

*ID Workbench
IBMIDDoc User's Guide and Reference
Release 3.4*

SH21-0783-09

October 30, 2000

Mike Temple

ID Workbench



IBMIDDoc User's Guide and Reference

Release 34

ID Workbench



IBMIDDoc User's Guide and Reference

Release 34

Note

Before using this information, be sure to read the general information under “Appendix C. Notices” on page 459.

This manual was produced using IBMIDDoc SGML, the Adept editor, and processed for print and online using the ID Workbench.

October 30, 2000

This edition applies to the IBMIDDoc Language, version 4.3.4; and to ID Workbench, release 3.4, and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 1992, 2000. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About This Book.	ix	+ Definition lists	26
Who Should Read This Book.	ix	+ Parameter lists	28
What You Should Already Know	ix	+ Message and code lists	29
How This Book is Organized.	ix	+ Overriding the message list subheadings	31
How to Use This Book.	ix	+ Compacting lists.	32
+ Summary of Changes	xi	+ Grouping list items	32
		+ Separating or bridging list items	33
Part 1. Introduction to IBMIDDoc.	1		
Chapter 1. Introduction to IBMIDDoc	3	+ Chapter 4. Highlighting, Citing, Noting, and Quoting	35
What is IBMIDDoc?	3	+ Highlighting	35
+ IBMIDDoc Documents	3	+ Simple title citations	37
IBMIDDoc Terms and Concepts	4	+ Notes	37
Documents and Markup	4	+ Note lists	38
Elements and Tags	4	+ Footnotes	38
+ Containment	5	+ Quotes and excerpts	39
Entities	5	+ Labeled boxes	40
+ Marked sections	7	+ The perils of processing: Attention, caution, and danger	40
+ Processing instructions	7	+ Annotations	41
Object Libraries	7	+ Qualifying information	41
Attributes	8	+ Trademarks	42
Property and Class Definition.	8		
+ Separation of Content and Style	9	Chapter 5. Examples, artwork, and multimedia	43
IBMIDDoc Markup Considerations and Rules	9	+ Just plain lines	43
Ending an Element	9	+ Examples of computer output	44
Omitted Tags and Implied Elements	10	+ Literal text data	45
Markup Rules	11	+ Including artwork in documents	45
Phrase-Like and Paragraph-Like Elements	12	+ Creating graphic links	46
Element Groupings in IBMIDDoc	12	Figures	47
		Figure captions and descriptions	48
Part 2. Using IBMIDDoc Markup	13	+ Character graphics	48
Chapter 2. Using basic IBMIDDoc elements to create a document	17	+ Screens	49
IBMIDDoc Document Structure.	17	+ Math formulas	50
Creating an IBMIDDoc Document.	17	+ Chapter 6. Cross-referencing	51
+ Creating the body of your document	17	+ Referencing a figure	52
+ Creating divisions (D element)	18	+ Referencing a table	53
+ Creating paragraphs (P element)	18	+ Referencing a list item	53
+ Deciding which elements to use	19	+ Referencing anything at all	54
+ Creating a heading hierarchy	20	+ Controlling the form of cross references	55
+ Division prologs.	21		
+ Division introductions.	21	+ Chapter 7. Creating IBMIDDoc Tables	57
Using parts to organize your chapters	22	+ IBMIDDoc Table Markup Concepts	57
+ Chapter 3. All kinds of lists	23	+ Creating simple tables.	58
+ Unordered lists	23	+ Specifying table column widths.	58
+ Simple lists	24	+ Table captions and descriptions.	59
+ Ordered lists	24	+ Page, column, and line-wide tables	60
+ Checkoff ordered lists	25	+ Splitting tables between pages	61
+ Customer setup lists	25	+ Affecting how a table appears	61
+ Continuing ordered lists	25	+ Defining the Column Specifications	64
		+ Defining Rows and Entries	65
		+ A Few Simple Table Examples	65

+ A Simple Table	65
+ A Simple Table with More Options	66
+ A Simple Table with a Table Header and IBMIDDoc Elements	66
+ A Complex Table with Row and Column Spans	67
+ A Complex Table Header	68
+ Adding footnotes to a table	68

