocol). Microsoft SQL Server is an example of a resource manager.

Reverse Address Resolution Protocol (RARP)
A TCP/IP protocol for determining the IP address (or logical address) of a node on a local area network connected to the Internet, when only the hardware address (or physical address) is known. Although the acronym RARP refers only to finding the IP address, and Address Resolution Protocol (ARP) technically refers to the opposite procedure, the acronym ARP is commonly used to describe both procedures.

RFC
See **Request For Comments**.

robot
An automated program such as a search engine, indexing program, or cataloging software, that requests Web pages much faster than human beings can. Other commonly used terms for robot include crawler and spider.

router
An intermediary device on a communications network that expedites message delivery. On a single network linking many computers through a mesh of possible connections, a router receives transmitted messages and forwards them to their correct destinations over the most efficient available route. On an interconnected set of local area networks (LANs) using the same communications protocols, a router serves the somewhat different function of acting as a link between LANs, enabling messages to be sent from one to another.

RPC
See **remote procedure call**.

RSA
A public-key encryption standard for Internet security. This acronym derives from the last names of the inventors of the technology: Rivest, Shamir, and Adleman.

run time
The time during which a program actually runs. *See also* **design time**.

S

scalability
The capability to use the same software environment on many classes of computers and hardware configurations. Although often associated with an evolution to large systems, larger organizations often have need for the same software service to be provided with good performance to both small and large groups of users.

scope
In programming, the extent to which an identifier, such as a constant, data type, variable, or routine, can be referenced within a program. Scope can be global or local. Scope can also be affected by redefining identifiers, such as by giving the same name to both a global variable and a local variable.

script
A kind of program that consists of a set of instructions for an application or utility program. A script can be embedded in a Web page. *See also* **ActiveX; Common Gateway Interface**.

scripting engine
A program that interprets and executes a script. *See also* **script**.

search expression
See **query restriction**.

search interface
See **query form**.

search string
See **query restriction**.

Secure Sockets Layer (SSL)
A protocol that supplies secure data communication through data encryption and decryption. SSL uses RSA public-key encryption for specific TCP/IP ports. It is intended for handling commerce payments. An alternative method is Secure-HTTP (S-HTTP), which is used to encrypt specific Web documents rather than the entire session. SSL is a general-purpose encryption standard. SSL can also be used for Web applications requiring a secure link, such as e-commerce applications, or for controlling access to Web-based subscription services.

security log
A log, generated by a firewall or other security device, that lists events that could affect security, such as access attempts or commands, and the information about the users involved.

semaphore
A locking mechanism used inside resource managers or resource dispensers. Semaphores have no symbolic names—only shared and exclusive mode access—no deadlock detection, and no automatic release or commit.

server
A term used for any of the following: a computer on a network that sends files to, or runs applications for, other computers on the network; the software that runs on the server computer and performs the work of serving files or running applications; or, in object-oriented programming, a piece of code that exchanges information with another piece of code upon request.

server certificate
A unique digital identification that forms the basis of a Web server's SSL security features. Server certificates are obtained from a mutually trusted, third-party organization, and provide a way for users to authenticate the identity of a Web site.

server cluster
A group of server computers that are networked together both physically and with software, in order to provide cluster features such as fault tolerance or load balancing. *See also* **fault tolerance; load balancing**.

server node
An individual computer in a server cluster.

server process
A process that hosts COM components. A COM component can be loaded into a surrogate server process, either on the client computer (local) or on another computer (remote). It can also be loaded into a client application process (in-process).

server scriptlet
A COM object that is created with Microsoft Server Scriptlet technology.

server-side include
A mechanism for including dynamic text in World Wide Web documents. Server-side includes are special command codes that are recognized and interpreted by the server; their output is placed in the document body before the document is sent to the browser. Server-side includes can be used, for example, to include the date/time stamp in the text of the file.

session key
A digital key that is created by the client, encrypted, and sent to the server. This key is used to encrypt data sent by the client. *See also* **certificate; digital signature; key pair**.

SGML
See **Standard Generalized Markup Language**.

shared property
In Component Services, a variable that is available to all objects in the same server process through the Shared Property Manager. The value of the property can be any type that can be represented by a variant.

Simple Mail Transfer Protocol (SMTP)
A TCP/IP protocol for sending messages from one computer to another on a network. This protocol is used on the Internet to route e-mail.

Simple Network Management Protocol (SNMP)
The network management protocol of TCP/IP. In SNMP, agents, which can be hardware as well as software, monitor the activity in the various devices on the network and report to the network console workstation. Control information about each device is maintained in a structure known as a management information block. *See also* **Management Information Base**.

single-threaded control
A model in which all objects are executed on a single thread.

sitename
See **host name**.

slow link
A modem connection, usually from 14,400 bps to 56,000 bps.

SMTP
See **Simple Mail Transfer Protocol**.

SNA
See **Systems Network Architecture**.

snap-in
Snap-ins are programs hosted within Microsoft Management Console (MMC) that administrators use to manage network services. MMC provides the environment in which management tools (snap-ins) are hosted; snap-ins provide the actual management behavior necessary to administer network services such as IIS.

sniffer
See **network sniffer**.

SNMP
See **Simple Network Management Protocol**.

socket
An identifier for a particular service on a particular node on a network. The socket consists of a node address and a port number, which identifies the service. For example, port 80 on an Internet node indicates a Web server.

spider
A fast, automated program—such as a search engine, indexing program, or cataloging software—that requests Web pages much faster than human beings can. Other commonly used terms for spider are crawler and robot.

spoofing
Impersonating another person or computer, usually by providing a false e-mail name, URL, or IP address.

SQL
See **Structured Query Language**

SQL Access Group (SAG)
A consortium of vendors established in November 1989 to accelerate the Remote Data Access standard and to deliver protocols for interconnectivity among multiple SQL-based software products.

SSL
See **Secure Sockets Layer**.

Standard Generalized Markup Language (SGML)
An ISO standard (ISO 8879:1986) which supplies a formal notation for the definition of generalized markup languages. It is an international standard for the definition of device-independent, system-independent methods of representing texts in an electronic form. SGML is a metalanguage—that is, a means of formally describing a language, in this case, a markup language. *See also* **Hypertext Markup Language; International Organization for Standardization; eXtensible Markup Language**.

stateful object
An object that holds private state accumulated from the execution of one or more client calls.

stateless object
An object that does not hold private state accumulated from the execution of one or more client calls.

static binding
Binding (converting symbolic addresses in the program to storage-related addresses) that occurs during program compilation or linkage.

static page
HTML pages prepared in advance of the request and sent to the client upon request. This page takes no special action when requested. *See also* **dynamic page**.

stripe set
Refers to the saving of data across identical partitions on different drives. A stripe set does not provide fault tolerance; however, stripe sets with parity do provide fault tolerance. *See also* **fault tolerance; partition; stripe sets with parity**.

stripe sets with parity
A method of data protection in which data is striped in large blocks across all the disks in an array. Data redundancy is provided by the parity information. This method provides fault tolerance. *See also* **fault tolerance; stripe set**.

Structured Query Language (SQL)
The international standard language for defining and accessing relational databases.

stub
A routine that contains no executable code and that generally consists of comments describing what will eventually be there; it is used as a placeholder for a routine to be written later.

subnet mask
A TCP/IP configuration parameter that extracts network and host configuration from an IP address.

System Data Source Name (DSN)
A name that can be used by any process on the computer. IIS uses system DSNs to access ODBC data sources.

Systems Network Architecture (SNA)
A widely used communications framework developed by IBM to define network functions and to establish standards for enabling its different models of computers to exchange and process data. SNA contains separate layers. As changes occur in one layer, no other layer need be changed.

T

T1
A U.S. telephone standard for a transmission facility at digital signal level 1 (DS1) with 1.544 Mbps in North America and 2.048 Mbps in Europe. The bit rate is with the equivalent bandwidth of approximately twenty-four 56 Kbps lines. A T1 circuit is capable of serving a minimum of 48 modems at 28.8 Kbps, or 96 modems at 14.4 Kbps. T1 circuits are also used for voice telephone connections. A single T1 line carries 24 telephone connections with 24 telephone numbers. When used for voice transmission, a T1 connection must be split into 24 separate circuits.

T3
A U.S. telephone standard for a transmission facility at digital signal level 3 (DS3). Equivalent in bandwidth to 28 T1s. The bit rate is 44.736 Mbps. T3 is sometimes called a 45-meg circuit.

TCP/IP
See **Transmission Control Protocol/Internet Protocol**.

telnet
A protocol that enables an Internet user to log onto and enter commands on a remote computer linked to the Internet, as if the user were using a text-based terminal directly attached to that computer. Telnet is part of the TCP/IP suite of protocols.

10BaseT
A variant of Ethernet which allows stations to be attached by a twisted-pair cable.

thin server
A client/server architecture in which most of an application is run on the client computer, which is called a fat client, with occasional data operations on a remote server. Such a configuration yields good client performance but complicates administrative tasks such as software upgrades. *See also* **fat server**.

thread
The basic entity to which the operating system allocates CPU time. A thread can execute any part of the application's code, including a part currently being executed by another thread. All threads of a process share the virtual address space, global variables, and operating-system resources of the process.

three-tier architecture
Divides a networked application into three logical areas: the user interface layer, the business logic layer, and the database layer. Layers may have one or more components. For example, there can be one or more user interfaces in the top tier, each user interface may communicate with more than one application in the middle tier at the same time, and the applications in the middle tier may use more than one database at a time. Components in a tier may run on a computer that is separate from the other tiers, communicating with the other components over a network.

throttling
Controlling the maximum amount of bandwidth dedicated to Internet traffic on a server. This feature is useful if there are other services (such as e-mail) sharing the server over a busy link.

thumbnail
A small version of a graphic with a hyperlink to a larger version of the same graphic.

time-out
A setting that automatically cancels an unanswered client request after a certain period of time.

token ring
A type of network with nodes wired into a ring. Each node constantly passes a control message (token) on to the next; whichever node has the token can send a message. Often, "token ring" is used to refer to the Institute of Electrical and Electronics Engineers (IEEE) 802.5 token ring standard, which is the most common type of token ring. *See also* **local area network**.

TP
See **transaction processing**.

transaction context object
An object used to allow a client to dynamically include one or more objects in one transaction.

transaction manager
A system service responsible for coordinating the outcome of transactions in order to achieve atomicity. The transaction manager ensures that the resource managers reach a consistent decision on whether the transaction should commit or fail.

transaction processing (TP)
The real-time handling of computerized business transactions as they are received by the system. Also called online transaction processing (OLTP).

Transmission Control Protocol/Internet Protocol (TCP/IP)
A communications standard for all computers on the Internet. On the sending end, TCP breaks the data to be sent into data segments. IP assembles segments into packets that contain data segments, as well as sender and destination addresses. IP then sends packets to the router for delivery. On the receiving end, IP receives the packets and breaks them down into data segments. TCP assembles the data segments into the original data set. *See also* **packet**.

tree, directory
A conceptual model used to describe the directory structure of a file directory or a Web site.

two-phase commit
A protocol that ensures that transactions that apply to more than one server are completed on all servers or none at all. Two-phase commit is coordinated by the transaction manager and supported by resource managers.

two-tier architecture
See **client/server architecture**.

type library
A binary file that describes a component's methods, properties, and data structure.

U

UNC
See **Universal Naming Convention**.

Uniform Resource Locator (URL)
A naming convention that uniquely identifies the location of a computer, directory, or file on the Internet. The URL also specifies the appropriate Internet protocol, such as HTTP or FTP. For example: http://www.microsoft.com.

universal naming convention (UNC)
The naming convention used for physical directories.

upload
In communications, the process of transferring a copy of a file from a local computer to a remote computer by means of a modem or network. With a modem-based communications link, the process generally involves instructing the remote computer to prepare to receive the file on its disk and then wait for the transmission to begin.

URL
See **Uniform Resource Locator**.

URL directory
See **virtual directory**.

URL mapping
A term sometimes used for the process of associating a URL with a physical directory. *See* **virtual directory**.

usage data
Information the administrator can use to learn how other people are accessing and using a site. Analyzing this data helps an administrator identify a site's most popular (or unpopular) areas and clarifies the most common navigational paths through the site.

Usenet
The most popular news group hierarchy on the Internet.

user type
Assigned to an identifier in the metabase, a DWORD that specifies how data is used.

V

VB
See **Microsoft Visual Basic for Applications**.

VBScript
See **Microsoft Visual Basic Scripting Edition**.

virtual directory
A directory name, used in an address, which corresponds to a physical directory on the server; sometimes called URL mapping.

virtual document
A term sometimes used for a document created automatically in response to information provided by the user; also called a dynamic document. A virtual document is created only in answer to a browser request, and is not permanently stored in a physical directory. An ASP page is an example of a virtual document.

virtual machine
Software that mimics the performance of a hardware device, such as a program that allows applications written for an Intel processor to be run on a Motorola processor.

Virtual Reality Modeling Language (VRML)
A language for coding three-dimensional HTML applications.

virtual server
Also called a Web site. A virtual computer that resides on an HTTP server but appears to the user as a separate HTTP server. Several virtual servers can reside on one computer, each capable of running its own programs and each with individualized access to input and peripheral devices. Each virtual server has its own domain name and IP address and appears to the user as an individual Web site or FTP site. Some Internet service providers use virtual servers for those clients who want to use their own domain names.

volatile objects
Typically, files that the Web site administrator updates frequently.

volume set
A combination of partitions on a physical disk that appear as one logical drive. *See also* **logical drive**.

VRML
See **Virtual Reality Modeling Language**.

W

W3C
See **World Wide Web Consortium**.

WAN
See **wide area network**.

Web application
A software program that uses HTTP for its core communication protocol and delivers Web-based information to the user in the HTML language. Also called a Web-based application.

WebDAV
See **Distributed Authoring and Versioning**.

Web page
A World Wide Web document. A Web page typically consists of an HTML file, with associated files for graphics and scripts, in a particular directory on a particular computer (and thus identifiable by a URL).

Web server
In general terms, a computer equipped with the server software that uses Internet protocols such as HTTP and FTP to respond to Web client requests on a TCP/IP network.

wide area network (WAN)
A communications network that connects geographically separated areas.

Windows Internet Name Service (WINS) server
A server that uses the WINS protocol to map Internet Protocol (IP) addresses to user-friendly names. *See also* **Domain Name System**.

Windows Open Services Architecture (WOSA)
Standards for creating cross-platform applications that utilize Windows services.

Windows Script Host (WSH)
A language-independent scripting host for ActiveX scripting engines on 32-bit Windows platforms.

worker thread
A thread that is created by a component or ISAPI extension or filter to perform asynchronous processing. Using worker threads frees up IIS I/O threads to process additional requests.

working directory
A term sometimes used to describe the directory in which the Web server software is installed.

working set
The RAM allocated to a process in the Windows operating system.

World Wide Web (WWW)
The most graphical service on the Internet, the Web also has the most sophisticated linking abilities. It is a set of services that run on top of the Internet providing a cost-effective way of publishing information, supporting collaboration and workflow, and delivering business applications to connected users all over the world. The Web is a collection of Internet host systems that make these services available on the Internet using the HTTP protocol. Web-based information is usually delivered in the form of hypertext and hypermedia using HTML.

World Wide Web Consortium (W3C)
Founded in 1994 to develop common standards for the World Wide Web, the W3C is an international industry consortium jointly hosted by the Massachusetts Institute of Technology Laboratory for Computer Science (MIT/LCS) in North America, by the Institut National de Recherche en Informatique et en Automatique (INRIA) in Europe, and by the Keio University Shonan Fujisawa Campus in Asia. Initially, the W3C was established in collaboration with CERN, where the Web originated, with support from DARPA and the European Commission. For more information, see http://www.w3.org/.

WOSA
See **Windows Open Services Architecture**.

WSH
See **Windows Script Host**.

WWW
See **World Wide Web**.

X

XML
See **eXtensible Markup Language**.

XSL
See **eXtensible Stylesheet Language**.