Chapter 8. The document structure of an IBMIDDoc document 71

About the IBMIDDoc tag	71
+ Getting in style, the document style, that is	71
+ Setting the IBM copyright	72
+ Setting the security classification	73
+ Setting page numbering to sequential or folio-by-chapter	73
+ Creating multiple volumes for a book	73
+ Controlling generated chapter, part, and appendix titles	74
+ Specifying the language of the document	74
Bookmarks for PDF tables of contents	75
Line justification for DBCS languages.	75
About the prolog	75
+ Document title	76
+ Document number	76
+ Author and Address	77
+ Date	77
+ Improving the searching of PDF books	77
Other prolog elements.	77
+ Adding to the front or back cover (CoverDef)	78
Using CopyRDefs	78
Using IBMProdInfo.	79
Using Property Definitions (PropDefs)	79
PropDefs and Common Property Values.	79
Using LDescs and Nameloc	82
Using GLDefs	83
Using BibEntryDefs.	84
+ Front matter (FrontM)	85
+ Notices and Edition notices	86
+ Table of contents	86
+ List of figures	87
+ List of tables	87
+ The preface	87
+ Summary of changes	87
+ Special sections	88
+ IBM Safety text	88
About back matter (BackM)	88
+ Using appendix	88
Using glossary	89
Using bibliography (Bibliog)	89
+ Using part number index (PNIndex)	89
+ Using Index	90
+ Using reader's comment form (RCF)	90

Chapter 9. Revision Elements and Marked Notes 91

+ Using Revisions	91
+ Defining Revisions in the RevDefs Element.	91
+ Indicating Revisions in the Document Markup	92
+ Marking text for deletion	93

+ Creating Collections of Marked Notes	93
+ Using the Mark Element	94
+ Defining Marked Actions and Classes	94
+ Using the MkNote Element	94
+ Generating a Collection with MarkList Element	95
+ A Marked Notes Markup Example	95

Chapter 10. Indexing 97

+ Structuring a basic index	98
+ Basic index tagging.	99
+ Placement of index tags	99
+ Position method	99
+ Cross referencing index entries	100
+ Where to put index entries	101
+ Defining index entries (central indexing)	102
+ Creating index entries by cross-indexing	102
+ Defining See and See-also references	103
+ Generating the index	105
+ Creating a master index	105

Chapter 11. All about linking 109

+ Linking 101	109
+ Creating links within a document	109
+ Linking to another document	110
+ Citation link to an IBMIDDoc document	111
+ Linking to an HTML (or web) document	111
+ Linking to headings in a PDF document	112
Linking to an IPF document	112

Chapter 12. Glossaries 115

Defining Terms	116
+ Separating letter groups in a glossary	116
Defining Classes for Terms	116

Chapter 13. Bibliographies and citations 119

+ Identifying books and documents	119
+ Using title citations	120
+ Citations	120
Automatically generating a bibliography	121
+ Defining library entries	121
Linking BibEntry elements and other documents	122
An example of using BibEntry and BibEntryDefs	122

+ Chapter 14. Program Syntax

+ Definitions 125

+ Defining the syntax diagram	125
+ The Syntax element	127
+ The Group element	128
+ The KWD (keyword) element	130
+ The VAR (variable) element	130
+ The OPER (operator) element	131
+ The SEP (separator) element	131
+ The Delim (delimiter) element.	131
+ The RepSep (repeat separator) element	132
+ The FRAGMENT and FRAGREF (fragment reference) element	133
+ Syntax Notes	134
+ Syntax Phrases	135

+ Examples of Syntax Definitions and Markup . . .	136
+ Example 1: A simple syntax definition . . .	136
+ Example 2: A simple syntax definition that repeats . . .	136
+ Example 3: A more complex syntax definition . . .	137
+ Example 4: A variation on Example 3 . . .	137
+ Example 5: A syntax definition showing a fragment and significant blanks . . .	138
+ Example 6: A syntax definition with automatic fragmenting . . .	139

+ Chapter 15. Developing Programming Language Reference Materials 141

+ The Structure of a Language Element Reference	
+ Section	141
+ Describing Your Reference Section	142
+ Describing the language element	144
+ Example of a Simple Language Element Reference	
+ Section	144
+ DISHDEF defining a dish	147
+ EVALUATE evaluate nutrition, cost, or preparation time	148

Chapter 16. Defining Modular Information 151

Examples of Using Modular Information 152

+ Chapter 17. File, text, and character	
+ entities and reusing information . . .	155
+ File and text entities	155
+ Special characters	156
+ Reusing elements from an object library	166
+ Reusing attributes in the CONLOC reference . .	168
+ Cross-referencing items that use CONLOC . .	169

+ Chapter 18. Conditionally including information 171

+ Property-Based Retrieval	171
+ The Props Attribute	171
+ Setting the properties to true or false	172
+ Specifying boolean properties	173
+ Retrieval alternatives	174
+ Using Marked Sections	175
+ Controlling SGML Delimiter Recognition . . .	176

Chapter 19. Property and Class Definition 177

Defining Element Properties	177
Defining Element Properties Directly	177
Defining Element Properties by Linking	178
Defining Element Properties Using Inheritance	179
Defining the Security Classification from an Element's Children	179
Defining Reusable Sets of Element Properties	179
Defining Element Classes	179
Property-Reference Links	181