Index

Special Characters

<!-- #include --!> statements 514
<%@ TRANSACTION=value %> directive 572
<%= expression %> scripting delimiter 262
= (equal sign), errors 297
216-color palettes, HTML standards 579

A

AbsolutePage property, Recordset object 353
Accept-Language HTTP header 261
access
  auditing with IIS logs 524
  determining
    custom authentication 510
    IIS access control 510
    IP address access control 509
    NTFS permissions 510
    overview 508
    user access control 509
  troubleshooting 510
access configuration directives, Apache
  access.conf directives and IIS properties (table) 96–97
Access Control Entries (ACEs) 471, 472
Access databases
  See also data access
  accessing remotely (table) 500
  security 81
access permissions, setting 78–79
Access this computer from network privilege 469
access.conf Apache directives, IIS properties (table) 96–97
AccessConfig Apache server directive 86
AccessFileName Apache resource configuration directive 92
AccessFlags (IISAO) property 525
accessing data *See* data access
accessing legacy applications and data *See* legacy applications and data
AccessSSLFlags (IISAO) property 525
Account Logon audit event 479
Account Management audit event 479
ACEs (Access Control Entries) 471, 472
ACID (Atomicity, Consistency, Isolation, and Durability) transaction properties, defined 15, 359
ACTION attribute of HTML tags 286
action pages, defined 286
Active Directory
  client authentication certificates 493
  clustering features 389
Active Directory Service Interfaces (ADSI) *See* ADSI (Active Directory Service Interfaces)
Active Directory Services provider, ADSDSOObject (OLE DB provider) 331
Active Scripting standard 66
Active Scripting 14, 266
Active Server Pages (ASP) *See* ASP (Active Server Pages)
Active Server Pages counters
  Errors During Script Run Time 242
  Errors/sec 240
  Memory Allocated 242
  Request Not Found 240
  Request Queued 246
  Request Wait Time 246
  Requests Queued 238, 242
  Requests Rejected 240
  Requests/sec 240, 244
  Script Engines Cached 219
  Template Cache Hit Rate 219
  Templates Cached 219
  Transactions/sec 246
ActiveConnection property 340, 584
ActiveX controls 254, 325, 385
ActiveX Data Objects (ADO) *See* ADO (ActiveX Data Objects)
ActiveX DLLs, RDS Data Factory and custom business objects 327
ActiveX scripting engines 397
ADC (Advanced Data Connector) vs. RDS 315
ADCLaunch registry key 325
Add Counters dialog box (PerfMon) 229
Add to Chart dialog box 217

AddDescription Apache resource configuration directive 92
AddEncoding Apache resource configuration directive 92
AddIcon Apache resource configuration directive 92
Additional Document Directories (URL Prefix, Map to Directory) NES 3.5 configuration setting 104
AddLanguage Apache resource configuration directive 92
AddOnServices registry key 426
AddType Apache resource configuration directive 92
ADISAPI (Advanced Data ISAPI) 326
adLockBatchOptimistic locking 346, 347
adLockOptimistic locking 346
adLockPessimistic locking 346
adLockReadOnly locking 343, 346
administration
  administering sites remotely 404–407
  automating *See* ISP installations, automating administration
  components (sample setup) 440
  Cookie.asp (sample code) 441
  ISP installations *See* ISP installations
  monitoring and tuning servers *See* monitoring and tuning servers
  Munge.asp (sample code) 441
  Redirect.asp (sample code) 442
administration interfaces, Apache vs. IIS 83
Administration Objects *See* IIS Admin Objects
administrative networks, sample installations 456
Administrator accounts 48, 78, 501
administrators
  See also ISPs (Internet service providers)
  capacity planning *See* capacity planning
  IIS 5.0 vs. IIS 4.0, tuning ASP queuing and thread pools 219
  IIS administrative architecture 9
  IIS new features (list) 3
ADO (ActiveX Data Objects)
  accessing
    data with ASP and COM components 328
    databases 314
    information 267
    legacy VSAM and AS/400 files 544
  client-side data access 318
  cursors *See* cursors
  data partitioning with RDS 322
  database access 125
  database connections 335–342
  designing custom business objects 328
ADO (ActiveX Data Objects) *(continued)*
  IIS architecture 8
  MDAC interactions 312
  preparing databases 329–334
  Recordset objects *See* Recordsets
  stored procedures 356
  vs. ODBC, TPS (transactions per second) (table) 317
ADO components, performance 338
ADO Connection object *See* Connection object (ADO)
ADO Recordset objects, using with ADO and RDS *See* ADO (ActiveX Data Objects); RDS (Remote Data Services); Recordsets
ADO type libraries, importing constants 278
ADODB.Recordset object 322, 328
adOpenDynamic ADO cursor 343
adOpenForwardOnly ADO cursor 343
adOpenKeyset ADO cursor 343
adOpenStatic ADO cursor 343
ADOR.Recordset object 322, 328
ADSDSOObject (Active Directory Services provider) 331
ADSI (Active Directory Service Interfaces)
  administering sites remotely 405
  automating
    administration 397
    tasks with command-line scripts 10
  security-related settings (list) 524
  troubleshooting (example) 520
  using the metabase 283
ADSI 2.0, new features 4
ADSI codes, changing cache time 495
ADSI scripts (examples) 400–403
Adsutil.exe, changing permissions 400
Advanced Data Connector (ADC) vs. RDS 315
Advanced Data ISAPI (ADISAPI) 326
Agents and Search NES 3.5 configuration setting 106
agile threading models 154
Alias Apache resource configuration directive 93
aliases, Apache vs. IIS 85
AliasMatch Apache resource configuration directive 93
Allow access permission, Apache vs. IIS 84
Allow IIS to control password option
  accessing remote resources (table) 501
  Anonymous authentication 483
  file and directory security 498
  IISAO property settings 525
AllowOverride access configuration directive 96

anonymous access
NTFS/IIS directory security (table) 80
setting IIS permissions 79
Anonymous authentication
IIS configurations, accessing remote resources (table) 501
Web authentication summary (table) 495
anonymous connections
monitoring counters (table) 225
tracking with PerfMon 224
anonymous passwords, IIS controlling 483
anonymous user accounts, creating 376
Anonymous User counters, defined 226
anonymous users 74, 78
Anonymous Web authentication 481
AnonymousPasswordSync (IISAO) property 525
AnonymousUserName (IISAO) property 525
AnonymousUserPass (IISAO) property 525
Apache custom modules, migrating 98, 116
Apache HTTP Server, migrating to IIS 5.0 *See* migrating from Apache HTTP Server
application architecture, migration 113
Application Configuration dialog box 301, 302
application errors, recovering from 396
Application events, debugging 300
application gateways, security 604
Application Layer (OSI) 207, 208, 212
application logic 113, 120
application mapping, setting 115
application namespaces, defined 275
Application object (ASP)
building ASP applications 274–278
functionality 267
management 270
overview 269
restricting connections across pages 341
selecting object scope 279
application root directories 561
application roots 275
application scope 562, 581
Application Service Providers, administering installations *See* ISP installations
application tiers 113, 249
Application_OnEnd event 276
Application_OnStart event 276
applications
See also Web applications
default for home pages 562
future on Internet 252
legacy *See* legacy applications and data
applications *(continued)*
migration process management
application tests (table) 50
testing 46
NES 3.5 configuration settings and IIS properties (table) 102
pages per second performance tests (table) 149
protection features 3
replicating and configuring 74
testing with
WCAT *See* WCAT (Web Capacity Analysis Tool)
Web Application Stress Tool *See* Web Application Stress Tool
three-tiers 555
architecture
applications 113, 250
IIS 5–16
Windows DNA family of technologies 251
Archive Log NES 3.5 configuration setting 102
arrays, minimizing re-dimensioning 582
AS/400 (IBM)
legacy applications and data *See* legacy applications and data
OLE DB provider *See* OLE DB provider for AS/400 and VSAM
ASP (Active Server Pages)
accessing
form input/decoding HTML 121
unprocessed input stream 123
accessing legacy applications and data
accessing host files (example) 545–548
extending transactions with COMTI (example) 540–542
integrating transaction processing data (example) 536–540
overview 531
using OLE DB and ADO 544
analyzing connection data 201
Application object *See* Application object (ASP)
changing HTML to .asp 262
combining HTML and server-side scripts 128
database access, ASP vs. Perl Dbperl/DBI 126
formatting HTML output 129
IIS architecture 6
migrating applications 111
migrating from CGI
accessing the file system 131
analyzing applications 120
ASP scripting support 120
business logic 125

ASP (Active Server Pages) *(continued)*
migrating from CGI *(continued)*
external gateways/database logic 125
handling output 128
maintaining state 127
overview 120
processing input 121
monitoring caching scripts 186
new features (list) 4
NTFS/IIS directory security 80
parsing input 123
programmability architecture 14
queuing, IIS 5.0 vs. IIS 4.0 218
reproducing CGI services 132–135
Request object *See* Request object
requesting files from IIS 264
Response object *See* Response object (ASP)
Session object *See* Session object (ASP)
scripts
debugging *See* debugging
developing middle-tier Web applications 261–269
IIS Admin Objects architecture 9
migrating applications 111
optimizing 220
security planning 601
tuning 221
vs. browser scripts 560
vs. business objects 362
vs. CGI scripts 128
viewing errors 296
vs.
CGI 560
ISAPI/CGI/static HTML pages performance tests 148
Visual Basic code 114
Web applications performance 148
ASP applications
debugging *See* debugging
deciding to port or rewrite CGI applications 112
developing middle-tier Web applications
Application and Session events 276
application boundaries 275
ASP session IDs 272
ASP session management 270
ASP session security 274
building applications 274
cookie-based ASP sessions 270
ending ASP sessions 277
Global.asa 276, 278
overview 269
selecting object scope 279
ASP applications *(continued)*
migrating to IIS 5.0 110
porting CGI applications 114
relocating with Server.MapPath method 574
running
in out-of-process pooling 396
on second tiers 452
testing with WCAT 391
ASP best practices
additional resources 586
HTML standards 578
overview 559
project directories and files 561–564
scripting for performance 581–585
style guide for scripts in ASP (list) 565
when to use/not use 560
ASP code, extending IIS security 497
ASP collections, obtaining list of values 124
ASP components, ADO data *See* ADO (ActiveX Data Objects); Recordsets
ASP delimiters, style guide 570
.asp files
HTML 262, 560, 564
layout order, style guide 571
security 601
ASP in-process, performance (table) 149
ASP intrinsic objects 111
ASP objects 98, 267
ASP out-of-process, performance (table) 149
ASP Page Counter Component 135
ASP page includes, NTFS/IIS directory security 80
ASP Session object *See* Session object (ASP)
/ASP subdirectory of /Content directory 562
AspAllowOutOfProcComponents property, IISWebService object 283
AspAllowOutOfProcComponents property, IISWebVirtualDir object 283
ASPProcessorThreadMax metabase property 219
assets, creating inventories 589
Async input/output counters (table) 214
atomicity (ACID property), defined 15, 359
Attach to Process command 303
attacks
defending against malicious 522
preventing, Windows 2000 Server 470
security (list) 464
threat assessment *See* threat assessment planning
audit logs, caching security tokens 495
audit settings, replicating 67

auditing
access with IIS logs 524
defined 463
events, Windows 2000 Server (list) 478
security 594, 603
user logons, troubleshooting 510
AuthDBGroupFile access configuration directive 96
AuthDBMGroupFile access configuration directive 96
AuthDBMUserFile access configuration directive 96
AuthDBUserFile access configuration directive 96
authenticated access 79
authenticated connections vs. nonanonymous connections 224
authentication
accessing remote resources (table) 501
creating ISAPI authentication filters 381
hidden form fields risks 291
IIS modes
Allow IIS to control password option 483
Anonymous Web authentication 481
Basic authentication 484
Certificate Services 491
client certificate mapping 489
Digest authentication 487
FTP authentication 496
functionality 481
integrated Windows authentication 486
Web authentication summary (table) 495
Web/FTP authentication (list) 480
Kerberos v5 authentication protocol *See* Kerberos v5 authentication protocol
security 462, 592, 602
Windows 2000 Server
authentication methods 472
DACLs 471
impersonation 471
IPSec 475
LocalSystem account 470
logon and token types 468
overview 467
privileges and attack prevention 470
services and service security 470
SIDs 468
AuthFlags (IISAO) property 525
AuthName access configuration directive 96
authorization 462, 472, 593, 602
AuthType access configuration directive 96
Auto Catalog NES 3.5 configuration setting 106
availability
defined 463
security 594, 603
Windows 2000 Server 477

B

back-end networks, sample installations 456
backing up
metabase 109
source servers 63
Web/FTP site contents 109
backlog monitors, disabling 221
bandwidth
capacity planning issues
network servers 144, 156
static vs. dynamic pages 142
traffic 140
measuring capacity 206
network input/output
lengthening connection queues 216
limiting bandwidth 214
monitoring bandwidth throttling (table) 214
monitoring bandwidth 213
monitoring connections 206
monitoring file transfers 207, 211
monitoring TCP connections 207, 212
monitoring transmissions rates (tables) 207–210
optimizing network connections 215
overview 206
requirements IIS-based servers 206
bandwidth throttling
enabling 214, 430
monitoring with counters (table) 214
baseline logging 247
Basic authentication, IIS configurations
accessing remote resources (table) 501
file and directory security 498
functionality 484
Web authentication summary (table) 495
batch logon, Windows 2000 Server 469
Binary Large Object (BLOB) 345, 355
binary modes, migrating UNIX Perl scripts 117
Bind To Address NES 3.5 configuration setting 100
BindAddress Apache server directive 86
bindings, disabling 459
blank lines in files, style guide 566
Blat.exe, migrating UNIX mail scripts 134
BLOB (Binary Large Object) 345, 355
Both threading model 338

both-threaded objects 155
bottlenecks
  building three-tier Web clusters 450
  checklist 154
  firewalls 458
  lengthening connection queues 216
  memory allocation 177
  ODBC connection pooling 583
  performance testing with counters *See* counters
  preventing processor bottlenecks
    analyzing activity data 196
    analyzing connection data 200
    improving processor usage and performance (list) 204
    monitoring connections 198
    monitoring processor activity 196
    monitoring server processors 195
    monitoring threads 201
    process throttling 198
  static vs. dynamic pages 142
  stress testing *See* Web Application Stress Tool
  upgrading and adding processors 227
browser access, permissions 420
browser cache 444
Browser Capabilities component 258
browser connections 584
browser scripts vs. scripts in ASP pages 560
browsers
  capacity planning for download time 143
  security 603
  supporting non-HTTP 1.1 compliant 438
  supporting text-only, HTML standards 578
  technologies supported (list) 258
  vs. applications on Internet 252
bugs, scripts and programs 423
bugtraq (UNIX) 600
built-in ASP objects (list) 267
built-in scripts, administrative architecture 10
business logic 125, 249, 268
business objects
  custom
    RDS.DataSpace object 325, 327
    RDSServer.DataFactory object 327
  vs. scripts in ASP pages 362
bytes, available in Inetinfo working sets 185

C

Cache Control Directives NES 3.5 configuration setting 104
Cache counters
  Copy Reads/sec 189
  Data Maps/sec 190
  Fast Reads/sec 190
  MDL Read Hits % 190
  MDL Reads/sec 190
  Pin Read Hits % 190
  Pin Reads/sec 190
  Read Aheads/sec 190
Cache Manager, memory shortages 192
CacheNegotiatedDocs Apache server directive 86
caches
  comparing size and performance data 192
  content expiration (example) 443
  File System Cache *See* File System Cache
  IIS Object Cache *See* Object Cache
  security tokens 495
  upgrading L2 cache 204
CacheSize property, Recordset object 342, 345
caching
  capacity planning issues 147
  data using RDS 324
  IIS 5.0 vs. IIS 4.0 218
  scripts, monitoring in ASP 186
Cacls command (FrontPage) 421
capacity planning
  additional resources 174
  checklist 154–156
  determining installation requirements 153
  guidelines 174
  microsoft.com (example) 166–174
  Network Load Balancing 152
  overview 139
  reliability 150
  Secure Sockets Layer (SSL) 147
  server clustering 150
  server-side/client-side bandwidth 146
  traffic 140–145
  WCAT *See* WCAT (Web Capacity Analysis Tool)
  Web Application Stress Tool *See* Web Application Stress Tool
  Web applications performance 148
  Web site scenarios
    Internet commerce Web sites 163
    Internet marketing Web sites 159
    Internet transactional Web sites 161

capacity planning *(continued)*
Web site scenarios *(continued)*
intranet Web sites 157
overview 156
cascading style sheets (CSS) 256
case sensitivity, UNIX vs. Windows file systems 71
CDO (Collaboration Data Objects) 132
CDONTS object (IIS) 132
certificate authentication 602
Certificate Manager Export Wizard 492
Certificate mapping (tables) 495, 501
Certificate Services 7, 8, 491, 492
Certificate Trust Lists (CTLs) 494
certificate types (list) 491
certificates
backing up
Web servers 507
when migrating to IIS 109
installing 82
mapping client certificates 489
storage, new features 2
using CryptoAPI 506
CGI (Common Gateway Interface) applications
accessing form input/decoding HTML 121
configuring script interpreters 116
converting
to ISAPI 119
Visual Basic applications 119
deciding to port or rewrite 112
developing middle-tier Web applications 260
IIS support 110
migrating to ASP
accessing the file system 131
analyzing applications 120
ASP scripting support 120
business logic 125
external gateways/database logic 125
handling output 128
maintaining state 127
overview 120
processing input 121
migrating to IIS 5.0 110
migrating UNIX mail scripts 134
optimizing memory usage 193
performance (table) 149
porting 114
process throttling limitations 198
programmability architecture 12
reproducing
e-mail delivery using ASP 132
page counters 135
running in isolation 395
CGI (Common Gateway Interface) applications *(continued)*
stopping runaway 433
thread-related issues 204
vs.
ASP 560
ISAPI/ASP/static HTML pages
performance tests 148–149
CGI 1.1 standards, IIS support 114
CGI Directory NES 3.5 configuration setting 102
CGI File Type NES 3.5 configuration setting 102
CGI scripts
causing ISAPI error 111
NTFS/IIS directory security 80
optimizing, converting to ASP/ISAPI scripts 220
vs. scripts in ASP pages 128
Cgi.pm (Perl) utility 121, 128
CGI-bin, security 84
Cgi-lib.pl (Perl) utility, input processing 121
Challenge/Response authentication
Digest authentication 487
integrated Windows authentication 486
Change (RWXD) directory 499
characters
illegal, UNIX vs. Windows file systems 71
special, migrating UNIX Perl scripts 117
Check Server Extensions command (FrontPage) 408, 413, 417–420
checklists
capacity planning 154–156
finding potential bottlenecks 154
rolling out new systems 54
CICS (Customer Information Control System), accessing legacy applications and data
extending transactions with COMTI (example) 540–542
integrating
IIS and applications 534
transaction processing data (example) 536–540
migrating to Component Services 553, 557
using COMTI 535
CICS transaction processing programs 536
/Classes directory, Java classes 562
client authentication certificates 472
client authentication, IIS access control 510
client certificate mapping
Active Directory mapping 494
IIS configurations
CTLs 494
functionality 489
IIS native mapping 492

Client Cursor Engine, using RDS 324
Client Monitor, HTTP Monitoring Tool 230
client tiers
  data access 318–325
  defined 249
  developing Web applications
    ActiveX controls 254
    browser support 258
    cascading style sheets 256
    client-side scripts 253
    data binding 257
    dynamic HTML 256
    graphics and multimedia 253
    hyperlinks 253
    limitations 259
    text and HTML 252
  middle tiers *See* middle tiers
ClientCertificates collection 288
client-side cursors 347
client-side data
  access 318
  client tiers *See* client tiers
  middle tiers *See* middle tiers
client-side scripts 253, 294
client-side technologies *See* client tiers, developing Web applications
Close method 339
Cluster Service
  building three-tier Web clusters
    building third tiers 452
    creating 448
    grouping features for reliability 448
  linking with Network Load Balancing 388
  Windows 2000 Server 478
clusters *See* Web clusters
coarse-grained components 363
COBOL (COmmon Business Oriented Language) programs, accessing legacy applications and data using COMTI
  extending transactions with COMTI (example) 540–542
  integrating transaction processing data (example) 536–540
  overview 535
Collaboration Data Objects (CDO) 132
collections
  ClientCertificates 288
  defined 123
  Fields (ADO) 298
  Parameters 356
  QueryString 123
  Request.Form 123, 287
  Request.QueryString 287
collections *(continued)*
  ServerVariables 124, 288
  Session 127, 302
color palettes, HTML standards 579
COM (Component Object Model) components
  See also Windows Script Components
  accessing file systems 131
  application scope 279
  client/server architecture 251
  combining with Active Scripting 266
  developing with Windows Script Components 269
  importing type library constants 278
  integrating data from CICS transaction processing 536
  NTFS/IIS directory security 81
  out-of-process components 282
  programmability architecture 14
  providing process isolation 280
  running in Component Services run-time environment 556
  transactional components 362–365
COM logging interface 428
COM objects
  ADO data *See* ADO (ActiveX Data Objects); Recordsets
  IIS Admin Objects 9
COM Transaction Integrator (COMTI) *See* COMTI (COM Transaction Integrator)
COM+ applications
  See also Component Services
  building Web clusters 446, 453
  database requirements (table) 366
  running out-of-process pools 396
  transaction processing 359, 363
Command object
  avoiding query timeouts 350
  calling stored procedures 356
  closing connections 339
  forward-only/read-only cursors 343
  sharing active connections 340
command-line administration, administering sites remotely 405
command-line scripting, using Fpsrvadm utility 409
commands
  Attach to Process 303
  Cacls (FrontPage) 421
  Check Server Extensions (FrontPage) 408, 413, 417–420
  Copy (Windows) 409
  .htaccess (Apache) 85
  Net Use 230

commands *(continued)*
NetStat 230
Open With FrontPage (FrontPage) 408
Publish (FrontPage) 409
Recalculate Web (FrontPage) 408, 409
RedirectMatch/AliasMatch (Apache) 85
Remove Server Extensions (FrontPage) 408
run (WCAT) 223
CommandTimeout property 350
comments in scripts, style guide 566
commerce Web sites, capacity planning scenarios 163
COmmon Business Oriented Language (COBOL) programs *See* COBOL (COmmon Business Oriented Language) programs
Common Gateway Interface (CGI) applications *See* CGI (Common Gateway Interface) applications
Component Object Model (COM) *See* COM (Component Object Model)
Component Services
See also COM+ applications
installing TakeANumber component 273
integrating
data from CICS transaction processing 536
with Message Queuing 8
migrating legacy transaction processes *See* migrating legacy transaction processes
programmability architecture 15
recovering from crashes 396
security 308
transaction processing on Web
benefits (list) 358
business objects vs. scripts in ASP pages 362
extending limits 360
transactional ASP 361
transactions explained 359
transactional components
distribution and scaling issues 367
participating in transactions 364
using database access interfaces 366
troubleshooting (example) 519
working with ASP 127
Component Services Explorer 556
Component Services programming model 554
Component Services run-time environment 556
components
accessing ADO data with ASP and COM components *See* ADO (ActiveX Data Objects); Recordsets
debugging *See* debugging
deploying with Content Deployment 387
components *(continued)*
encapsulating business logic 125
factoring applications 285
fine-grained vs. coarse-grained 363
stateless vs. stateful objects 363
updating with process isolation 281
vs. scripts in ASP pages 268
Components Wizard (Windows) 230
Computer Management tool 376
COMTI (COM Transaction Integrator)
integrating IIS and legacy applications
extending transactions (example) 540–542
functionality 535
identifying strategies 532
integrating transaction processing data (example) 536–540
using with Information Management System 543
migrating to Component Services 557
COMTI components, developing 537
COMTI objects, developing 540
confidentiality, defined 463
configuration
Apache directives and IIS properties
access.conf directives (table) 96–97
httpd.conf directives (table) 86–91
srm.conf directives (table) 92–95
e-mail 421
FrontPage Server Extensions *See* FrontPage Server Extensions, configuring for ISP installation
FTP sites 74
IIS
creating company domain Web sites 374–379
creating personal Web sites 379
overview 374
restricting content 381
troubleshooting 379
isolated processes 282
migrating
applications 74
settings 75
NES 3.5 configuration settings and IIS properties
application settings (table) 102
content management settings (table) 104
overview 99
server logging settings (table) 102
server preferences (table) 100

configuration *(continued)*
overlapping/nonoverlapping virtual servers 411
replicating configuration settings with Iissync utility 109
script interpreters 114
security
IIS *See* security, IIS configurations
migrating users and groups 77
overview 77
setting IIS permissions 79
setting NTFS permissions 78
setting permissions based on content (table) 80
Windows 2000 Server *See* security, Windows 2000 Server configurations
testing with WCAT *See* WCAT (Web Capacity Analysis Tool)
configuration files, running Script.cfg (WCAT) 392
connection management 127
Connection Manager Administration Kit, remote access component 425
Connection Manager 336
Connection object (ADO)
avoiding query timeouts 350
choosing data providers 331
cursors 343
enhancing performance on SQL Server systems 338
establishing database connections 335
extending limits of transactions 360
restricting connections across pages 341
sharing active connections 340
timing out connection requests 584
Connection Point Services, remote access component 425
Connection Pool Timeout (CPTimeout) property 335, 583
connection pooling, defined 335
connection requests, timing out 584
Connection.Execute method 350
connections
analyzing counter data 200
limiting 194, 205
monitoring 198
optimizing (list) 337–342
optimizing network input/output 215
retrying dead 336
scripting for performance 583
tracking anonymous/nonanonymous 224
using NetStat command 230
ConnectionTimeout property 338, 584
consistency (ACID property), defined 15, 359
constant names, style guide 568
content
backing up Web/FTP sites 109
control options for virtual directories (list) 499
database-centric publishing 311
dynamic, pages vs. databases 308
expiring time-sensitive 73
factoring applications 285
managing content created in FrontPage 409
nested in overlapping virtual servers 410
passthrough to UNC-shared 502
personalizing 73
rating with PICS 445
replicating Web and FTP sites 67
restricting on Web sites 381
setting permissions based on content (table) 80
storing outside Web server root directories 65
updating dynamically 73
uploading through FTP 424
Content Deployment
building three-tier Web clusters 451
replicating and clustering 387
replicating through 383
vs. Content Replication Server 383
/Content directory 562
content expiration 443, 444
content management, NES 3.5 configuration settings and IIS properties (table) 104
Content Replication Server *See* Content Deployment
content specialization and striping
Internet commerce Web sites 165
Internet marketing Web sites 160
Internet transactional Web sites 162
intranet Web sites 158
ContentType property, Response object 129
context switches 202
context switching, style guide 568
Convert 2.0 ACL file NES 3.5 configuration setting 100
Cookie Munger 442
Cookie.asp (sample code) 441
cookies
saving with Web Application Stress Tool 239
session IDs security risks 274
Session object management 270
Session, maintaining state 127
supporting non-HTTP 1.1 compliant browsers 438
Copy command (Windows) 409
copy/paste, replicating issues 67

corruption of data attacks, defined 465
counters
  analyzing data
    anonymous/nonanonymous connections/not-found errors 227
    bandwidth throttling 215
    connections 200
    File System Cache 191
    file transfers 211
    Object Cache 187
    processor activity 196
    SQL Server/Web servers 247
    TCP connections 213
    threads 202
    transmission rates 210
    working sets 185
  baseline logging 247
  collecting performance data 223
  displaying WCAT results 392
  measuring bandwidth in OSI layers (list) 207
  monitoring
    remote computers 217
    Template/Script Engine Caches (list) 219
    Web servers (list) 238
  monitoring memory
    File System Cache (table) 189
    Object Cache (table) 187
    paging 181
    working sets (table) 184
  monitoring network input/output
    anonymous/nonanonymous connections (table) 225
    bandwidth throttling (table) 214
    file transfers (table) 211
    not-found errors (table) 226
    TCP connections (table) 212
    transmission rates at Application Layer (table) 208
    transmission rates at Data Link Layer (table) 209
    transmission rates at Network Layer (table) 209
    transmission rates at Transport Layer (table) 209
  monitoring processor bottlenecks
    connections (table) 200
    processor activity (table) 196
    threads (table) 202
  performance counters
    functionality 228
    using Web Application Stress Tool Sample Script 234
counters *(continued)*
  stress testing with Web Application Stress Tool
    both SQL Server/Web servers performance (table) 245
    SQL Server performance (table) 244
    Web server performance (table) 239, 246
CPTimeout (Connection Pool Timeout) property 335, 583
CPU usage, static vs. dynamic pages 142
CpuAppEnabled property 431
CpuCgiEnabled property 431
CPULimitPause metabase property 198
crash recovery, fault isolation 281
CreateInstance method, ObjectContext object 364
CreateObject method, Server object 266, 276
CreateParameter method 357
CRL-checking, enabling 506
CRLs (Certificate Revocation Lists) 506
CryptoAPI (CAPI) 2.0, performing certificate work 506
cryptography cards, PCMCIA-based 505
cryptography, Windows 2000 Server 473
Cscript.exe 400
CTL Wizard (certificate trust lists) 2
CTLs (certificate trust lists) 494
Cursor Service, using RDS 324
CursorLocation property, Recordset object 322, 347
cursors
  ADO lock types (list) 346
  ADO types (list) 343
  concurrency 346
  forward-only 344
  keyset 346
  location 347
  overview 342
  Recordset functionality by type (list) 343
  static vs. dynamic 345
CursorType property 347
custom Apache modules, migrating 98, 116
custom authentication 510
custom business objects 325–328
custom error messages 3, 85
Custom Errors 218
custom HTTP headers 443
Customer Information Control System (CICS) *See* CICS (Customer Information Control System)
customizing
  extended logging 428
  information returned to users with ASP 560

customizing *(continued)*
ISP installations *See* ISP installations
scripts, administrative architecture 10

D

DACLs (discretionary access control lists)
Anonymous access conflicts 482
cannot perform authoring tasks 418
security 498, 602
setting
for domain Web sites 376
for personal Web sites 379
with Terminal Services 405
Windows 2000 Server 471
DAO (Data Access Objects) 316
data
disclosure and corruption attacks 465
legacy *See* legacy applications and data
data access
access from UNIX-based servers 63
accessing ADO data with ASP and COM components *See* ADO (ActiveX Data Objects); Recordsets
additional resources 371
client tiers *See* client tiers, data access
Component Services 366
Message Queuing 368
middle tiers *See* middle tiers, data access
overview 307
Web database technologies
ADC 315
ADO and RDS 314
benefits (list) 308
cost of data access 317
data publishing considerations (list) 309
database-centric publishing 310–312
Jet and DAO 316
MDAC 312
ODBC and OLE DB 313
overview 308
RDO 316
Data Access Components *See* MDAC (Microsoft Data Access Components)
data access components written in Visual Basic 6.0, troubleshooting (example) 520
Data Access Objects (DAO) 316
data binding, using DHTML 257
/Data directory 563
Data Link Layer (OSI) 207, 209
data loss, security planning 595
data partitioning, ADO using RDS 322
data providers *See* OLE DB provider for AS/400 and VSAM
Data Replicator Manager 552
Data Source Name (DSN) *See* DSN (Data Source Name)
Data Transformation Services (DTS) 549
data warehousing 311
data-aware controls 320
Database Access components 8
database access, ASP vs. Perl Dbperl/DBI 126
database connections, optimizing (list) 337–342
database content, NTFS/IIS directory security 81
database logic 125
database management systems (DBMS) *See* DBMS (database management systems)
databases *See* Web databases
datagrams, counters measuring transmission rates (list) 209
datestamps, capacity planning issues 147
DB2 database tables 550
DBI Perl packages vs. ASP 126
DBMS (database management systems) 8
Dbperl packages vs. ASP 126
DCOM (Distributed COM) components, NTFS/IIS directory security 81
debug exception handling, disabling 302
debugging
HTML files 579
ISAPI and server components 302
just-in-time 6
Script Debugger, IIS architecture 6
script in ASP pages
avoiding common mistakes 296–298
debug tracing 300
overview 296
script management 301
tracking events in Global.asa 300
using Script Debugger 298–299
security planning 600
Web applications and components 295
debugging tools 303
declaratives, ASP server-side scripting 262
decryption 222, 473
default directory structures 65
default home pages for applications 562
Default MIME Type NES 3.5 configuration setting 104
default scripting language, style guide 571
Default Web Site 65, 420
Default.asp, ASP best practices 562
Default.htm, ASP best practices 562

DefaultIcon Apache resource configuration directive 93
DefaultLogonDomain metabase property 79
DefaultType Apache resource configuration directive 93
deferred procedure calls (DPCs) handling, improving performance 204
defragmenting disks, optimizing 193, 220
delegation, support (table) 501
delimiters
  pathnames, migrating UNIX Perl scripts 117
  style guide 569
denial of service, security 465, 595, 603
Deny access permission, Apache vs. IIS 84
deploying
  applications *See* legacy applications and data
  business solutions *See* migration process management
deployment options 61
DER (Distinguished Encoding Rules) certificate type 491
Design Template (Design.doc) 37, 56
destination servers, migration 63
DFS (distributed file system), clustering 389
DHCP (Dynamic Host Configuration Protocol), clustering features 389
DHTML (dynamic HTML) 73, 256
dial-in permissions, granting 378
dialog boxes
  Add Counters (PerfMon) 229
  Add to Chart 217
  Application Configuration 301, 302
  Extended Logging Options 428
  Log on locally 415
  Object Properties 521
  Secure Communications 504
Dictionary object (ASP) 302
Dictionary object (VBScript) 583
Digest authentication
  defined 2
  IIS configurations
    accessing remote resources (table) 501
    file and directory security 498
    functionality 487
    Web authentication summary (table) 495
digital certificates, defined 7
digital signatures, issuing for SSL and Transport Layer Security 491
Dim statement 296, 581, 582
directives
  <%@ TRANSACTION=value %> 572
  Apache directives and IIS properties
    access.conf directives (table) 96–97
    httpd.conf directives (table) 86–91
    srm.conf directives (table) 92–95
  ENABLESESSIONSTATE 582, 583
  @LANGUAGE 571
directories
  root, ASP best practices 561
  security 498
  setting NTFS permissions 78
  users, Apache vs. IIS 84
directory aliases
  Apache vs. IIS 85
  NES vs. IIS virtual directories 99
directory browsing and indexing 73
Directory browsing permissions, customizing scripts (example) 402
directory hierarchy, UNIX 71, 118
Directory Indexing NES 3.5 configuration setting 105
directory separators, UNIX 71, 118
Directory Service Access audit event 479
<Directory> access configuration directive 96
DirectoryIndex Apache resource configuration directive 93
<DirectoryMatch> access configuration directive 97
DisableBacklogMonitor registry key 221
DisableCommit method, ObjectContext object 363
disclosure of data attacks, defined 465
discretionary access control lists (DACLs) *See* DACLs (discretionary access control lists)
Disk (Logical or Physical) counters
  % Disk Time 246
  Avg. Disk Bytes/Transfer 246
  Queue Length 246
disk input/output, memory 217
disk mirroring, optimizing memory 193
disk paging, memory monitoring 181
disk partitions, migrating 63
disk quotas, Windows 2000 Server 477
disk space, migration issues 63
Distinguished Encoding Rules (DER) certificate type 491
Distributed COM (DCOM) components, NTFS/IIS directory security 81
distributed file system (DFS), clustering 389
Distributed interNet Applications (DNA), client/server architecture 251
distribution files, running Script.dst (WCAT) 392

/DLLs directory 563
DNA (Distributed interNet Applications), client/server architecture 251
DNS (Domain Name System) round-robin distribution, capacity planning issues 152
DNS Management tool 374
DNS servers
  creating three-tier Web clusters 450
  sample installations 455
Document Footer (Footer Text) NES 3.5 configuration setting 104
document footers, specifying 73
Document Root NES 3.5 configuration setting 105
DocumentRoot Apache resource configuration directive 93
Domain Name System (DNS) round-robin distribution *See* DNS (Domain Name System) round-robin distribution
Domain Name System (DNS) servers *See* DNS servers
domain names 374, 435
domain Web sites 374, 379
dot notation (example) 401
download time, capacity planning 143
downloadable executable files, security 81
downtime, during migration 55
DPC (deferred procedure calls) handling, improving performance 204
DS1/T1 connections, traffic 141, 144
DS3/T3 connections, traffic 144
DSL connections, traffic 141
DSN (Data Source Name) 329, 332
DSN-less connections 330
DTS (Data Transformation Services), accessing legacy applications and data 549
DTS Export Wizard 550
DTS Import Wizard 550
DTS packages, creating with wizards 550
durability (ACID property), defined 15, 359
Dynamic Configuration Files NES 3.5 configuration setting 100
dynamic content 73, 308
dynamic cursors vs. static cursors 345
Dynamic Host Configuration Protocol (DHCP), clustering features 389
dynamic HTML, client-side controls 256
dynamic pages vs. static pages 142, 309, 317

E

ECB (Extension Control Block) 119
e-commerce Web sites 388
e-mail
  formatting with CGI 132
  migrating UNIX mail scripts 134
  problems submitting forms 421
  security 603
  sending using ASP 132
embed statements, HTML standards 580
<EMBED> HTML tag 580
Enable DNS NES 3.5 configuration setting 100
EnableCommit method, ObjectContext object 363
ENABLESESSIONSTATE directive 582, 583
encryption
  capacity planning issues 164
  monitoring security overhead 222
  resources 529
  using
    CryptoAPI 506
    Fortezza 505
    Server-Gated Cryptography 2
    SSL and Transport Layer Security 504
    Transport Layer Security 474
  Windows 2000 Server 473
encryption key, session security 274
Encryption NES 3.5 configuration setting 100
End If statement 298
end-of-line markers 69, 117
end-point servers 384, 451
Enterprise Service Providers, administering installations *See* ISP installations
environment variables
  Http_Referer 133
  migrating from CGI to ASP 124
  Query_String 123
  Remote_User 133
  Server_Name 124, 131
EOF (End of File) property (Recordset) 342
Err object 268
error messages
  custom 3, 85
  displaying, HTML standards 580
  using Script Debugger 6
Error Responses NES 3.5 configuration setting 100
ErrorDocument Apache resource configuration directive 94
ErrorLog Apache server directive 86
errors
  <!-- #include --!> statements 514
  = (equal sign) 297
  401 Access Denied 481
  401/403 HTTP 508
  403 517

errors *(continued)*
403,7: Forbidden—client certificate required 510
403,x: Access Forbidden 510
404 Object Not Found 439
500: Server Too Busy 430
Access forbidden error 403 226
ASP 0115 302
ASP 0177: Server.CreateObject Failed 305
Cannot open file 414
Cannot rename file 414
Cannot save file 414
controlling with process isolation 281
ErrorDocument 404 (Apache) 85
fatal application errors 396
FrontPage error when submitting forms 421
Header Error 292
HTTP status error 404 226
ISAPI error 111
monitoring remote computers with PerfMon 217
ODBC Drivers error '80004005' 332–333
overlapping virtual servers vs. installing FrontPage 411
scripting (list) 296–298
Server object error 'ASP 0196' 282
uploading executable scripts and programs 424
VBS Script Error: Object Required: Session 265
Ethernet connections, traffic 141
event handlers
_OnClick 289
_OnSubmit 289
OnTransactionAbort 361
OnTransactionCommit 361
Event Viewer (Eventvwr.msc) 217, 231, 478, 482
events
Account Logon (audit) 479
Account Management (audit) 479
Application_OnEnd 276
Application_OnStart 276
Directory Service Access (audit) 479
Logon (audit) 479
Object Access (audit) 479
OnClick 322
Policy Change (audit) 479
Privilege Use (audit) 479
Process Tracking (audit) 479
Session_OnEnd 276
Session_OnStart 271, 276, 295
System (audit) 479
events *(continued)*
tracking in Global.asa 300
Window_OnLoad 265, 352
Eventvwr.msc (Event Viewer) 217, 231, 478, 482
exception handling, disabling 302
executable CGI applications, NTFS/IIS directory security 80
executable scripts, viruses and bugs 423
Execute method, Connection/Command objects 343
Execute permissions 81
ExpiresActive Apache server directive 87
ExpiresDefault Apache server directive 87
expiring content 73
Extended Logging Options dialog box 428
Extension Control Block (ECB) 119
extension lengths, UNIX vs. Windows file systems 70
extensions for files, standards 564
extranets, database-centric publishing 310