Chapter 20. Creating maintenance analysis procedures 183

MAP 0010: Baby Johnny is crying	184
MAP 0020: The Steak is Frozen	185
Using ProcEntry for Entry Requirements	185
Using ProcStep and ProcCmnd to Describe Each Step	185
Using DecisionPnt for Outcome-Dependent Action Descriptions	186
Using RefKeys to Refer to Labels in a Graphic . .	186
Using ProcExit to Complete a Procedure or Sub-Procedure	187
Procedure Markup Examples	187
Starting the Procedure	187
Describing the Entry Point for the Procedure . .	187
Entering the Procedure Steps	187
Exiting the Procedure.	188
Controlling Procedure Output Styles	188

+ Chapter 21. Creating parts catalog lists 191

+ Assembly 1: Bicycle	192
+ Markup source	192
+ Creating the heading for a component list . . .	192
+ Developing the component list	193
+ Including comments in the component list . . .	194
+ Cross-referencing part assemblies and component lists	194
+ Assembly 2: Wheel, front	195
+ Keeping track of assemblies and parts	195
+ Getting an assembly list	195
+ Getting a part number index	196

Part 3. IBMIDDoc Markup Reference 197

Chapter 22. Reference Explanation 201

Element and Attribute Descriptions	201
How to Read the Syntax Diagrams	201
Common Element Attributes (large set).	204
Common Element Attributes (small set)	205

Chapter 23. IBMIDDoc Elements . . . 207

+ Abbrev (abbreviations)	207
+ Abstract (abstract)	207
+ Address (address)	208
+ Annot (annotation)	209
+ AnnotBody (annotation body)	210
+ APL (APL data)	211
+ Appendix	212
+ Approvers (document approvers).	212
+ AreaDef (defines graphic hot spot area)	213
+ AsmList (list of parts assemblies).	214
+ Attention (safety notice)	214
+ Author	215
+ Authors	216
+ BackCover (back cover)	217
+ BackM (back matter)	217
+ BibEntry (bibliographic entry)	218

+ BibEntryDefs (contains bibliographic entries)	219	FrontCover	270
+ Bibliog (bibliography)	219	+ FrontM (front matter).	270
+ BibList (bibliography entry list)	220	+ GendTitle (default title specification)	272
+ Bin (binary data)	221	+ GL (glossary list)	272
+ Body (document body)	222	+ GLBlk (glossary list block)	274
+ BOFNum (bill of forms number)	222	+ GLDefs (glossary definitions)	275
+ Bridge (bridge between concepts).	223	+ GLEntry (glossary list entry)	275
+ Cap (caption)	225	+ Glossary	276
+ Caution (caution notice).	225	+ Group	277
+ CGraphic (character graphic)	226	+ Hex (hexadecimal).	277
+ Char (character data)	227	+ IBM BibEntry (IBM bibliographic entry).	279
+ CI (component item)	228	+ IBMBOFNum (bill of forms number)	280
+ Cit (document citation)	229	+ IBMDOcNum (IBM document number).	280
+ ClassDef (element class definition)	230	+ IBMFeatNum (IBM feature number).	281
CLE (content list entry)	231	IBMIDDoc (IBM-specific product documentation)	281
+ Code (message code number)	232	IBMLibEntry (IBM document library definition)	287
+ ColSpec (column specification)	232	IBMMail (IBMMail e-mail address)	289
+ CompCmt (component comment)	234	+ IBMPartNum (IBM part number).	290
+ CompL (component list).	234	+ IBMPgmNum (IBM program number)	290
Cond (procedure result)	235	+ IBMProdInfo (IBM product information)	291
+ ContainedDocs (documents in IBMLibEntry and LibEntry).	236	+ IBMSafety (IBM safety notices)	291
CopyR (copyrights)	237	+ IdxDefs (central index entries).	292
CopyRDefs (copyright definitions)	237	+ IdxTerm (index term).	293
+ Corp (enterprise name and address).	238	+ Index	293
+ CorpName (corporation name)	239	+ Internet (internet e-mail address).	294
+ CoverDef (cover definition).	239	+ IRef (index entry reference).	294
+ CritDate (critical date for a document)	240	+ ISBN (document ISBN number)	295
+ CritDates (set of critical dates).	240	+ I1 (primary index entry).	296
+ D (hierarchical division).	242	+ I2 (secondary index entry)	297
+ Danger (danger notice)	243	+ I3 (tertiary index entry)	298
+ Date	244	Kwd (syntax keyword)	299
+ DBlk (Division block).	245	+ L (explicit link)	300
+ DBody (division body)	246	LDescs (link descriptions)	302
+ Dec (decimal number)	247	+ LE (language element)	302
DecisionPnt (decision point)	247	LeDesc (language element description)	304
+ Defn (definition of a term)	249	+ LEDI (language element description item).	304
+ DefnHd (definition description heading)	250	+ Legend	306
+ Delim (syntax delimiter).	250	+ LEN (language element name).	307
+ Desc (element description)	251	+ LERS (language element reference section)	308
+ DIntro (division introduction)	252	+ LERSDef (LERS property definition).	310
+ DL (definition list).	253	+ LI (list item).	311
+ DLBlk (definition list block)	254	+ LibEntry (document library definition)	312
+ DLEntry (definition list entry).	255	+ LIBlk (list item block)	314
+ DocTitle (document title)	256	+ Library	314
+ DProlog (division prolog)	256	+ Lines (text with line boundaries)	315
+ DSum (division summary)	257	+ Litdata (literal data)	316
+ DVCFObj (DVCF Migration Element)	258	+ LQ (stand-alone quotation)	318
+ EdNotices (edition notices)	259	+ Maintainer (reader comment)	320
Else (other procedure path to follow)	259	+ Mark (marked note definition).	320
+ Entry (table entry).	260	+ MarkList (marked note list).	321
+ ExternalFileName	262	+ MasterIndex (master index).	323
+ Fig (figure)	262	+ MasterIndexInfo (master index information)	324
+ FigList (list of figures)	263	+ MasterIndexObj (master index object)	324
+ FigSeg (figure segment)	264	+ MasterIndexPrefix (master index prefix)	325
+ FileNum (file number)	265	+ MD (marked deletion)	326
+ Fn (footnote)	265	+ MkAction (marked note action definition).	327
+ FNList (footnote list)	266	+ MkClass (marked note class definition).	328
+ Formula (math formula).	267	MkDesc (mark description).	329
+ Fragment (syntax fragment)	268	MkNote (marked note)	331
+ FragRef (syntax fragment reference)	269	+ MMOBj (multi-media object; artwork)	333
		+ MMOBjLink (multi-media object link)	335