F

failback, defined 389
failover, defined 389
false repudiation, threat assessment 594
FancyIndexing Apache resource configuration directive 94
FAT (File Allocation Table) file systems 69, 78, 418
fault isolation, crash recovery 281
fault tolerance, clustering 388
faults, defined 192
Field object 351, 355
Fields collection (ADO) 298
file access, troubleshooting 510
File Allocation Table (FAT) file systems *See* FAT (File Allocation Table) file systems
file conventions (table) 118
file descriptors, UNIX 118
file handles, UNIX adhering to Windows conventions 118
file links, Windows file systems issues 69
file name extension standards 564
file names, UNIX 70, 118
file ownership, Windows file systems 69
file security, IIS configurations 498
file sharing with UNIX-based servers 63
file system access, migrating 131
File System Cache
analyzing counter data 191
defined 189
IIS 5.0 functionality 189

File System Cache *(continued)*
memory monitoring specifics 181
memory requirements 178
monitoring with counters (table) 189
optimizing memory usage 194
file systems, Windows vs. UNIX 70
file transfers
measuring 207
monitoring counters (table) 211
<Files> access configuration directive 97
<FilesMatch> access configuration directive 97
FileSystemObject (ASP) 98, 131
fine-grained components vs. coarse-grained components 363
firewalls 455, 458, 604
Footer Text (Document Footer) NES 3.5 configuration setting 104
footers 73, 446
Form collection 123
Form object 289
Form Wizard, creating 291
<FORM> HTML tag 123, 286
Format NES 3.5 configuration setting 103
formatting disk partitions, security 63
forms *See* HTML forms
Fortezza 2, 505
Forward Requests To NES 3.5 configuration setting 106
forward-only cursors 344
Fpremadm utility 408, 409
Fpsrvadm utility 408, 409
Fpsrvadm.exe command-line utility 414
Fpsrvwin utility vs. FrontPage snap-in 408
front-end networks, sample installations 455
FrontPage administration commands 408
FrontPage Server Extensions
administering
ISP installations 373
permissions on content roots (list) 410
changing Web site roots 410
configuring for ISP installation
configuring e-mail 421
introducing FrontPage snap-in 408
managing content 409
overlapping virtual servers 411
overview 407
uploading executable scripts and programs 423
creating Web sites 378
installing 411, 451
permissions
administering with (list) 410
cannot access Web sites 417
FrontPage Server Extensions *(continued)*
permissions *(continued)*
cannot perform authoring tasks 418
common problems/resolutions 413
granting Log on locally rights 415
resetting in sub-webs 417
resetting on virtual servers 419
setting security on IIS 5.0 Server 412
setting 413
troubleshooting for unauthorized users 412
publishing on Web sites 18
FrontPage snap-in, functionality 408
FrontPage Web documents 562
FrontPage webs 18, 412
FrontPage, repairing hyperlinks 72
FTP 19, 424
FTP authentication, configuring IIS 496
FTP connections, analyzing data 201
FTP servers, IIS architecture 5
FTP Service counters
Bytes Received/sec 208
Bytes Sent/sec 208
Bytes Total/sec 208
Current Anonymous Users 225
Current Connections 200
Current NonAnonymous Users 225
Files Received 211
Files Sent 211
Files Total 211
Maximum Anonymous Users 225
Maximum Connections 200
Maximum NonAnonymous Users 225
Total Anonymous Users 225
Total NonAnonymous Users 225
FTP site directories 65
FTP sites
limiting access 74
migrating 74, 109
replicating 67
setting IIS permissions 79
full-domain sites, creating 374
Functional Specification Template (FuncSpec.doc) 36, 56
functions
ASP server-side scripting 263
performance 581
style guide 576

G

gateway logic 125
Generate Report NES 3.5 configuration setting 103

GET method (ASP) 123, 287
GET request 141, 429
GET/PUT notation (example) 401
GetChunk method, Field object 355
Global counters *See* Internet Information Services Global counters
global scope, performance 581
global variables, performance 581
Global.asa
  declaring objects 278, 562
  functionality 276
  importing type library constants 278
  tracking events 300
  using <OBJECT> tag 582
GlobalPageCounter object (Global.asa) 278
graphics, Web applications 253
group accounts 77
Group Policy editor 468

H

hard page faults, memory monitoring 181
hardware
  microsoft.com case study 168, 171
  migrating Web servers to IIS 5.0 61
  migration considerations 40–43
  three-tier Web clusters 450
  upgrading to IIS 107
  using custom for security overhead 227
Hardware Virtual Servers (IP Address, Document Root) NES 3.5 configuration setting 105
hardware virtual servers, NES vs. IIS 98
hash tables, storing Transmission Control Blocks 199
HDR (Host Data Replicator), legacy databases
  flexible processing and filtering 551
  gathering statistics 551
  identifying strategies 532
  performance 552
  performing two-way replication 551
  replicating DB2 tables 550
  scheduling 551
  security 552
  supported platforms (list) 552
Header Apache server directive 87
HeaderName Apache resource configuration directive 94
headers
  host *See* host headers
  HTTP 1.1, capacity planning 146
  HTTP *See* HTTP headers
  If-Modified-Since, capacity planning 147
  Negotiate 486
helper files 563
hit counters, displaying 135
hits, defined 192
home page names, UNIX (list) 76
Home Page NES 3.5 configuration setting 106
Home Page/Index File NES 3.5 configuration setting 105
home pages, default for applications 562
Host Data Replicator (HDR) *See* HDR (Host Data Replicator)
host header names, assigning 374, 434
host headers
  migrating user Web sites 72
  NES vs. IIS 99
  one computer/same IP address/multiple hosts 437
  supporting older browsers 438
HostnameLookups Apache server directive 87
HSE_REQ_TRANSMIT_FILE (ServerSupportFunction) function 148
.htaccess command (Apache) 85
/HTM subdirectory of /Content directory 563
HTML
  combining with server-side scripts 128
  creating on the fly *See* ASP (Active Server Pages), scripts
  formatting output 129
  functionality in Web applications 252
  publishing with databases *See* Web databases
  pure vs. special dialect 128
  supporting non-HTTP 1.1 compliant browsers 438
  vs. .asp 560, 564
HTML Administration Forms, administering sites remotely 409
HTML files, checking and debugging 579
HTML footers, customizing 446
HTML forms
  accessing input, migration 121
  content 121
  developing Web applications
    GET vs. POST 287
    hidden form fields 291
    using forms for input 286
    validating client-side forms 289
  factoring applications 285
  security risks 291
HTML output logic 128
HTML pages 142, 262
HTML standards 578
HTML tags
  ACTION attribute 286
  <EMBED> 580

HTML tags *(continued)*
<FORM> 123, 286
<NOEMBED> 580
<OBJECT> 278, 580, 582, 585
<OPTION> 265
RUNAT=SERVER attribute 263, 278
<SCRIPT> 263, 276, 278, 301
<SELECT> 265, 351
HTTP 1.1
header entries 146
publishing on Web sites 17
supporting non-compliant browsers 438–442
HTTP 1.1 Host Header standard 72
HTTP compression, defined 4
HTTP connection data structures 178
HTTP connections, analyzing data 201
HTTP cookies, Session object 270
HTTP headers
Accept-Language 261
redirecting requests 292
Set-Cookie 270
setting up custom 443
vs. default MIME types 129
HTTP Keep-Alives 199, 217
HTTP Monitoring Tool
functionality 230
monitoring
performance 433
TCP connections 212
HTTP Persistent Connection Timeout NES 3.5 configuration setting 100
HTTP Post (RFC 1867) protocol 386
HTTP redirect, specifying from Web sites 73
HTTP server applications, counters 208
Http_Referer environment variable 133
httpd.conf Apache directives, IIS properties (table) 86–91
HttpExtensionProc ISAPI call 260
hyperlinks
factoring applications 285
functionality in Web applications 253
repairing during migration 72
replicating Web sites 68
testing migration 48

I

IBM AS/400 *See* legacy applications and data
IBM DB2 database tables 550
IBM Multiple Virtual Storage (MVS) *See* legacy applications and data
IBM OS/400 *See* legacy applications and data
IdentityCheck Apache server directive 87
IDs, session (ASP) *See* session IDs (ASP)
If statement 297
<IfDefine> Apache server directive 87
If-Modified-Since header 147
IIS (Internet Information Services) 5.0
Access databases security limitations 81
accessing legacy applications and data *See* legacy applications and data
additional resources 19
administering older version servers 426
architecture
administrative 9
overview 5–8
programmability 11–16
configuring
creating company domain Web sites 374–379
creating personal Web sites 379
overview 374
restricting content 381
troubleshooting 379
directory security, setting permissions based on content (table) 80
enabling CRL-checking 506
migrating from
Apache HTTP Server *See* migrating from Apache HTTP Server
NES *See* migrating from Netscape Enterprise Server
new features
administration (list) 3
Internet standards (list) 4
programmability (list) 3
security (list) 2
publishing on Web sites
overview 17
using FrontPage Server Extensions 18
using FTP 19
using WebDAV 17
security *See* security, IIS configurations
vs. IIS 4.0, tuning ASP queuing and thread pools 218
IIS 5.0 working sets *See* working sets
IIS Admin Objects
administering sites remotely 404–407
architecture 9
automating
administration 397
tasks with command-line scripts 10
security-related settings (list) 524
working with Windows Script Host and ADSI 10

IIS Admin Service, setting permissions 78
IIS Administration Script Utility 79
IIS directories, populating with Web/FTP site content 67
IIS log files, migrating 75
IIS logging 200, 524
IIS mapper, accessing remote resources (table) 501
IIS metabase, setting IIS properties 75
IIS Migration Wizard
  defined 39, 57
  migrating Apache directives
    Apache access.conf directives and IIS properties (table) 96–97
    Apache srm.conf directives and IIS properties (table) 92–95
    httpd.conf directives and IIS properties (table) 86–91
  migrating Web servers to IIS 5.0 67
  NES 3.5 configuration settings and IIS properties
    application settings (table) 102
    content management settings (table) 104
    server logging settings (table) 102
    server preferences (table) 100
    Web publishing/Agents and Search/Auto Catalog 106
  resources 135
  upgrading or replicating IIS Web servers 107
IIS native mapping 492
IIS Object Cache *See* Object Cache
IIS permissions, configuring 79
IIS properties, setting 75
IIS script maps 261
IIS snap-in
  administering sites remotely 404, 405
  configuring servers 75
  controling connections 194
  Default Web Site 65
  deploying content 385
  FrontPage error when submitting forms 422
  MMC snap-in *See* MMC snap-in
  NES 3.5 configuration settings and IIS properties
    application settings (table) 102
    content management settings (table) 104
    server logging settings (table) 102
    server preferences (table) 100
  vs.
    FrontPage snap-in 408
    Windows Explorer 76
IIS Template Cache
  caching scripts in ASP 186
  IIS 5.0 vs. IIS 4.0 218
  memory requirements 178
  monitoring with PerfMon counters (list) 219
IIS Web servers, upgrading or replicating *See* migrating Web servers, upgrading or replicating IIS Web servers
IISAO (IIS Admin Objects) *See* IIS Admin Objects
IIsCertMapper (IISAO) property 525
Iissync utility 109
IISWebService object 283
IISWebVirtualDir object 283
illegal characters, UNIX vs. Windows file systems 71
Image File Execution Options registry key 304
/Images subdirectory of /Content directory 563
impersonation, Windows 2000 Server 471
Import Wizard (COBOL) 538, 539, 541, 542
IMS (Information Management System), accessing legacy applications and data
  COMTI support 543
  extending transactions with COMTI (example) 540–542
  integrating IIS and applications 534
  migrating to Component Services 553
  using COMTI 535
.inc file extension standards 564
Include Apache server directive 87
include files 564, 601
includes, IIS support for server-side 73
indentations, style guide 570
Index Filenames NES 3.5 configuration setting 105
IndexIgnore Apache resource configuration directive 94
Indexing Service
  IIS architecture 7
  MSIDXS (OLE DB provider) support 331
indexing, enabling on Web sites 73
Inetinfo working sets *See* working sets
Inetinfo.exe 183, 304
Information Management System (IMS) *See* IMS (Information Management System)
infrastructure deployment *See* migration process management
in-process applications, performance 119
in-process extensions, ISAPI 261
in-process isolation, running 394
input processing, migrating 121
input stream, accessing unprocessed 123

installation requirements, capacity 153
installations, administering ISP installations *See* ISP installations
integrated Windows authentication
  accessing remote resources (table) 501
  file and directory security 498
  functionality 486
  Web authentication summary (table) 495
integration test labs 40, 46
integration tests (table) 49
integrity 463, 593, 602
integrity protocols 473
interactive logons 469
interactive permissions 499
Interactive user accounts, security 498
International Characters NES 3.5 configuration setting 105
Internet Authentication Services, remote access component 425
Internet Connection Services for remote access 425
Internet Explorer
  enabling CRL-checking 506
  locking in security settings 604
  posting content 386
Internet Information Services (IIS) 5.0 *See* IIS (Internet Information Services) 5.0
Internet Information Services Global counters
  Cache Flushes 187
  Cache Hits 187
  Cache Hits % 187, 246
  Cache Misses 187
  Current Blocked Async I/O Requests 214
  Directory Listings 187
  Measured Async I/O Bandwidth Usage/Minute 214
  Objects 187
  Total Allowed Async I/O Requests 214
  Total Blocked Async I/O Requests 214
  Total Rejected Async I/O Requests 214
Internet Server Application Programming Interface (ISAPI) applications *See* ISAPI (Internet Server Application Programming Interface) applications
Internet service providers (ISPs) *See* ISPs (Internet service providers)
Internet Services Manager 10, 562
Internet Services Manager (HTML) 10, 404
Internet standards, IIS new features (list) 4
Internet, threat assessment 592
InterNIC, domain names 374, 435
Interoperability Lab, resources 57
interoperating servers, migrating 37
interoperation tools, Web links 136
interoperation, file sharing 63
Interrupt Request Level (IRQL) 204
interrupts, improving handling 204
intranets
  capacity planning scenarios 157
  database-centric publishing 310
  expanding with content striping and specialization 158
  threat assessment 590
intrinsic objects (ASP) 111
inventories, creating for security 589
IP Address NES 3.5 configuration setting 105
IP addresses
  assigning to domains 374
  configuring IIS security 509
  hosting multiple sites/one IP address 434
  migrating user Web sites 72
  one computer/multiple IP addresses 435
  one computer/same IP address/multiple hosts 437
  pinging 379
IP counters
  Datagrams Forwarded/sec 209
  Datagrams Received/sec 209
  Datagrams Sent/sec 209
  Datagrams/sec 209
IP packet filtering, security 604
IP Security (IPSec) 475–476
ISAPI (Internet Server Application Programming Interface) applications
  converting CGI applications 119
  errors with CGI scripts 111
  migrating to IIS 5.0 110
  running in isolation 395
  vs. ASP/CGI/static HTML pages performance tests 148
  Web applications performance 148–149
ISAPI authentication filters 381, 518
ISAPI DLLs 119, 184, 395
ISAPI Extension Control Block (ECB) 260
ISAPI extensions 13, 111, 112, 260
ISAPI filters
  Cookie Munger 442
  developing middle-tier Web applications 261
  extending IIS security 497
  migrating
    Apache custom modules 98
    applications 111
  NTFS/IIS directory security 80
  programmability architecture 13

ISAPI in-process, performance tests and comparisons (table) 149
ISAPI out-of-process, performance tests and comparisons (table) 149
ISAPI server extensions, NTFS/IIS directory security 80
ISDN connections, traffic capacity 141
isolation *See* process isolation
isolation (ACID property), defined 15, 359
ISP installations
  adapting IIS to sample installations
    administrative networks 456
    advantages of topology 456
    back-end networks 456
    expanding platforms 457
    front-end networks 455
    overview 453
    querying SQL Server 457
    security 458
    several business clients 454
    three networks in one 455
  additional resources 460
  administering older version servers 426
  administering sites remotely
    command line 405
    IIS snap-in 404
    Internet Services Manager (HTML) 404
    overview 404
    Telnet 406
    Terminal Services 405
    Web site Operators user accounts 407
  allocating resources
    controlling bandwidth 430
    disabling socket pooling 430
    logging resources 428
    overview 428
    performance with HTTP Monitoring Tool 433
    process accounting and throttling 431
    stopping CGI applications 433
  automating administration
    changing permissions (example) 400
    creating Web sites (example) 400
    customizing scripts (example) 401
    executing scripts 398
    IIS Admin Objects/ADSI 398
    overview 397
    Windows Script Host 397
  building Web clusters *See* Web clusters
  configuring FrontPage Server Extensions *See* FrontPage Server Extensions, configuring for ISP installation
ISP installations *(continued)*
  configuring IIS
    creating company domain Web sites 374–379
    creating personal Web sites 379
    overview 374
    restricting content 381
  customizing installation
    components for administration (sample setup) 440
    customizing HTML footers 446
    hosting multiple sites with one IP address 434
    making redirects work 442
    overview 434
    setting up custom HTTP headers 443
    supporting non-HTTP 1.1 compliant browsers 438–442
  enhancing reliability
    clustering 388
    hosting applications 396
    out-of-process pooling 396
    overview 382
    recovering from crashes 396
    replicating 383
    replication and clustering in IIS 383
    running applications in isolation 394
  Internet Connection Services for remote access 425
  managing installations 382
  overview 373
  testing applications with
    WCAT *See* WCAT (Web Capacity Analysis Tool)
    Web Application Stress Tool *See* Web Application Stress Tool
  uploading content through FTP 424
ISPs (Internet service providers)
  See also administrators
  capacity planning issues 156
  IIS
    administrative architecture 9
    new features (list) 3
  monitoring and tuning servers *See* monitoring and tuning servers
  setting up user Web sites 72
  user diagram 34
IUSR_computername accounts
  data source permissions 332
  defending against malicious attacks 523
  IIS configurations
    anonymous Web authentication 481
    DACLs conflicts 482