Mod (information module)	336	+ Prolog (document metainformation)	395
ModInfo (modular information)	338	+ PropDef (property set definition)	395
ModInfoDef (modular information property definition)	340	+ PropDefs (property definitions)	396
ModItemDef (item class definitions)	341	+ PropDesc (property description)	397
ModDesc (modular content description)	343	+ PropGroup (property group)	397
ModItem (module description item)	344	+ PrtLoc (country where printed)	398
+ ModLvl (modification level)	346	+ PublicID (public identifier)	399
ModName (modular information element name)	346	+ Publisher (document publisher)	400
+ Msg (message or code description)	348	+ PV (parameter variable)	400
+ MsgItem (message description item)	348	+ Q (quotation phrase)	402
+ MsgItemDef (definition of message description items)	350	+ Qualif (qualification)	403
+ MsgList (list of message or code descriptions)	352	+ QualifDefs (qualification definitions)	404
+ MsgNum (message identifier)	353	+ RCF (reader comment form)	404
MsgText (message text)	354	+ RefKey (reference key)	405
+ MV (message variable)	355	+ Release (product release identifier)	406
+ Name (person's name)	357	+ RepSep (syntax repeat separator)	406
+ NameLoc (named location)	357	+ RetKey (retrieval key)	407
+ NItem (notice item)	359	+ Rev (revision)	408
NMList (named list of IDs or entities)	360	+ RevDefs (revision tracking information)	409
+ Note	362	+ Row (table row)	409
+ NoteBody (note body)	362	+ Safety (safety notices)	411
+ NoteList (ordered note list)	363	+ Screen (display screen)	411
+ Notices (contains notices)	364	+ Sem (semantic meaning)	412
Notloc (notation-specific location)	365	+ Sep (syntactic separator)	412
+ Num (number)	366	+ SOA (summary of amendments)	413
+ ObjLib (object library)	367	+ SpanSpec (span specification)	414
ObjLibBody (object library body)	367	+ SpecDProlog (special section division prolog)	415
+ ObjRef (object reference)	368	+ StepNotes (step notes)	416
+ Oct (octal number)	369	+ StepRef (procedure step reference)	416
+ OL (ordered list)	370	+ STitle (shortened title)	416
Oper (syntax operator)	371	+ SubTitle (descriptive subtitle)	417
OrderNum (order number)	372	+ SynBlk (syntax block)	417
+ OrigBMDocNum (original IBM document number)	372	+ SynNote (syntax note)	418
+ Owners	373	+ SynPh (syntax phrase)	419
+ P (paragraph)	374	+ Syntax (syntax diagram)	420
+ Parm (parameter list entry)	375	+ Table	423
+ ParmBlk (parameter list block)	376	+ TBody (table body)	424
+ ParmL (parameter list)	376	+ Term	425
+ Part (major document part)	378	+ TermHd (term heading)	426
+ PartAsm (part assembly)	379	+ TextAlt (text alternative)	427
+ PartAsmSeg (part assembly segment)	380	+ TFoot (table footer)	427
+ PBlk (paragraph block)	380	+ TGroup (table group)	428
+ Person (person's name and address)	381	+ THead (table heading)	429
+ Ph (Phrase)	382	+ Then (procedure action to take)	430
+ Phone (telephone number)	384	+ Title	431
+ PK (programming keyword)	385	TitleBlk (title information)	432
+ PNIndex (part number index)	386	+ TList (list of tables)	432
+ PostalCode (postal or zip code)	387	+ TM (Trademark)	433
+ Preface	387	+ TOC (table of contents)	435
+ Proc (procedure)	388	+ UL (unordered list)	436
+ ProcCmdnd (procedure command)	389	+ Var (syntax variable)	437
+ ProcEntry (procedure entry point)	390	+ Version (product version number)	437
+ ProcExit (procedure exit point)	391	+ VNet (IBM VNet mail address)	438
+ ProcIntro (procedure introduction)	391	+ VolId (volume identifier)	438
+ ProcStep (procedure step)	392	+ Warning (warning notice)	439
+ ProcSumm (procedure summary)	393	+ WebPage	440
+ ProcSummItem (procedure summary item)	393	+ Xmp (example)	440
+ ProdInfo (product information)	393	+ XPh (example phrase)	441
+ ProdName (product name)	394	+ XRef (cross reference)	442