IUSR_computername accounts *(continued)*
IIS configurations *(continued)*
directories containing DLLs 498
resecuring sites 412
security tokens 495
inability to create components 305
managing permissions with FrontPage, common problems/resolutions 413
resetting permissions 417, 419
security and SQL Server 332
setting permissions 78, 79, 499
troubleshooting 513, 519
IWAM_computername accounts 519, 523

J

Java classes, /Classes directory 562
Java NES 3.5 configuration setting 102
Java, deploying content 385
Jet (Joint Engine Technology) database engine, accessing with DAO 316
JIT (just-in-time)
debugging 6
activation 364
Job object 247
JScript
IIS support 110
style guide for scripts in ASP pages (list) 565
JScript scripting engines 397
just-in-time (JIT)
debugging 6
activation 364

K

KeepAlive Apache server directive 88
Keep-Alives, HTTP 199, 217
KeepAliveTimeout Apache server directive 88
Kerberos v5 authentication protocol
accessing remote resources (table) 501
authentication protocol 82
security 2, 602
vs. integrated Windows authentication 472
Windows 2000 support 486
keyset cursors 346
keysets, defined 346

L

L2 caches, upgrading 204
language default, style guide 571
@LANGUAGE directive 571
LanguagePriority Apache resource configuration directive 94
layout order of scripts, style guide 571
LBound method, VBArray object (VBScript/JScript) 298
LCase() 576
legacy applications and data
accessing data
file data and IIS 543
host files (example) 545–548
overview 543
using OLE DB and ADO 544
additional resources 558
expanding platforms, guidelines 458
identifying strategies
connecting to SNA 533
developing and deploying applications on Windows 534
overview 532
integrating IIS using COMTI
extending transactions (example) 540–542
functionality 535
integrating transaction processing data (example) 536–540
overview 534
using with Information Management System 543
migrating transaction processes *See* migrating legacy transaction processes
replicating databases *See* replicating, legacy databases
legacy transaction processing programs *See* transaction processing programs, legacy
<Limit> access configuration directive 97
<LimitExcept> access configuration directive 97
line feed characters, migrating UNIX Perl scripts 117
list boxes, filling with VBScript (example) 351
Listen Apache server directive 88
Listen_Back_Log registry key 216
ListenBacklog Apache server directive 88
load balancing 152, 388, 389
local scope, performance 581
local variables, performance 581
LocalSystem accounts 78, 470
Location object 294
<Location> access configuration directive 97
<LocationMatch> access configuration directive 97
Lock method, Application object 279
lock types
adLockBatchOptimistic 346, 347
adLockOptimistic 346

lock types *(continued)*
adLockPessimistic 346
adLockReadOnly 343, 346
Log Client Accesses NES 3.5 configuration setting 103
log files 75, 178
Log on locally dialog box 415
Log on locally permission 376
Log on locally privilege 469
Log on locally user right 484
Log Preferences NES 3.5 configuration setting 103
logging
allocating resources 428
baseline 247
troubleshooting 510
logic, maintaining state between forms 127
Logon audit event 479
logon privileges, Windows 2000 Server 468
logon types, Windows 2000 Server 468
Logon.asp, troubleshooting (example) 519
LogonMethod (IISAO) property 525
loop pages, factoring applications 286
loopback, defined 210

M

mail, migrating UNIX mail scripts 134
mainframe transaction processing *See* transaction processing programs
malicious attacks
defending against 522
threat assessment planning *See* threat assessment planning
Map to Directory NES 3.5 configuration setting 104
MapPath method 276, 574
mapping
Active Directory 494
client certificates 489
IIS native 492
setting applications 115
transaction tasks to Windows 2000 Server 557
marketing Web sites, capacity 159
MaxClients Apache server directive 88
Maximum Simultaneous Requests NES 3.5 configuration setting 101
MaxKeepAliveRequests Apache server directive 88
MaxPoolThreads registry key 203
MaxRecords property, Recordset object 353
MaxRequestsPerChild Apache server directive 88
MD_DISABLE_SOCKET_POOLING metabase property 182
MDAC (Microsoft Data Access Components)
ADO and RDS 314, 318
cost of data access 317
data publishing considerations 309
functionality 312
ODBC and OLE DB 313
using with ADO 267
MDL (Memory Descriptor List) reads counters 190, 192
/Media subdirectory of /Content directory 563
memory
adding 227
allocation 177
bottlenecks, improving performance 204
capacity planning guidelines 174
disk input/output 217
management 177
monitoring
security overhead 222
with counters *See* counters
monitoring overall server memory
maintaining Object Cache 186
memory monitoring specifics 180
monitoring File System Cache 189
monitoring IIS 5.0 working sets 183
optimizing memory usage (list) 193
overview 180
tuning TCP sockets 182
using Inetinfo working sets 183
optimizing Web applications 220
overview 177
requirements of servers (list) 178
tuning ASP queuing and thread pools, IIS 5.0 vs. IIS 4.0 219
Memory counters
Available Bytes 181, 184, 191, 245, 246
Cache Bytes 189
Cache Faults/sec 181, 189
Committed Bytes 245, 246
Page Faults/sec 181, 184, 190, 242
Page Reads/sec 181, 184, 190
Pages Input/sec 184
Pages/sec 245, 246
Pool Nonpaged Bytes 182, 245, 246
Pool Paged Bytes 182
Memory Descriptor List (MDL) reads counters 190, 192
Message Queuing
architecture 8
clustering features 389

Message Queuing *(continued)*
functionality 368
transaction processing 369
metabase
backing up when migrating to IIS 109
editing settings 182
using IIS 283
metabase properties
ASPProcessorThreadMax 219
CPULimitPause 198
DefaultLogonDomain 79
MD_DISABLE_SOCKET_POOLING 182
MetaDir Apache resource configuration directive 94
MetaSuffux Apache resource configuration directive 94
method calls
Redirect 536
Reformat 536
methods
Close 339
Connection.Execute 350
Create Object, Server object 266
CreateInstance, ObjectContext object 364
CreateObject, Server object 276
CreateParameter 357
DisableCommit, ObjectContext object 363
EnableCommit, ObjectContext object 363
Execute, Connection/Command objects 343
GET (ASP) 123, 287
GetChunk, Field object 355
LBound, VBArray object (VBScript/JScript) 298
Lock, Application object 279
MapPath (ASP) 574
MapPath, Server object 276
Open, ADO Connection object 335
POST (ASP) 287
Query, RDS.DataFactory object 322
Recordset.Open 343, 350
Redirect, Response object 292
Refresh, Command object 356
Requery, Recordset object 348
Response.AppendToLog, Response object 300
Response.Write, Response object 276, 296
Resync 342, 344, 349
Server.CreateObject (ASP) 278, 282, 305
Server.MapPath (ASP) 574
Server.UrlEncode (ASP) 287
Session.Abandon, Session object 276
SetAbort, ObjectContext object 361
SetComplete, ObjectContext object 361
style guide 573
methods *(continued)*
Submit, Form object 289
UBound, VBArray object (VBScript/JScript) 298
Unlock, Application object 279
Write, Response object 262, 295
Microsoft Access *See* Access databases
Microsoft Access data provider (Microsoft.Jet.OLEDB.3.51) 331
Microsoft Active Directory Service Interfaces (ADSI) *See* ADSI (Active Directory Service Interfaces)
Microsoft Certificate Services *See* Certificate Services
Microsoft COM Transaction Integrator (COMTI) *See* COMTI (COM Transaction Integrator)
Microsoft CryptoAPI (CAPI) 2.0 506
Microsoft Data Access Components (MDAC) *See* MDAC (Microsoft Data Access Components)
Microsoft Federal Security Initiative 505
Microsoft Host Data Replicator (HDR) *See* HDR (Host Data Replicator)
Microsoft Indexing Service
IIS architecture 7
MSIDXS (OLE DB provider) support 331
Microsoft Internet Explorer
enabling CRL-checking 506
locking in security settings 604
posting content 386
Microsoft Interoperability Lab, resources 57
Microsoft Management Console (MMC) *See* MMC (Microsoft Management Console)
Microsoft Network File Sharing (NFS), file sharing with UNIX-based servers 63
Microsoft OLE DB providers *See* OLE DB providers
Microsoft Script Debugger 6, 299
Microsoft SNA (Systems Network Architecture) Server 4.0 SP2, accessing legacy applications and data 531, 534, 536
Microsoft Solutions Framework (MSF) for Infrastructure Deployment *See* migration process management
Microsoft Telnet Service 406
Microsoft Visual Basic Scripting Edition (VBScript) *See* VBScript (Visual Basic Scripting Edition)
Microsoft Visual InterDev 562
Microsoft Web Publishing Wizard 386
Microsoft Windows DNA (Distributed interNet Applications) 251

microsoft.com Web site
large-site case study 166–174
migration testing 47
servers' memory 193
Microsoft.Jet.OLEDB.3.51 (Microsoft Access data provider) 331
middle tiers
building three-tier Web clusters 452
client tiers *See* client tiers
data access 326
defined 249
developing Web applications
application testing 284
ASP applications *See* ASP applications, developing middle-tier Web applications
CGI applications 260
ISAPI extensions and filters 260
overview 259
process isolation and crash recovery *See* process isolation
scripts in ASP pages 261–269
using Component Services for legacy applications 555
migrating from Apache HTTP Server
Apache vs. IIS 83
migrating custom modules 98
migrating directives
Apache access.conf directives and IIS properties (table) 96–97
Apache srm.conf directives and IIS properties (table) 92–95
httpd.conf directives and IIS properties (table) 86–91
overview 86
overview 83
using IIS Migration Wizard 67
migrating from Netscape Enterprise Server 3.5
migrating configuration settings
configuration styles 104
NES 3.5 application settings and IIS properties (table) 102
NES 3.5 server logging and IIS properties (table) 102
NES 3.5 server preferences and IIS properties (table) 100
overview 99
Web publishing/Agents and Search/Auto Catalog 106
NES 3.5 vs. IIS, terminology (list) 98
using IIS Migration Wizard 67
migrating legacy transaction processes
See also transaction processing programs
migrating to Component Services
Component Services Explorer 556
features and capabilities 554
mapping tasks to Windows 2000 Server-based applications 557
overview 553
planning migration 557
second-tier run-time environment 556
three-tier programming model 554
transaction functionality 553
migrating Web servers to IIS 5.0
additional resources 135
identifying items to migrate 63
migrating Web applications
converting CGI to ISAPI 119
deciding to port or rewrite CGI applications 112
IIS application technologies (list) 110
migrating from CGI to ASP 120–131
migrating UNIX mail scripts 134
overview 110
porting 114
reproducing CGI e-mail delivery 132
reproducing CGI page counters 135
UNIX applications 116–118
migration steps
assessing hardware requirements 61
completing FTP site migration 74
completing Web site migration 72
configuring security 77
integrating UNIX and Windows 2000 Server security 82
migrating configuration settings 75
migrating log files 75
migrating security certificates 82
migrating users and groups 77
migrating Web and FTP sites 67
overview 60
preparing destination server 63–66
replicating and configuring applications 74
replicating UNIX-based files 69–71
replicating Windows-based files 67
setting IIS permissions 79
setting NTFS permissions 78
setting permissions based on content (table) 80
using IIS Migration Wizard 67
overview 59
tools for UNIX-based servers (list) 33
upgrading or replicating IIS Web servers 107

migration process management
additional resources 56
assessing risk (table) 28
Deploying phase 52
Developing phase 45–52
Envisioning phase 24–29
hardware 40
IIS Migration Wizard *See* IIS Migration Wizard
interoperating 37
migration tools 39, 57
MSF process model and migration activities (table) 23
Planning phase 31–45
policy and procedure review 35
Release milestone 45
requirements definition (table) 26
software 40
support documents
Design Template 37
Functional Specification Template 36
Project Plan Template 43
Risk Assessment Form 28
Sample Project Schedule 44
Sample Test Plan 47
Standards Review Form 35
Technical Environment Survey 32
Tools Checklist 33
system users of ISP diagram 34
team roles (table) 29, 32, 46, 53
tests
application (table) 50
integration (table) 49
unit (table) 48
Vision and Scope documents
assessing risk 28
creating requirements definition 26
creating 24
defining project teams 29
defining projects 25
developing conceptual designs 27
migration tools 57
Migration Wizard *See* IIS Migration Wizard
MIME content 355
MIME formats 322
MIME mapping 280
MIME packets 326
MIME Types NES 3.5 configuration setting 101
MIME types 92, 104, 129
Min/MaxSpareServers Apache server directive 89
misses, defined 192
mixed environments, file sharing with UNIX-based servers 63
MMC (Microsoft Management Console) 6, 199
MMC snap-in
IIS architecture 6
IIS snap-in *See* IIS snap-in
Internet Services Manager, administrative architecture 10
Monitor Current Activity NES 3.5 configuration setting 103
monitoring and tuning servers
memory
maintaining Object Cache 186
monitoring File System Cache 189
monitoring IIS 5.0 working sets 183
monitoring overall server memory 180
monitoring specifics 180
optimizing memory usage (list) 193
overview 177
requirements of servers running IIS 5.0 (list) 178
tuning TCP sockets 182
using Inetinfo working sets 183
monitoring security overhead
analyzing data/planning upgrades 227
counting not-found errors 226
measuring with WCAT 223
overview 222
testing security features 223
tracking anonymous/nonanonymous connections 224
monitoring with counters *See* counters
network input/output
bandwidth requirements of servers running IIS 5.0 206
disk input/output 217
monitoring bandwidth 213
monitoring connections 206
monitoring file transfers 207, 211
monitoring remote computers 217
monitoring TCP connections 207, 212
monitoring transmissions rates 207–210
optimizing connections 215
overview 206
optimizing for Web applications (list) 220
overview 175
preventing processor bottlenecks
improving processor usage and performance (list) 204
monitoring activity 196
monitoring connections 198
monitoring server processors 195

monitoring and tuning servers *(continued)*
preventing processor bottlenecks *(continued)*
monitoring threads 201
overview 195
process throttling limits 198
tools
counters for stress testing (table) 239–246
HTTP Monitoring Tool 230
NetStat and NetMon 230
overview 228
PerfMon *See* PerfMon (System Monitor)
Performance Counter Check 230
Process Viewer/Process Explode/Process Monitor/Event Viewer 231
WCAT *See* WCAT (Web Capacity Analysis Tool)
Web Application Stress Tool *See* Web Application Stress Tool
tuning ASP queuing and thread pools 218
MS Remote (RDS disconnected recordset provider) 331
msadc virtual directories, creating 326
MSDAORA (OLE DB provider for Oracle) 331
MSDAOSP (simple OLE DB provider) 331
MSDASQL, OLE DB provider for ODBC 331
MSDataShape provider 328, 331
MSF (Microsoft Solutions Framework) for Infrastructure Deployment *See* migration process management
MSIDXS (provider for Microsoft Indexing Service) 331
MSPersist (persisted recordset provider) 331
MTA Host and NNTP Host NES 3.5 configuration setting 101
multimedia, functionality in Web applications 253
multiple instances of servers, NES 3.5 vs. IIS 99
multiple threads, CGI vs. ASP 560
multiple users, CGI vs. ASP 560
Multiple Virtual Storage (IBM MVS), accessing legacy applications and data *See* legacy applications and data
multiprocessor scaling 150
multitier applications, tiers (list) 249
multitiers
client tiers *See* client tiers
designs 249
middle tiers *See* middle tiers
Munge.asp (sample code) 441
munging URLs 438, 442
MVS (IBM Multiple Virtual Storage), accessing legacy applications and data *See* legacy applications and data

N

named-based virtual hosts, Apache vs. IIS 84
namespaces 275, 302
NameVirtualHost Apache server directive 89
naming conventions, style guide 573
naming home pages, UNIX compatible (list) 76
National Security Agency (NSA) 505
Negotiate header 486
NES (Netscape Enterprise Server) 3.5, migrating to IIS 5.0 *See* migrating from Netscape Enterprise Server 3.5
nested virtual servers, overlapping vs. installing FrontPage 411
Net Use command 230
NetMon (Network Monitor) 213, 230
Netscape Enterprise Server 3.5, migrating to IIS 5.0 *See* migrating from Netscape Enterprise Server 3.5
Netscape Navigator 2.02, posting content 386
NetStat command 230
network adapters 205, 389
network authentication protocols 602
network bandwidth *See* bandwidth
network capacity, measuring 206
network cards, upgrading 205
network connections, optimizing 215
network environment survey 32
Network File Sharing (NFS) 63
network input/output *See* bandwidth, network input/output
Network Interface counters
Bytes Received/sec: Adapter# 209
Bytes Sent/sec: Adapter# 209
Bytes Total/sec: Adapter# 209
Network Layer (OSI) 207, 209
Network Load Balancing
building three-tier Web clusters 448–453
capacity planning issues 152
functionality 389
linking with Cluster Service 388
network logon, Windows 2000 Server 469
Network Monitor (NetMon) 213, 230
network permissions 499
network protocol analyzers, security 463
network servers 144, 500
network sniffing 463, 485
network traffic 222, 457

Network user accounts 498
networks, sample installations 455
New Virtual Directory Wizard 93, 99
NFS (Network File Sharing) 63
<NOEMBED> HTML tag 580
nonanonymous connections
  monitoring counters (table) 225
  tracking with PerfMon 224
Nonanonymous User counters, defined 226
nonpaged memory pools 178
nonpaged pools 178, 182, 199
nonrepudiation, security 463, 594, 603
not-found errors 226
NSA (National Security Agency) 505
NSLookup utility 379
NT Challenge Response vs. integrated Windows authentication 472
NT LAN Manager (NTLM), IIS configurations
  accessing remote resources (table) 501
  vs. integrated Windows authentication 486
ntbugtraq (Windows) 600
NTFS file system
  converting UNIX file names and pathnames 70
  directory security 80
  disk quotas 477
  file sharing with UNIX-based servers 63
  IIS configurations, security 484, 498
  permissions
    determining access 510
    IIS security working with FrontPage 412
    limiting access for FTP sites 74
    managing with FrontPage 413, 421
    setting 78, 115, 332
    vs. Web server permissions 401
  setting DACLs 471
  vs. FAT file systems 69
NTFS Full Control permissions 78
n-tier architecture, middle tiers 259
NTLM (NT LAN Manager), IIS configurations
  accessing remote resources (table) 501
  vs. integrated Windows authentication 486

O

Object Access audit event 479
Object Cache
  analyzing performance 187
  maintaining in working sets 186
  memory requirements 178
  monitoring with counters (table) 187
  retiming to optimize memory usage 194
  working with File System Cache 189
Object counter, Threads 245
object names, style guide 573
Object Properties dialog box 521
object proxies, creating 325
object statements, HTML standards 580
<OBJECT> HTML tag 278, 580, 582, 585
object-cache scavenger, functionality 187
ObjectCacheTTL registry key 188, 194
ObjectContext object (ASP) 267, 361, 363
objects
  See also methods
  ADO Recordset *See* Recordsets
  ADODB.Recordset 322, 328
  ADOR.Recordset 322, 328
  Application (ASP) 267
  CDONTS (IIS) 132
  Command *See* Command object
  Connection (ADO) *See* Connection object (ADO)
  declaring with <OBJECT> tag 582
  declaring with Global.asa 562
  Dictionary (ASP) 302
  Dictionary (VBScript) 583
  Err 268
  Field 351, 355
  FileSystemObject (ASP) 98, 131
  Form 289
  GlobalPageCounter (Global.asa) 278
  IISWebService 283
  IISWebVirtualDir 283
  Job 247
  Location 294
  ObjectContext (ASP) 267, 361, 363
  PageCounter 135
  RDS.DataControl 319–325
  RDS.DataSpace 319–325, 327
  RDSServer.DataFactory 319–324, 327
  Redirect (ASP) *See* Redirect object (ASP)
  Request (ASP) *See* Request object (ASP)
  Response (ASP) *See* Response object (ASP)
  Server (ASP) *See* Server object (ASP)
  Session (ASP) *See* Session object (ASP)
  stateless vs. stateful 363
  TextStream 131
  User 493
  UserCertificate 493
  using
    application scope 581
    out of scope 296
  VBArray (VBScript/JScript) 298
OCI2 SONET, microsoft.com statistics 168
ODBC (Open Database Connectivity) 8, 312, 313, 317

ODBC Connection Manager 336
ODBC connection pooling 335, 583
ODBC drivers 125
ODBC DSN (Data Source Name) 329
ODBC-compliant databases, standards 308
OLAP (online analytical processing) 311
OLE DB 312, 313
OLE DB connection pooling 583
OLE DB connection strings 329
OLE DB format 549
OLE DB provider for AS/400 and VSAM, accessing legacy applications and data
  accessing host files (example) 545–548
  configuring Data Provider 546
  functionality 544
  identifying strategies 532
  using OLE DB and ADO 544
OLE DB providers
  ADO and RDS 314
  database access 125
  IIS architecture 8
  selecting (list) 331
OLE DB–compliant databases, standards 308
OLTP (online transaction processing) 311
On Error Resume Next statement 298
On/Off NES 3.5 configuration setting 101
_OnClick event handler 289
OnClick event 322
OnEnd event handler 276
online analytical processing (OLAP) 311
online assets, creating inventories 589
online transaction processing (OLTP) 311
_OnSubmit event handler 289
OnTransactionAbort event handler 361
OnTransactionCommit event handler 361
Open Database Connectivity (ODBC) *See* ODBC (Open Database Connectivity)
Open method, ADO Connection object 335
Open With FrontPage command 408
optimizing
  database connections (list) 337–342
  memory usage (list) 193
  migrating legacy transaction processes 556
  network connections 215
  performance testing with counters *See* counters
  thread values 203
  upgrading/adding processors 220, 227
  Web applications 220
Option Explicit statement (VBScript) 296, 571, 576
<OPTION> HTML tag 265
Options Indexes access configuration directive 97
Options, ExecCGI access configuration directive 97
Organization Environment Survey (OrgEnvir.doc) 32, 56
OS/400 (IBM), accessing legacy applications and data *See* legacy applications and data
OSI (Open Systems Interconnectivity) layers 207, 476
Outlook, locking in security settings 604
out-of-process application pooling 396
out-of-process applications 119, 431
out-of-process components 282
out-of-process extensions 261
out-of-process isolation 394
out-of-process pools 183
output handling, migrating 128
output logic, returning results 128
overhead
  CGI/ISAPI/ASP performance 112
  ISAPI/ASP/CGI/static HTML pages performance tests 148
  monitoring
    connections performance 199
    security overhead *See* security overhead
  optimizing Web applications (list) 220
  traffic capacity planning 140
overlapping virtual servers vs. installing FrontPage 411
Owner/Creator files, giving NTFS Full Control permissions 78

P

page count components, troubleshooting (example) 522
page counters, reproducing CGI page counters 135
page faults 181, 185
page files, extension standards 564
page groups, using Web Application Stress Tool Sample Script 233
page scope, performance 581
PageCount property, Recordset object 353
PageCounter object 135
paged pools, memory requirements 182
PageSize property, Recordset object 353
paging
  disks 181
  files 181, 193
  through Recordsets with Visual Basic 353
Parameters collection 356

Parse Accept Language Header NES 3.5 configuration setting 105
Parse HTML NES 3.5 configuration setting 105
password-cracking programs 593
PasswordCacheTTL (IISAO) property 525
passwords
  Allow IIS to control password option *See* Allow IIS to control password option
  IBM DB2 552
  migrating 77
  passing through to remote resources 502
pathname delimiters, migrating UNIX Perl scripts 117
pathnames, UNIX
  adhering to Windows conventions 118
  converting file names and pathnames 70
  preserving when migrating 65
  vs. Windows 85
paths
  converting UNIX file names and pathnames 70
  preserving when migrating to IIS 5.0 65
  using MapPath (ASP) 574
PCMCIA-based cryptography cards 505
PCT (Private Communication Technology) protocols, IIS architecture 7
peer-to-peer, transmitting legacy data 533
PerfLog (Performance Data Log Service)
  logging
    bandwidth throttling counters 215
    file transfers counters 211
    TCP connection counters 213
  monitoring
    server processors 195
    threads 202
PerfMon (System Monitor)
  analyzing data
    anonymous/nonanonymous connections/not-found errors 227
    bandwidth throttling 215
    cache reads 192
    connections 200
    File System Cache 191
    file transfers 211
    Object Cache 187
    processor activity 196
    SQL Server/Web servers 247
    TCP connections 213
    threads 202
    transmission rates 210
    working sets 185
  baseline logging 247
PerfMon (System Monitor) *(continued)*
  building three-tier Web clusters, calculating hardware needs 450
  counting not-found errors 226
  displaying WCAT results 392
  functionality 228
  monitoring
    remote computers 217
    Template/Script Engine Caches (list) 219
  monitoring memory
    File System Cache (table) 189
    IIS Object Cache (table) 186
    IIS 5.0 working sets (table) 184
  monitoring network input/output
    anonymous/nonanonymous connections (table) 224
    bandwidth throttling (table) 214
    file transfers (table) 211
    not-found errors (table) 226
    TCP connections (table) 212
    transmission rates at Application Layer (table) 208
    transmission rates at Data Link Layer (table) 209
    transmission rates at Network Layer (table) 209
    transmission rates at Transport Layer (table) 209
    transmission rates overview 207
  monitoring processor bottlenecks
    connections (table) 200
    processor activity (table) 196
    threads (table) 202
  performance counters, functionality 228
  stress testing with Web Application Stress Tool
    both SQL Server/Web servers performance (table) 245
    SQL Server performance (table) 244
    suggestions 238
    Web server performance (table) 239, 246
Perfmon.msc vs. Perfmon.exe 180
performance
  analyzing cache data 192
  applications, pages per second tests (table) 149
  CGI applications vs. ISAPI extensions 111
  components vs. scripts in ASP pages 268
  cost of data access 317
  enhancing
    on SQL Server systems 337
    reliability *See* reliability, enhancing

performance *(continued)*
improving processor usage (list) 204
ISAPI/ASP/CGI/static HTML pages
performance tests 148
migrating legacy transaction processes 556
monitoring
security overhead *See* security overhead
with HTTP Monitoring Tool 433
optimizing memory usage (list) 193
out-of-process application pooling 396
porting or rewriting CGI applications 112
replicating DB2 tables with Host Data Replicator 552
scripting
object and variable initialization (list) 581
Visual Basic applications as DLLs 585
working with connections 583
servers and process pooling 119
testing
applications 284
with counters *See* counters
with Web Application Stress Tool *See* Web Application Stress Tool
tuning ASP queuing and thread pools, IIS 5.0 vs. IIS 4.0 218
tuning tips 221
using PerfMon *See* PerfMon (System Performance Counter)
Web applications, capacity 148
performance boost, disabling 194
Performance Counter Check utility 230
performance counters *See* counters
Performance Data Log Service (PerfLog) *See* PerfLog (Performance Data Log Service)
performance overhead 199
Perl 115, 117, 126
Perl CGI applications, migrating 116
Perl for ISAPI (PerlIS.dll) 115, 119
Perl for Win32 (Perl.exe) 115, 117, 119
Perl MAILPROGRAM scripts 134
Perl scripts 110, 117
Perl.exe (Perl for Win32) 115, 117, 119
PerlEX, migrating Perl CGI applications 116
PerlIS.dll (Perl for ISAPI) 115, 119
PerlScript 115, 352
permissions
Allow/Deny access, Apache vs. IIS 84
changing with ADSI scripts (example) 400
COM/DCOM components security 81
configuring IIS 79
customizing scripts (example) 401
FrontPage Server Extensions *See* FrontPage Server Extensions, permissions
permissions *(continued)*
granting dial-in 378
Log on locally 376
migration testing 48
NTFS Full Control 78
NTFS *See* NTFS file system, permissions
replicating with XCopy 67
setting
based on content (table) 80
DSN 332
for domain Web sites 377
for personal Web sites 379
for script interpreters 115
for user and group accounts 78
on SQL Server 332
troubleshooting 510
Permissions Wizard 2, 79, 500
personal Web sites 379
PhysicalDisk counters
% Disk Time 245
Disk Queue Length 244
PICS (Platform for Internet Content Selection), rating content 445
PidFile Apache server directive 89
pilot testing, migration 46, 51
Ping utility 379
PKCS (Public Key Cryptography Standards) certificate type 491
planning capacity *See* capacity planning
Platform for Internet Content Selection (PICS), rating content 445
platforms, expanding 457
Pmon.exe (Process Monitor) 202, 231
Policy Change audit event 479
pool I/O threads 204
pool threads 178, 201
pooled processes, CGI/ISAPI/ASP 112
pooled-process applications 119
pooling
connections 335
sockets 182
pools, nonpaged *See* nonpaged pools
populating IIS directories 67
Port Apache server directive 89
port numbers, creating unique 72
Portable Operating System Interface (POSIX) environments 117
porting applications
CGI applications 112
CGI to ASP 120
migrating to IIS 5.0 110
UNIX applications 116, 117

POSIX (Portable Operating System Interface) environments 117
POST method (ASP) 287
POST requests 121
POST scripts 232
Primary Document Directory NES 3.5 configuration setting 106
privacy 463, 593, 602
privacy protocols 473, 475
Private Communication Technology (PCT) protocols *See* PCT (Private Communication Technology) protocols
Privilege Use audit event 479
privileges, Windows 2000 Server 468, 470
procedure call names, style guide 573
process accounting 3, 431
Process counters
  % Processor Time 196, 202
  Page Faults/sec: Inetinfo 184, 188
  Page File Bytes: Total 181
  Pool Nonpaged Bytes: Inetinfo 182
  Pool Paged Bytes: Inetinfo 182
  Private Bytes - Inetinfo 242
  Private Bytes: _Total 245
  Thread Count 245
  Thread Count: Inetinfo 202
  Working Set: Inetinfo 184
Process Explode (Pview.exe) 202, 231
process isolation
  developing middle-tier Web applications
    applications and processes 279
    configuring isolated processes 282
    out-of-process components 282
    security considerations 284
    using the metabase 283
  running applications 394
Process Monitor (Pmon.exe) 202, 231
process throttling 3, 198, 431
Process Tracking audit event 479
Process Viewer (Pviewer.exe) 202, 231
process, defined 177
processor bottlenecks *See* bottlenecks, preventing processor bottlenecks
processor caches, upgrading 204
Processor counters
  % Privileged Time 196
  % Processor Time (Total instance) 196
  % Processor Time 196, 246
  % Total Processor Time 245
  % User Time 196
  Interrupts/sec 246
  System Processor Queue Length 246
ProcessorAffinityMask registry key 204
processors
  monitoring security overhead 222
  preventing bottlenecks *See* bottlenecks, preventing processor bottlenecks
  stress testing 238
  upgrading and adding 220, 227
ProcessorThreadMax, IIS 5.0 vs. IIS 4.0 218
production servers
  building three-tier Web clusters 451
  deploying content 383
  recovering from crashes 396
  sample installations 455
program code for IIS, memory requirements 178
programmability, IIS
  architecture 11–16
  new features (list) 3
programmatic server administration, Apache vs. IIS 83
Project Plan Template (ProjPlan.doc) 43, 56
project requirements and risks, migration process management *See* migration process management
ProjSche.mpp (Sample Project Schedule) 44, 56
proof-of-concept testing 38
properties
  AbsolutePage, Recordset object 353
  AccessFlags (IISAO) 525
  AccessSSLFlags (IISAO) 525
  Active Connection 584
  ActiveConnection, Command/Recordset objects 340
  AnonymousPasswordSync (IISAO) 525
  AnonymousUser name (IISAO) 525
  AnonymousUserPass (IISAO) 525
  AspAllowOutOfProcComponents, IISWebService object 283
  AspAllowOutOfProcComponents, IISWebVirtualDir object 283
  AuthFlags (IISAO) 525
  CacheSize, Recordset object 342, 345
  CommandTimeout 350
  ConnectionTimeout 338, 584
  ContentType, Response object 129
  CPTimeout (Connection Pool Timeout) 335, 583
  CpuAppEnabled 431
  CpuCgiEnabled 431
  CursorLocation, Recordset object 322, 347
  CursorType 347
  EOF (End of File), Recordset object 342
  IIsCertMapper (IISAO) 525
  LogonMethod (IISAO) 525
  MaxRecords, Recordset object 353

properties *(continued)*
PageCount, Recordset object 353
PageSize, Recordset object 353
PasswordCacheTTL (IISAO) 525
Provider, Connection object 331
RecordCount 350
Response.IsClientConnected 584
Retry Wait 336
SessionID, Session object 277
Timeout, Session object 277
UNCAuthenticationPassThrough (IISAO) 525
UserCertificate 493
Value 351, 356
properties sheets, configuring servers 75
prototyping, migration 38
Provider property, Connection object 331
proxies, business objects and RDS.DataSpace object 325
Proxy Cache Parameters Apache server directive 89
Proxy cache, enabling content expiration 444
proxy servers, security 604
ProxyRequests Apache server directive 89
public key cryptography 473
Public Key Cryptography Standards (PKCS) certificate type 491
Public variables vs. Dim variables 581
public-key certificate authentication 602
Publish command (FrontPage) 409
publishing
setting up 378
Web databases *See* Web databases
Web sites 17
Publishing Service, debugging 304
Pview.exe (Process Explode) 202, 231
Pviewer.exe (Process Viewer) 202, 231
Python scripts, IIS support 110

Q

queries
avoiding timeouts 350
improving performance 358
Query Handler NES 3.5 configuration setting 102
Query method, RDS.DataFactory object 322
Query_String environment variable 123
QueryString collection 123
Querystring editor, stress testing 239
queues
lengthening connection queues 216
monitoring security overhead 222

R

RAM *See* memory
rcp daemon, replicating UNIX files 69
RDO (Remote Data Objects) vs. RDS 316
RDS (Remote Data Services)
Client Cursor Engine 324
client tiers 319
client-side data access
designing custom business objects 327
IIS 326
middle-tier elements 326
overview 318
RDS Data Factory and custom business objects 327
Data Cache 324
databases 314
IIS architecture 8
partitioning data for ADO 322
RDS disconnected recordset providers 331
RDS.DataControl object 319–325
RDS.DataSpace object 319–325, 327
RDSServer.DataFactory object 319–324, 327
Read (RX) directory 499
read aheads, boosting performance 189
Read permissions
customizing scripts (example) 402
setting for
domain Web sites 377
personal Web sites 379
read requests, analyzing data 215
Read Script only permissions, customizing scripts (example) 402
ReadmeName Apache resource configuration directive 94
real-time information 311
Real-Time Sampling Service 230
Recalculate Web command (FrontPage) 408, 409
Record Domain Names/IP Addresses NES 3.5 configuration setting 103
RecordCount property 350
records, updating online with ASP 560
recordset providers (list) 331
Recordset.Open method 343, 350
Recordsets
client-side data access
data-aware controls 320
designing custom business objects 327
IIS 326
middle-tier elements 326