Part 4. Appendixes 447

Appendix A. IBMIDDoc Supported Notations	449
---	------------

Appendix B. Proposed IBM Standard for Formal Public Identifiers	451
Owner Identifier	451
Public Text Class and Public Text Description	452

Public Text Language.	457
-------------------------------	-----

Appendix C. Notices	459
--------------------------------------	------------

Part Number Index	461
------------------------------------	------------

Index	463
------------------------	------------

About This Book

This book describes how to use IBMIDDoc, which is a document markup language based on Standard Generalized Markup Language (SGML).

Who Should Read This Book

Anyone who wants to create documents with IBMIDDoc markup or design a library of documents that have IBMIDDoc markup should read this book.

If you are new to SGML and to the ID Workbench, please first get and read the *IDWB Getting Started and User's Guide*. Access the latest versions of these books from the ID Workbench Documents page:

<http://w3.rchland.ibm.com/projects/IDWB/documents/idwbdocs.htm>

What You Should Already Know

You should be familiar with the process of creating a document and the general concepts of document markup. You should also know how to use an SGML editor.

Although IBMIDDoc markup can be entered using a text editor, an SGML editor such as the Arbortext Adept editor or Frame2000 is strongly recommended. The SGML-aware editors ensure your markup is correct before you do any formatting; thus saving you time and money in extra formatting and debugging runs.

How This Book is Organized

This book is organized into the following parts:

- “Part 1. Introduction to IBMIDDoc” on page 1 describes basic IBMIDDoc terms and concepts and IBMIDDoc markup rules.
- “Part 2. Using IBMIDDoc Markup” on page 13 describes how to use IBMIDDoc markup to create the different parts of a document.
- “Part 3. IBMIDDoc Markup Reference” on page 197 describes the markup for IBMIDDoc elements and attributes.

How to Use This Book

If you are creating information using IBMIDDoc, you should understand the concepts in “Part 1. Introduction to IBMIDDoc” on page 1 and create your information using “Part 2. Using IBMIDDoc Markup” on page 13. Refer to “Part 3. IBMIDDoc Markup Reference” on page 197 as necessary.

If you are planning or designing libraries or information, you should be generally familiar with IBMIDDoc and be very familiar with the following information, which is essential to planning and designing information:

- “Chapter 1. Introduction to IBMIDDoc” on page 3
- “Chapter 2. Using basic IBMIDDoc elements to create a document” on page 17
- “Chapter 11. All about linking” on page 109
- “Chapter 15. Developing Programming Language Reference Materials” on page 141

- “Chapter 17. File, text, and character entities and reusing information” on page 155
- “Chapter 19. Property and Class Definition” on page 177
- “Chapter 18. Conditionally including information” on page 171

Summary of Changes

Changes for IBMIDDoc version 4.3.4 and IDWB release 3.4:

- Updates after October 12th:
 - New symbols for the e(logo)server logos; see “Special characters” on page 156.
 - New BRAND and NEWBRAND attributes added to the IBMIDDoc tag; see “IBMIDDoc (IBM-specific product documentation)” on page 281.
- Add MMOBJ to the content model for Screen. Requirement R004878.
- Line justification for DBCS languages: `ibmiddoc style="xpp:(justify)"`. This is used only for DBCS languages. Given limited scope and that it is not used by Frame2000 users, it was left as a passthrough. Requirement R005448.
- RETKEY=None | First | Last | FirstLast | NoDup on LERS, MSGLIST, and GL. Used to enable or disable the automatic running heads for the LERS, Msglist, GL in a document. Cannot set at the individual element level. All explicitly coded Retkey elements are honored. If you nest elements that can generate a running head (for example: msglist in lers), only the outer active generated head is used. That is, if you have specified automated retkey generation for LERS and MSGLIST, then a MsgNo inside LERs would not be used in the retkey area. But if you had an explicit retkey inside the msg, then the retkey is honored as an explicit override. The Xyvision transform will only use one style of retrieval per retkey type First for Lers; NoDup for MsgList and GL. Requirement RALMAR04.1997a.
- IBMIDDoc MAXTOC=number to enable you to specify the highest level head to go in the Table of Contents. MAXTOC was picked to distinguish it slightly from the Bookie :docprof toc=123 which actually let authors skip some heads altogether but still get lower level heads. Currently done as a passthrough for Bookie and Frame2000, no support for Xyvision. Requirement R004769.
- D toc=toc|notoc> also for special D type elements. Controls whether this particular heading is included in the TOC if the TOC includes those levels of headings. Currently supported as a passthrough for use by the PC Company.
- Change PGWIDE values for TABLE: `pgwide=0|1|2` The new value 2 on the `pgwide` is to indicate width=textline behavior. This is currently done with passthroughs.
- Add FRAME attribute to Fig Frame: `Fig frame=NONE|BOX|RULES`. Requirement R004763.
- Add SynStyle to syntax: `SynStyle= Space| Box| Rule| LblBox`. Requirement R004763
- Add one and two character termwidth settings to DL and PARML. The defined attribute values for termwidth are now small | medium | large | 1 | 2. Requirement R005298.
- Change processing of DVCF OBJ to error message.
- Add DBLK to support including multiple divisions from an object library. It is allowed wherever D is allowed.
- Add PBLK to content model for FrontCover, to allow for multiple paragraphs and label boxes on cover. Requirement R005266
- Add LAYOUT to TOC, FIGLIST, TLIST, and Index. This is to support using less columns than defined in the default style to provide room for long terms. The values will be onecol, twocol, threecol, and default-layout. Requirements R005530 and R005509

- Provide support for index folio prefix used for multiple volumes: Added MULTIVOL=OneVol | Index-Folio to IBMIDDOC , which will add X- as a prefix for the page numbers in the index and start the numbering from 1. Requirement R004964.
- Added comments to DTDs with Language and DocStyle values currently supported.
- Allow compact lists: LINESPACE=SPACE|COMPACT on all lists (ol ul gl msglist codel parml dl notel), sublists automatically inherit the linespace but can override. We chose this attribute rather than COMPACT=COMPACT|NOCOMPACT to allow for future growth like doublespace. Requirement R005298
- With patch IDWXF036, the CONLOC attribute passes the attributes from the elements contained in an OBJLIB. See “Reusing elements from an object library” on page 166 for more information.

Part 1. Introduction to IBMIDDoc

Chapter 1. Introduction to IBMIDDoc	3
What is IBMIDDoc?	3
+ IBMIDDoc Documents	3
IBMIDDoc Terms and Concepts	4
Documents and Markup	4
Elements and Tags	4
+ Containment	5
Entities	5
+ Marked sections	7
+ Processing instructions	7
Object Libraries	7
Attributes	8
Property and Class Definition.	8
+ Separation of Content and Style	9
IBMIDDoc Markup Considerations and Rules	9
Ending an Element	9
Omitted Tags and Implied Elements	10
Markup Rules	11
Phrase-Like and Paragraph-Like Elements	12
Element Groupings in IBMIDDoc	12

Chapter 1. Introduction to IBMIDDoc

This chapter introduces IBMIDDoc and describes IBMIDDoc terms and concepts. It also includes considerations and rules for IBMIDDoc markup.

What is IBMIDDoc?

IBMIDDoc is a document markup language based on Standard Generalized Markup Language (SGML). SGML is an international standard for representing the elements and structure of electronically stored information so that a person or computer program can understand and use those elements and structure. The electronically stored information can be one or more files that make up a document.

IBMIDDoc Documents

An IBMIDDoc document is a valid SGML document. A valid SGML document is comprised of:

- A document type declaration that contains or references a document type definition (DTD)
- A document instance (your text) which conforms to the DTD contained in or referenced in the document type declaration

The IBMIDDoc DTD must be referenced by all IBMIDDoc documents. Figure 1 illustrates a valid IBMIDDoc document type reference.

```
<!DOCTYPE IBMIDDOC PUBLIC "+//ISBN 0-933186::IBM//DTD IBMIDDoc//EN" >
```

Figure 1. Document Type Declaration for an IBMIDDoc Document

Figure 1 shows the document type declaration, which names the document type (IBMIDDoc). It also references the PUBLIC identifier for the DTD. The public identifier is a name that uses a format defined by the SGML standard. This name format allows us to point to information in a system-independent way. The SGML application that is processing the SGML data uses the identifier to transform the data being read to an identifier that works on the SGML system being used.