Recordsets *(continued)*
client-side data access *(continued)*
overview 319
RDS Data Factory and custom business objects 327
client-side Data Cache 322
creating 331
cursors *See* cursors
dynamically creating connections 335
increasing size of cache 342
managing records
avoiding query timeouts 350
filling list boxes with VBScript (example) 351
filling tables with PerlScript (example) 352
keeping track of new records 348
limiting number of records 353
overview 348
paging with Visual Basic (example) 353
purging deleted records 350
references to field values 350
retrieving image data 355
sharing connections 340
Redim statement 582
Redirect Apache resource configuration directive 94
Redirect method calls 536
Redirect method, Response object 292
Redirect object (ASP), load balancing 271
Redirect.asp (sample code) 442
redirection
client-side 294
during Session_OnStart 295
factoring applications 286
functionality 292
specifying from Web sites 73
RedirectMatch/AliasMatch command (Apache) 85
RedirectPermanent Apache resource configuration directive 94
redirects 85, 442
RedirectTemp Apache resource configuration directive 94
references *See* resources
Reformat method calls 536
Refresh method, Command object 356
refresh process, capacity planning issues 147
Regedt32.exe vs. Regedit.exe 68
registry, editing 68, 188
Registry Editor, Inetinfo.exe subkey 304
registry editors, administering older version servers 426
registry keys
ADCLaunch 325
AddOnServices 426
DisableBacklogMonitor 221
Image File Execution Options 304
Listen_Back_Log 216
MaxPoolThreads 203
ObjectCacheTTL 188, 194
ODBC connection pooling 336
ProcessorAffinityMask 204
UsePoolThreadForCGI 204
registry settings, editing incorrectly 182
Release milestone 45
reliability
capacity planning issues 150
data publishing considerations 309
enhancing
clustering 388
overview 382
replicating with Content Deployment 383
replication and clustering in IIS 383
grouping load balancing features 448
hosting applications 396
out-of-process application pooling 396
recovering from crashes 396
running applications in isolation 394
testing applications 390
remote access 425
remote administration 404–407
remote computers 217
Remote Data Objects (RDO) vs. RDS 316
Remote Data Service (RDS) *See* RDS (Remote Data Service)
remote resources
authentication protocol for accessing (table) 501
passing through user credentials 503
Remote_User environment variable 133
Remove Server Extensions command (FrontPage) 408
replicating
and clustering in IIS 383
applications for migration 74
configuration settings 109
IIS Web servers 107
legacy databases 548–552
UNIX-based files 69–71
Web and FTP site content 67
Windows-based files 67
Requery method, Recordset object 348
Request object (ASP)
accessing form input 123
building ASP applications 276

Request object (ASP) *(continued)*
functionality 267
QueryString collection 123
ServerVariables collection 124
using with collections 288
Request Queue, IIS 5.0 vs. IIS 4.0 218
Request.Form collection 123, 287
Request.QueryString collection 287
requests
analyzing bandwidth throttling data 215
redirecting 292
requirements definition, creating 26
resource configuration directives, Apache
srm.conf directives and IIS properties
(table) 92–95
resource consumption, measuring 432
Resource Kit companion CD
IIS Migration Wizard 57, 135
migration management, support documents
(list) 56
monitoring and tuning servers tools 246
test tools (list) 57
resource logging 428
ResourceConfig Apache resource configuration
directive 95
resources
books
ASP best practices 586
capacity planning 174
data access and transactions 372
developing Web applications 306
encryption 529
general administration 57
general networking 529
legacy applications and data 558
migrating Web servers to IIS 136
migration and integration management 57
monitoring and tuning servers 246
security 527
site security planning 609
Windows NT security 529
migration process management, assessing
needs 38
Web links
administering ISP installations 460
application migration tools 136
ASP best practices 586
capacity planning 174
data access and transactions 371
developing Web applications 305
IIS and products running on Windows 2000
Server 19
legacy applications and data 558
resources *(continued)*
Web links *(continued)*
migration tools 57
monitoring and tuning servers 246
security 136
server administration and interoperation
tools 136
site security planning 608
Response object (ASP)
functionality 267
handling output 129
load balancing 271
methods available during Session_OnStart
event 295
redirection 292
Response statements 300
Response.AppendToLog method, Response
object 300
Response.Buffer, Response object 292
Response.Clear, Response object 292
Response.End, Response object 293
Response.IsClientConnected property 584
Response.Redirect, Response object 292
Response.Write method, Response
object 276, 296
Restore Configuration NES 3.5 configuration
setting 101
Restrict Access NES 3.5 configuration
setting 101
Resync method 342, 344, 349
retiming Object Cache 194
Retry Wait property 336
REXX scripts, IIS support 110
RFC-822 e-mail messages 132
Risk Assessment Form (RiskAsmt.doc) 28, 56
risks, assessing (table) 28
RlimitCPU Apache server directive 89
root directories, ASP best practices 561
Rotate Log NES 3.5 configuration setting 103
round-robin distribution, capacity 152
routines, Window_OnLoad 322
run command (WCAT) 223
RUNAT=SERVER HTML attribute 263, 278
run-time error, using Script Debugger 6

S

S/MIME (Secure/Multipurpose Internet Mail
Extensions), issuing certificates 491
SAMBA 63, 69
sample installations, adapting IIS
administrative networks 456
advantages of topology 456

sample installations, adapting IIS *(continued)*
back-end networks 456
expanding platforms 457
front-end networks 455
overview 453
querying SQL Server 457
security 458
several business clients 454
three networks in one 455
Sample Project Schedule (ProjSche.mpp) 44, 56
Sample Script (Web Application Stress Tool) 232
sample setup, components for administration 440
Sample Test Plan (TestPlan.doc) 47, 56
scalability
building three-tier Web clusters *See* Web clusters, building three tiers
data publishing considerations 309
running CGI applications in isolation 395
transaction processing on Web 359, 367
scaling multiple processors 150
scoping variables, performance 581
ScoreBoardFile Apache server directive 90
screens, HTML standards for small 580
script components
COM components *See* COM (Component Object Model)
Windows Script Components 269
Script Debugger 6, 299
Script Engine Cache
caching scripts in ASP pages 186
memory requirements 178
monitoring with PerfMon counters 219
tuning ASP queuing and thread pools, IIS 5.0 vs. IIS 4.0 218
script files, running Script.scr (WCAT) 392
script interpreters 66, 114
Script.cfg (WCAT configuration file) 392
Script.dst (WCAT distribution file) 392
Script.scr (WCAT script file) 392
<SCRIPT> HTML tag 263, 276, 278, 301
ScriptAlias Apache resource configuration directive 95
scripting
common mistakes (list) 296–298
performance
object and variable initialization (list) 581
Visual Basic applications as DLLs 585
working with connections 583
scripting delimiters, <%= expression %> 262
scripting engines 66, 397
scripting languages, errors 297
Scripting.FileSystemObject 300
script map (IIS) 261
scripts
See also ASP (Active Server Pages), scripts
automating administration with ADSI scripts
changing permissions (example) 400
creating Web sites (example) 400
customizing scripts (example) 401
executing scripts 398
built-in and customized 10
debugging ASP 298–301
IIS support 110
layout order, style guide 571
server-side, security 601
style guide 566, 574
using
client-side in Web applications 253
Web Application Stress Tool Sample Script 232
viruses and bugs 423
Secure Communications dialog box 504
Secure Sockets Layer (SSL) protocols *See* SSL (Secure Sockets Layer) protocols
Secure/Multipurpose Internet Mail Extensions (S/MIME) 491
security
Access databases, IIS limitations 81
additional resources 527
Apache vs. IIS 84
assigning relative costs (example) 596
auditing access with IIS logs 524
certificates *See* certificates
COM/DCOM components 81
configuring servers 77–80
core components (list) 462
cost considerations 466
creating inventories 589
data protection with Component Services 308
defending against malicious attacks 522
estimating potential asset damages and security costs 589
expanding platforms 458
FAT file systems 69
formatting disk partitions 63
FTP sites, limiting access 74
hidden form fields risks 291
IIS Admin Objects/ADSI security settings (list) 524
integrating UNIX and Windows 2000 Server security 82
Kerberos v5 authentication protocol *See* Kerberos v5 authentication protocol
locking in Internet Explorer and Outlook settings 604

security *(continued)*
migrating certificates 82
migration process management, testing 49
NTFS/IIS directory security (table) 80
permissions *See* permissions
process isolation risks 284
replicating DB2 tables with Host Data Replicator 552
session IDs cookie risks 274
SQL Server 332
SSL, capacity planning issues 147
threats/vulnerabilities/attacks 464
troubleshooting (example) 515–522
viruses and bugs 423
Web links 136
Windows file systems 69
Security Identifiers (SIDs) 468, 472
security overhead
monitoring
analyzing data/planning upgrades 227
counting not-found errors 226
measuring with WCAT 223
overview 222
testing security features 223
tracking anonymous/nonanonymous connections
counters (table) 225
using PerfMon 224
security tokens, caching 495
security, IIS configurations
authentication modes
Anonymous Web authentication 481
Basic authentication 484
Certificate Services 491
client certificate mapping 489
Digest authentication 487
FTP authentication 496
functionality 481
integrated Windows authentication 486
Web authentication summary (table) 495
Web/FTP authentication (list) 480
determining access
custom authentication 510
IIS access control 510
IP address access control 509
NTFS permissions 510
overview 508
user access control 509
extending with ISAPI filters/ASP code 496
files and directories 498
IIS security checklist 525
new features (list) 2
security, IIS configurations *(continued)*
overview 480
secure communications with SSL and Transport Layer Security 503–507
security initialization checklist 605
troubleshooting permissions 510
virtual directories 499
working with FrontPage 412
security, site planning
additional resources 608
adopting technologies and standards
client security management 603
firewalls 604
server security initialization checklist 605
server-side 602
assessing threats
attacker motives and targets 589
identifying 588
impact 592
on intranets 590
over Internet 592
overview 588
balancing needs 595–598
making policy 599
overview 587
security, Windows 2000 Server configurations
auditing 478
authentication
authentication methods 472
DACLs 471
impersonation 471
LocalSystem account 470
logon and token types 468
overview 467
privileges and attack prevention 470
services and service security 470
SIDs 468
authorization 472
availability
Clustering Service 478
NTFS disk quotas 477
built-in features (list) 467
IIS security checklist 525
privacy and integrity mechanisms 473
security initialization checklist 605
Select Case statement 574
<SELECT> HTML tag 265, 351
sendmail mail server 132
sequential reads 189
server administration, Web links 136
server certificates, security 602
server configuration directives, Apache httpd.conf directives and IIS properties (table) 86–91

Server counters
Logon/sec 202
Total Bytes/sec 242
server logging, NES 3.5 configuration settings and IIS properties (table) 102
Server Manager (NES) 99
Server Message Block (SMB) protocol, file sharing 63
Server Name NES 3.5 configuration setting 101
Server object (ASP) 267, 276
Server Port NES 3.5 configuration setting 101
server preferences, NES 3.5 configuration settings and IIS properties (table) 100
server process overhead, performance 112
server security initialization checklist 605
Server Side JavaScript NES 3.5 configuration setting 102
Server status reports access configuration directive 97
server variables, performance 582
Server.CreateObject method (ASP) 278, 282, 305
Server.CreateObject() 582, 585
Server.MapPath method (ASP) 574
Server.UrlEncode method (ASP) 287
Server_Name environment variable 124, 131
ServerAdmin Apache server directive 90
ServerAlias Apache server directive 90
Server-Gated Cryptography (SGC) 2
ServerName Apache server directive 90
ServerPath Apache server directive 90
ServerRoot Apache server directive 90
servers
See also Web servers
analyzing File System Cache memory 191
capacity planning for networks 144
changing properties 194
clustering, capacity planning issues 150
configuration 75
considering load demands with databases 309
hosting multiple Web sites 99
memory capacity planning 174
monitoring and tuning servers *See* monitoring and tuning servers
performance and process pooling 119
programmatic administration, Apache vs. IIS 83
Windows 2000 Server *See* Windows 2000 Server
server-side cursors 348
server-side includes 73
server-side includes, NTFS/IIS directory security 80
server-side scripting delimiters 262
server-side scripts
combining with HTML 128
manipulating client-side objects 265
planning security 601
<SCRIPT> tags vs. delimiters 263
scripts in ASP pages 262
server-side, defined 249
ServerSupportFunction (HSE_REQ_TRANSMIT_FILE) function 148
ServerType Apache server directive 90
ServerVariables collection 124, 288
service environment, migration 32, 37
service security, Windows 2000 Server 470
Session collection 127, 302
session cookie, maintaining state 127
Session events, debugging 300
session IDs (ASP) 272
Session object (ASP)
attempting access 265
building ASP applications
Application and Session events 276
application boundaries 275
ending ASP sessions 277
Global.asa 276, 278
overview 274
closing connections 339
enhancing performance 338
functionality 267
GET vs. POST 288
management
security risks 274
using cookies 270
using session IDs 272
overview 269
paging through Recordsets 354
restricting connections across pages 341
selecting object scope 279
testing applications 284
using
dictionaries to partition namespaces 302
hidden form fields 291
out of scope 296
session scope 562, 581
Session variables 277, 350
Session.Abandon method, Session object (ASP) 276
Session.Timeout, Session object (ASP) 284
Session_OnEnd event 276
Session_OnStart event 271, 276, 295
SessionID property, Session object (ASP) 277
SetAbort method, ObjectContext object 361
SetComplete method, ObjectContext object 361

Set-Cookie HTTP header 270
Setup Wizard 62
SGC (Server-Gated Cryptography) 2
share information, preserving 68
ShellCGI Directory NES 3.5 configuration setting 102
SIDs (Security Identifiers) 468, 472
Simple Management Network Protocol (SMNP) Sub-Agent Configuration NES 3.5 configuration setting 103
small screens, HTML standards 580
smart-card authentication, security 602
SMB (Server Message Block) protocol, file sharing 63
SMTP Service counters
  analyzing data 201
  Bytes Received/sec 208
  Bytes Sent/sec 208
  Bytes Total/sec 208
  Messages Received Total 211
  Messages Sent Total 211
SMTP service, sending e-mail messages 132
SNA (Systems Network Architecture) Server 4.0 SP2, accessing legacy applications and data
  connecting to 533
  developing and deploying applications 534
  overview 531
  using COMTI 536
SNA Distributed Data Management (DDM), accessing legacy applications and data 544
SNA environment, transmitting legacy data peer-to-peer 533
SNAOLEDB (OLE DB provider for SNA Server) 331
sniffing networks, IPSec 475
socket pooling, disabling 182, 430
software
  microsoft.com case study 168, 171
  migration considerations 40
Software Virtual Servers (URL Host, Home Page) NES 3.5 configuration setting 106
software virtual servers, NES vs. IIS 99
source servers 63, 68
Special (R) directory 499
Special (RW) directory 499
Special (X) directory 499
special characters, migrating UNIX Perl scripts 117
spikes 169, 217, 221
spoofing 475, 593
SQL Reporting Server, HTTP Monitoring Tool component 230
SQL Server
  clustering features 388
  database access 125
  enhancing performance 337
  importing/exporting/transforming legacy data 549
  monitoring performance during stress testing 244, 245, 247
  querying (sample installations) 457
  security 332
  Web databases *See* data access
SQL Server Connection Manager 337
SQLOLEDB (SQL Server OLE DB data provider) 331
SQLServer - Locks counter, Total Blocking Locks 244
SQLServer counters
  Cache Hit Ratio 244
  I/O—Lazy Writes/sec 244
  Net—Network Reads/sec 244
  User Connections 244
srm.conf Apache directives, IIS properties (table) 92–95
SSL (Secure Sockets Layer)
  capacity planning issues 147
  IIS configurations, Basic authentiction 485
  issuing digital signatures 491
  secure communications
    authentication and trust 503
    certificates and CryptoAPI 506
    enabling on IIS 504
    encryption 504
    IIS and Fortezza 505
    overview 503
  setting properties 525
  Windows 2000 Server 474–476
SSL protocol 7, 222, 227
ssl.testname, security overhead 223
staging servers 383, 451, 455
standards
  migration process management (table) 35
  server-side security planning 602
Standards Review Form (Standard.doc) 35, 56
StartServers Apache server directive 91
state management, migrating from CGI to ASP 127
stateless objects vs. stateful objects 363
statements
  <!-- #include --!> 514
  Dim 296, 581, 582
  embed 580
  End If 298
  If 297

statements *(continued)*
object 580
On Error Resume Next 298
Option Explicit (VBScript) 296, 571, 576
Redim 582
Response 300
Select Case 574
style guide 575
While 297
Write 300
static content, security 80
static cursors vs. dynamic cursors 345
static files, performance tests (table) 149
static HTML pages
analyzing bandwidth throttling data 215
customizing HTML footers 446
limiting bandwidth 214
optimizing (list) 220
vs.
dynamic pages 142, 309, 317
ISAPI/ASP/CGI performance tests 148
.stm files, server-side includes 73
stored procedures, defined 356
stress testing
PerfMon *See* PerfMon (System Monitor)
WCAT *See* WCAT (Web Capacity Analysis Tool)
Web Application Stress Tool *See* Web Application Stress Tool
string concatenation, style guide 575
string functions, style guide 576
structure Web directories 65
subauthentication DLL 483, 501
Submit method, Form object 289
subroutines 263, 581
sub-webs, resetting permissions 417
support, additional resources *See* resources
symbolic links, Windows file systems 69
symmetric key cryptography 473
System audit event 479
System counters
% Total Processor Time 238
Context Switches/sec 246
Processor Queue Length 196
Total Interrupts/sec 245
System Monitor (PerfMon) *See* PerfMon (System Monitor)
system operations standards, migration process management (table) 35
Systems Network Architecture (SNA) *See* SNA (Systems Network Architecture) Server 4.0 SP2