The document instance must conform to the document type definition. For IBMIDDoc documents, this is the IBMIDDoc DTD. Figure 2 shows the absolute minimum IBMIDDoc document.

```
<!DOCTYPE IBMIDDoc public "+//ISBN 0-933186::IBM//DTD IBMIDDoc//EN">
<IBMIDDOC>
<body>
```

Figure 2. Minimum IBMIDDoc Document Markup

Figure 3 on page 4 shows a more complete, but still small, IBMIDDoc document:

```

<!DOCTYPE IBMIDDoc public "-//ISBN 0-933186::IBM//DTD IBMIDDoc//EN">
<ibmidoc>
<body>
<d>
<dprolog><titleblk>
<title>My Little Document</title>
</titleblk></dprolog>
<dbody>
<p>This is my first sample document. Thank-you for
reading it.</p>
</dbody></d>
</body>
</ibmidoc>

```

Figure 3. IBMIDDoc Document

IBMIDDoc Terms and Concepts

This section introduces IBMIDDoc terms and concepts, including markup and tags, containment, entities, object libraries, attributes, property and class definition, and separation of content and style.

Documents and Markup

A *document* is a collection of information that is processed as a unit. An IBMIDDoc document consists of information (text and graphics) and IBMIDDoc markup that defines and identifies the structure and the elements of the document.

The IBMIDDoc *document type definition* (DTD, not "DDT") defines the document type and the valid elements or tags you can use. Your document that contains the information (text and graphics) and corresponding markup is called the *document instance*.

Markup is information in a source document that enables a person or system to process the document. The kinds of markup you can use in IBMIDDoc are: descriptive markup (tags), markup declarations, entity references, marked sections, and processing instructions.

Elements and Tags

An *element* is a component of a document such as a paragraph, an unordered list, or a figure. *Tags* (or *descriptive markup*) are used to identify elements. A tag is composed of a tag open delimiter (< for a start tag), an element identifier (for example, p for a paragraph), and a tag close delimiter (>). Ending tags begin with </, an element identifier, and >.

In the following example of a paragraph, <P> marks the start of the paragraph element and </P> marks the end of the paragraph element:

```

<P>This is the content
of a paragraph element.
</P>

```

The *content* of an element is whatever is between the start and end tags for the element. An element can contain information (text or multimedia objects), other elements, or a mixture of information and elements. Elements that have no content do not have end tags, such as the XRef element.

Containment

One element can contain another element either directly or indirectly, known as *direct containment* or *indirect containment*, respectively.

In the following example, the first paragraph directly contains an ordered list, the ordered list directly contains two list items, and the first list item directly contains a paragraph. The first paragraph indirectly contains the list items and the paragraph contained by the first list item.

```
<p>This is a list:
<ol>
<li>First list item
<p>This is a paragraph within the
first list item.</p>
</li>
<li>Second list item</li>
</ol>
</p>
```

This next example, the first paragraph does not contain the ordered list, because the first paragraph is closed before the list is opened.

```
<p>This is a list:
</p>
<ol>
<li>First list item
<p>This is a paragraph within the
first list item.</p>
</li>
<li>Second list item</li>
</ol>
```

The containment structure also determines the inheritance of properties from an element to the elements it directly contains (its children). This structure also determines the properties of elements that are indirectly contained by other elements.

Containment also determines a hierarchical structure. Divisions within divisions determine headings and subordinate headings. Several SGML editors can display a tree view of a document. This view lets you see these containment and hierarchical relationships. You can tell which elements are peers, parents, or children by this kind of view.

Entities

An *entity* is any information that is referred to as a unit from a document. It can be a character string, a file, a graphic, or a collection of files. It can even be an entire document. Entities enable the reuse of information, and organization of that information into separate files.

An entity must be defined by a *markup declaration*, which is a kind of markup that controls the interpretation of other markup. Entities are not dynamic; the definition of an entity cannot be changed after it is defined. In addition, all entities must be defined at the beginning of a document. Entity declarations are part of the DTD. The declarations are usually put in the DTD subset, which is the part of the DTD that is specific to a given document.

An *entity reference* requests that entity data replace the entity reference at the place where the reference occurs. Entity references are delimited by the entity reference open delimiter (&) and the entity reference close delimiter (;).