T

tab characters, migrating UNIX Perl scripts 117
tables, filling with PerlScript (example) 352
TakeANumber component, installing 273
Tar.exe, replicating UNIX files 69
target servers, preparing for migration 63
Task Manager 195
TCB (Transmission Control Block) tables 178, 199
TCI scripts, IIS support 110
TCP (Transmission Control Protocol) connections
analyzing counter data 213
measuring 207
monitoring counters (table) 212
traffic issues 140
TCP checksum offloading 205
TCP counters
Connections Established 212
Connections Failures 212
Connections Reset 212
Segments Received/sec 209
Segments Retransmitted/sec 209
Segments Sent/sec 209
Segments/sec 209
TCP sockets 178, 182
TCP/IP connections 201, 333
TCP/IP Keep-Alives vs. HTTP Keep-Alives 199
TCP/IP networks, security 588
TCP/IP overhead 140
TCP/IP Telnet clients 406
TCP/IP Telnet Server 406
teams, migration process management
assessing resource needs 38
Deploying phase (table) 53
Developing phase (table) 46
Envisioning phase (table) 29
Planning phase (table) 32
Technical Environment Survey (TechEnvi.doc) 32, 56
Telnet Service 406
templates
Design Template (Design.doc) 37, 56
Functional Specification Template (FuncSpec.doc) 36, 56
IIS Template Cache *See* IIS Template Cache
Project Plan Template (ProjPlan.doc) 43, 56

templates *(continued)*
security 523
Vision and Scope Template (VisScope.doc) 56
Terminal Services 3, 405
Test group accounts 78
test labs *See* testing
testing
applications
pages per second performance tests (table) 149
performance 284
building test labs 154
counters *See* counters
migration process management
application tests (table) 50
integration tests (table) 49
pilot testing 51
test/unit/integration labs 40
testing at microsoft.com 47
tools (list) 57
unit tests (table) 48
unit/integration/application/pilot testing 46
proof-of-concept testing 38
using
PerfMon *See* PerfMon (System Monitor)
WCAT *See* WCAT (Web Capacity Analysis Tool)
Web Application Stress Tool *See* Web Application Stress Tool
TestPlan.doc (Sample Test Plan) 47, 56
text file line termination, UNIX feature 118
text-only browsing, HTML standards 578
TextStream object 131
/Themes subdirectory of /Content directory 563
thin clients, administering sites remotely 405
third tiers, defined 249
Thread counters
% Processor Time: Inetinfo => Thread # 202
Context Switches/sec: Inetinfo => Thread # 202
thread management
CGI vs. ISAPI on Windows 2000 260
managing ASP 127
thread pools, IIS 5.0 vs. IIS 4.0 218
threading models, performance 154, 338
thread-per-client vs. worker threads 201
threads
adjusting for bottlenecks 205
analyzing performance data 202
CGI-related issues 204
defined 177
threads *(continued)*
disabling backlog monitors 221
monitoring counters (table) 202
multiple, CGI vs. ASP 560
optimizing values 203
threat assessment planning
assigning potential damage (example) 595
identifying
attacker motives and targets 589
impact 592
on intranets 590
over Internet 592
overview 588
threats
defined 464
from inside organizations 590
throttled sites, disabling pooling 182
throttling
bandwidth *See* bandwidth throttling
process *See* process throttling
tiers
applications, migration considerations 113
client tiers *See* client tiers
middle tiers *See* middle tiers
multitier designs 249
three-tier legacy applications 555
three-tier programming models, migrating legacy transaction processes 554
three-tier Web clusters *See* Web clusters, building three tiers
time, browser download 143
Timeout Apache server directive 91
Timeout property, Session object 277
timeouts 338, 350
time-sensitive content, expiring 73
timestamps, capacity planning issues 147
TLS (Transport Layer Security)
issuing digital signatures 491
secure communications 503
Windows 2000 Server 474–476
token types, Window 2000 Server 468
tokens, caching 495
tools
debugging IIS 303
monitoring and tuning servers
counters for stress testing (table) 239–246
HTTP Monitoring Tool 230
NetStat and NetMon 230
overview 228
PerfMon *See* PerfMon (System Monitor)
Performance Counter Check 230

tools *(continued)*
monitoring and tuning servers *(continued)*
Process Viewer/Process Explode/Process Monitor/Event Viewer 231
WCAT *See* WCAT (Web Capacity Analysis Tool)
Web Application Stress Tool *See* Web Application Stress Tool
Tools Checklist (ToolsChk.doc) 33, 56
topology, sample installations 456
TPC/IP protocol stack, SSL vs. Transport Layer Security vs. IPSec 476
TPS (transactions per second), ADO vs. ODBC (table) 317
tracerouting, finding bottlenecks 154
traffic, capacity planning issues 140–142
transaction processing programs, legacy
See also legacy applications and data
extending transactions with COMTI (example) 540–542
integrating IIS and applications 534
migrating to Component Services 553, 557
troubleshooting mainframe 540
using COMTI 535
transactional components, troubleshooting (example) 519
transactional Web sites, capacity 161
transactions
See also Component Services
introducing Message Queuing 368
managing ASP 127
migrating legacy transaction processes *See* migrating legacy transaction processes
processing on Web
business objects vs. scripts in ASP pages 362
Component Services benefits (list) 358
extending limits 360
transactional ASP 361
transactions explained 359
transactional components
business logic in components 362
distribution and scaling issues 367
overview 362
participating in transactions 364
using database access interfaces 366
transactions per second (TPS), ADO vs. ODBC (table) 317
TransferLog Apache server directive 91
Transmission Control Block (TCB) table 178, 199
Transmission Control Protocol (TCP)
connections *See* TCP (Transmission Control Protocol)
transmission rates
analyzing counter data 210
counters measuring
Application Layer (table) 208
Data Link Layer (table) 209
Network Layer (table) 209
Transport Layer (table) 209
overview 207
transmission requests, analyzing data 215
TransmitFile, ISAPI vs. ASP/CGI 148
Transport Layer (OSI) 207, 209
Transport Layer Security (TLS) *See* TLS (Transport Layer Security)
troubleshooting
additional resources *See* resources
creating company domain Web sites 379
mainframe transaction processing programs 540
permissions 510
resecuring sites 412
security (example)
authenticating users 519
calling ISAPI authentication filter 518
data access components written in Visual Basic 6.0 520
loading Logon.asp 519
logging on user 517
overview 515
performing lookup with ADO 520
updating page counts 522
WCAT 393
trusted for delegation, IIS configurations 501
.txt file extension standards 564
type libraries, Global.asa 562
type library constants, Global.asa 278
TypesConfig Apache resource configuration directive 95

U

UBound method, VBArray object (VBScript/JScript) 298
UCase() 576
UNCAuthenticationPassThrough (IISAO) property 525
UNC-shared (Uniform Naming Convention) Web content, passing through to 502
unique port numbers, creating 72

unit test labs, migration process management
defined 46
development and test environment 40
unit tests (table) 48
UNIX application tools, compatibility 116
UNIX applications
migrating
consideratons 116
UNIX Perl scripts 117
vs. Windows file descriptors/conventions (table) 118
UNIX end-point servers, deployment 384, 387
UNIX file names, converting 70
UNIX mail scripts, migrating 134
UNIX Perl scripts, migrating 117
UNIX systems, bugtraq 600
UNIX tools, migrating 66
UNIX utilities, migrating 66
UNIX-based files, replicating 69–71
UNIX-based servers
administering sites remotely 406
file sharing in mixed environments 63
formatting disk partitions 63
integrating Windows 2000 Server security 82
making home page names compatible (list) 76
migrating to IIS 5.0 *See* migrating Web servers to IIS 5.0
porting CGI applications to IIS 114
vs. Windows 2000 Server logging 75
UNIX-style pathnames *See* pathnames
Unlock method, Application object 279
unprocessed input, accessing 123
updating content dynamically 73
upgrading
IIS Web servers 107
to IIS 5.0 *See* migration process management
uploading content through FTP 424
URL Host NES 3.5 configuration setting 106
URL munging 438, 442
URL paths, one computer/same IP address/multiple hosts 437
URL Prefix NES 3.5 configuration setting 102, 104
URL Prefix, Forward Requests To NES 3.5 configuration setting 106
UsePoolThreadForCGI registry key 204
user accounts 77, 82, 407
user credentials, passing through to remote resources 503
user directories, Apache vs. IIS 84
User object 493
user Web sites
Apache vs. IIS 84
options for implementing 72
User/Group Apache server directive 91
UserCertificate object 493
UserCertificate property 493
UserDir Apache resource configuration directive 95
users, ISP diagram 34
utilities
Cgi.pm (Perl) 121, 128
Cgi-lib.pl (Perl) 121
Fpremadm 408, 409
Fpsrvadm 408, 409
Fpsrvadm.exe command-line 414
Fpsrvwin 408
IIS Administration Script 79
Iissync 109
NSLookup 379
Ping 379
XCopy 67, 109

V

Value property 351, 356
variable case values, style guide 576
variable declarations, style guide 576
variable names, style guide 576
variable value trimming, style guide 578
variables
application performance 284
frequent use in applications 581
mixing data types 297
performance, initializing 581
Public vs. Dim 581
scripting run-time errors 296
server performance 582
using out of scope 296
VB (Visual Basic) *See* Visual Basic
VBArray object (VBScript/JScript) 298
VBScript (Visual Basic Scripting Edition)
filling list boxes (example) 351
IIS 8, 110
scripting for performance 581, 583
style guide for scripts in ASP pages (list) 565
vs. DLLs 585
VBScript scripting engines 397
View Access Log NES 3.5 configuration setting 103
View Error Log NES 3.5 configuration setting 103
vigilance, security planning 600

virtual directories
application roots 275
IIS configurations
access and content control options (list) 499
accessing resources on network servers 500
passing through to UNC-shared Web content 502
trusting for delegation 501
using Permissions Wizard 500
IIS snap-in vs. Windows Explorer 76
virtual hosts vs. virtual servers 84
vs.
aliases/directory aliases 85
NES 3.5 directory aliases 99
Virtual Directory, publishing content 65
Virtual FTP Server, limiting access 74
Virtual Host, Apache vs. IIS 84
Virtual Sequential Access Method (VSAM) *See* OLE DB provider for AS/400 and VSAM
virtual servers
Apache vs. IIS 84
overlapping vs. installing FrontPage 411
resetting permissions 419
vs. NES 3.5 hardware servers 98
virtual user directories, Apache vs. IIS 84
<VirtualHost> Apache server directive 91
viruses, uploading executable scripts and programs 423
Vision and Scope documents, migration process management
assessing risk 28
creating requirements definition 26
creating 24
defining projects 25
developing conceptual designs 27
Envisioning phase 29
Vision and Scope Template (VisScope.doc) 56
Visual Basic
data access components, troubleshooting (example) 520
paging through Recordsets (example) 353
Visual Basic applications, VBScript vs. DLLs 585
Visual Basic code vs. ASP 114
Visual InterDev 562
VSAM (Virtual Sequential Access Method) *See* OLE DB provider for AS/400 and VSAM
vulnerabilities, defined 464

W

W3C Extended Logging Format, UNIX vs. Windows 2000 Server 75
WAI Management NES 3.5 configuration setting 102
WAM (Web Application Manager) 280
WCAT (Web Capacity Analysis Tool)
functionality 231
monitoring
security overhead 223
server processors 195
reading performance counters 228
testing applications
checking performance counters 392
overview 391
performance testing 393
stress testing 218, 394
testing conditions 393
troubleshooting 393
WCAT scripts, running (table) 392
Web Application Manager (WAM) 280
Web Application Stress Tool
applications, pages-per-second tests (table) 149
building three-tier Web clusters 450
general guidelines for using 238
installing 232
monitoring server processors 195
overview 231
reading performance counters 228
stress-testing counters
both SQL Server/Web servers performance (table) 245
SQL Server performance (table) 244
Web server performance (table) 239, 246
testing Web applications 218, 390
using Sample Script
clients 236
overview 232
page groups 233
performance counters 234
reporting 236
running tests 236
Script view/script items/Script Item Detail view 232

Web Application Stress Tool *(continued)*
using Sample Script *(continued)*
settings 235
users 235
Web applications
additional resources 305
building on client/server 248–252
client-side technologies *See* client tiers, developing Web applications
debugging *See* debugging
design patterns
client-side redirection 294
factoring applications 285
GET vs. POST 287
hidden form fields 291
overview 285
redirecting 292, 295
using forms for input 286
validating client-side forms 289
developing 247
enhancing reliablity 394
IIS 5.0 vs. IIS 4.0 218
middle tiers *See* ASP (Active Server Pages), scripts; ASP applications, developing middle-tier Web applications; middle tiers, developing Web applications
migrating Web servers to IIS 5.0 110–114
monitoring and tuning servers *See* monitoring and tuning servers, Web applications
performance and capacity planning 148
process isolation *See* process isolation
testing with
WCAT *See* WCAT (Web Capacity Analysis Tool)
Web Application Stress Tool *See* Web Application Stress Tool
Web authentication summary (table) 495
Web browsers *See* browsers
Web clusters
building three tiers
calculating hardware needs 450
creating first tiers 450
creating second tiers 452
creating third tiers 452
defining clustering 447
defining load balancing 447
grouping load balancing features 448
overview 446, 448
reviewing the three tiers 453
capacity planning issues 150
enhancing reliability 382
planning future expansion 458
Web clusters *(continued)*
replication and clustering in IIS
clustering 388
Content Deployment servers 384
fault tolerance and load balancing 389
new features (list) 385
overview 383
replicating with Content Deployment 383
type of projects 386
using ASP Session 271
Web databases
benefits (list) 308
cost of data access 317
data publishing considerations (list) 309
database-centric publishing 310
MDAC
ADO and RDS 314
ODBC and OLE DB 313
overview 312
other data access methods
ADC 315
Jet and DAO 316
RDO 316
overview 308
vs. pages for dynamic content 308
Web Distributed Authoring and Versioning (WebDAV) 4, 17
Web links *See* resources, Web links
Web logging, troubleshooting 510
Web pages, Indexing Service architecture 7
Web publishing *See* Web databases
Web publishing NES 3.5 configuration setting 106
Web Publishing Wizard 386
Web Server Certificate Wizard 2, 82, 504
Web server certificates, backing up 507
Web server directory structures 65
Web server permissions, customizing scripts (example) 401
Web server root directories 65
Web servers
See also servers
administering older version servers 426
IIS architecture 5
installing clusters *See* Web clusters
migrating to IIS 5.0 *See* migrating Web servers to IIS 5.0
migration process management *See* migration process management

Web servers *(continued)*
monitoring
and tuning servers *See* monitoring and tuning servers
performance during stress testing (table) 239, 245, 246
troubleshooting permissions 513
Web Service counters
Anonymous Users/sec 225
Bytes Received/sec 208
Bytes Sent/sec 208
Bytes Total/sec 208, 239, 246
Bytes/sec threads 202
Connection Attempts/sec 238
Current Anonymous Users 225
Current Connections 200, 202, 240
Current NonAnonymous Users 225, 240
Files Received 211
Files Sent 211
Files Total 211
Maximum Anonymous Users 225
Maximum Connections 200, 239
Maximum NonAnonymous Users 225
NonAnonymous Users/sec 225
Not Found Errors 240
Not Found Errors/sec 226
Total Anonymous Users 225
Total NonAnonymous Users 225
Web site Operators user accounts 407
Web sites
backing up content when migrating 109
capacity planning scenarios
Internet commerce Web sites 163
Internet marketing Web sites 159
Internet transactional Web sites 161
intranet Web sites 157
changing roots 410
creating
company domain Web sites 374–379
overview 374
personal Web sites 379
troubleshooting 379
with ADSI scripts (example) 400
hosting multiple 99
implementing special features (list) 73
IP addresses 434
measuring resource consumption 432
microsoft.com case study 166–174
publishing
using FrontPage Server Extensions 18
using FTP 19
using WebDAV 17
redesigning to prevent bottlenecks 205
Web sites *(continued)*
replicating content 67
restricting content 381
setting IIS permissions 79
users
Apache vs. IIS 84
options for implementing 72
Web Application Stress Tool *See* Web Application Stress Tool
Web structures, replicating Web sites 68
WebDAV (Web Distributed Authoring and Versioning) 4, 17
While loop 298
While statement 297
Win32 environments, porting UNIX applications 117
Win32-compatible script interpreters 66
WinCGI Directory NES 3.5 configuration setting 102
WinCGI, porting limitations 114
Window_OnLoad event 265, 352
Window_OnLoad routine 322
Windows 2000 Advanced Server
clustering features (list) 388
Network Load Balancing capacity issues 152
Windows 2000 Cache Manager 189, 191
Windows 2000 Components Wizard 230
Windows 2000 Computer Management tool, creating company domain Web sites 376
Windows 2000 Server
accessing legacy applications and data
connecting to SNA 533
migrating to Component Services 553
clustering, capacity planning issues 151
commands, Apache vs. IIS 83
compatibility with UNIX application tools 116
installing for migration 63
migrating from UNIX servers 82
ntbugtraq 600
security *See* security, Windows 2000 Server configurations
vs.
UNIX logging 75
Windows NT 4.0 logon privileges 469
Windows 2000 Server environment, accessing legacy applications and data
integrating IIS and applications 534
mapping transaction tasks 557
using COMTI 535
Windows Active Directory 389, 493
Windows DNA (Distributed interNet Applications) 251
Windows Explorer vs. IIS snap-in 76

Windows file conventions vs. UNIX (table) 118
Windows file descriptors vs. UNIX (table) 118
Windows file systems vs. UNIX 69
Windows files, converting end-of-line markers 69
Windows Internet Name Service (WINS), clustering features 389
Windows Management Instrumentation (WMI), clustering features 389
Windows mapper, accessing remote resources (table) 501
Windows NT 4.0 vs. Windows 2000 logon privileges 469
Windows NT Services for UNIX 63
Windows Script Components
  building for Web applications 269
  COM components *See* COM (Component Object Model) components
Windows Script Host (WSH)
  automating administration 397
  working with IIS Admin Objects/ADSI 10
Windows-based servers
  administering sites remotely 406
  memory management 177
WINS (Windows Internet Name Service), clustering features 389
WinZip, replicating UNIX files 69
wizards
  Certificate Manager Export Wizard 492
  Components Wizard (Windows) 230
  CTL Wizard 2
  DTS (Data Transformation Services) Export Wizard 550
  DTS (Data Transformation Services) Import Wizard 550
  Form Wizard 291
  IIS Migration Wizard *See* IIS Migration Wizard
  Import Wizard (COBOL) 538, 539, 541, 542
  New Virtual Directory Wizard 93, 99
  Permissions Wizard 2, 79, 500
  Setup Wizard 62
  Web Publishing Wizard 386
  Web Server Certificate Wizard 2, 82, 504
WMI (Windows Management Instrumentation), clustering features 389
worker threads vs. thread-per-client 201
working sets
  defined 177
  IIS 5.0 working sets
    maintaining Object Cache 186
    monitoring working sets with counters (table) 184
working sets *(continued)*
  IIS 5.0 working sets *(continued)*
    overview 183
    using PerfMon 184
  Inetinfo working sets
    available bytes 185
    caching scripts in ASP pages 186
    monitoring 183
    page faults 185
  optimizing memory usage 194
World Wide Web Publishing Service 304
Write method, Response object (ASP) 262, 295
Write permissions 81, 379, 402
write requests, analyzing data 215
Write statements 300
WSH (Windows Script Host)
  automating administration 397
  working with IIS Admin Objects/ADSI 10

X

X.509 certificate type 491
XCopy utility 67, 109