Entities are either internal or external. An *internal entity* is an entity whose declaration includes the replacement text (the text that is to replace the entity reference). In the following example, which shows how to define and refer to an internal entity, IBMIDDoc is the replacement text and &product; the entity reference.

```
<!ENTITY product "IBMIDDoc">
:
This book teaches you how to use &product;.
```

Internal entities that are defined by IBMIDDoc include special characters such as the em-dash or backslash, which are referred to with the entity references &emdash; and &bslash;;, respectively.

These entities are contained in the IDDBKSYM.ENT file. Many of these character entity declarations use the same names as the character entities that are defined by BookMaster.

An *external entity* is an entity whose declaration defines where the replacement text can be found but does not include the replacement text in the declaration. These are more commonly called symbols (replacement words or phrases) or imbeds (files such as a chapter).

In the following example, which shows how to define and refer to an external entity, the XXXL0INT.IDE file is the external entity and &introfile; is the entity reference:

```
<!ENTITY introfile SYSTEM "xxxl0int.ide">
:
&introfile;
```

The ENTITY line defines "introfile" as the name of the entity; "system" indicates the following string "xxxl0int.ide" is a file name.

The example that follows shows the entity declarations for several IBMIDDoc files that are included in a master file document. This is like using the .im Bookmaster macro to imbed SCRIPT files in a master document.

```
:
<!Entity RKTLEDN SYSTEM 'RKTLEDN.IDE'>
<!Entity RKTLENOT SYSTEM 'RKTLENOT.IDE'>
<!Entity RKTLEPRE SYSTEM 'RKTLEPRE.IDE'>
:
<!-- Edition notice This includes the entity containing Edition Info-->
&RKTLEDN;
<TOC><GENDTITLE>
<!-- Notices This includes the entity containing the Notices -->
&RKTLENOT;
<!-- Preface This includes the entity containing the EdNotices -->
&RKTLEPRE;
</FRONTM>
```

In the first line of the example, RKTLEDN is the entity name, SYSTEM declares where the entity is stored on the local system, and 'RKTLEDN.IDE' is the actual file name. In the front-matter portion of the document, the entities are referenced where they should appear during processing.

For more information about using entities and entity references, see "Chapter 17. File, text, and character entities and reusing information" on page 155.

Adept Editor Note

For information about creating, declaring, and referencing internal and external entities using Adept, see the chapter “Editing SGML Documents with Adept” in the *ID Workbench Getting Started and User’s Guide*. If you’re not using Adept, see the user guide for the SGML editor that you are using.

Migration Note

Internal entities are “symbols” in BookMaster*, and external entities are “embedded files” in BookMaster.

You will often see entities referred to in the following ways:

Character Entity

Contains values for a special character set, as in the IDDBKSYM.ENT file.

Text Entity

Contains the replacement text in the markup declaration.

File Entity

Contains a reference to the name of another file that you want to reference within your document.

Marked sections

Marked sections are a special way of controlling a part of a document for processing. You can indicate, in the source, a part to include or ignore. IBMIDDoc has a better way of doing this at run time; using properties; see “Chapter 18. Conditionally including information” on page 171.

To use a marked section to condition text (maybe to hide text and SGML source you want to save but not have in the document), You use a marked section parameter to surround the text. For example, this is hiding a paragraph; notice the brackets and the %comment – these cause the tagged content to be ignored.

```
<![ %comment; [<p>Here's a little paragraph I want to hide.</p>]]>
```

The %comment needs to be declared, and set to ignore:

```
<!ENTITY % comment "IGNORE">
```

Processing instructions

These are special instructions that you add to your source. They are allowed almost anywhere. You use them within IBMIDDoc documents to include special formatting controls such as page breaks.

For example, the following processing instruction tells the Xyvision formatter to start a new page:

```
<?XPP:page>
```

Object Libraries

An *object library* is a collection of elements that can be used elsewhere in a document. Object libraries, like entities, also enable reuse of information. Elements in an object library can be used only within the document that contains the object

library. Object libraries can also be used for conditional processing. Conditional processing allows you to turn text on or off as you process your document.

For more information about using the elements in an object library, see “Chapter 17. File, text, and character entities and reusing information” on page 155 and “Chapter 18. Conditionally including information” on page 171.

Migration Note

Object libraries provide the function of BookMaster document version control facility (DVCF) side files.

Attributes

An *attribute* is a characteristic of an element (other than type or content) that is included with a start tag to further describe the element. Many attributes defined by IBMIDDoc are common to all elements.

In IBMIDDoc, an attribute name must:

- begin with an alphabetic character, A-Z or a-z
- contain only A-Z, a-z, 0-9, - (hyphen), . (period), and _ (underscore)
- be no more than 64 characters in length

IBMIDDoc attributes are divided into the following classes:

- **Identifying attributes** identify a given element. The ID attribute, which is a common attribute, is an identifying attribute.
- **Property attributes** define the properties of an element, such as its owner or class, and control which elements are to be processed. Language and Props are property attributes.

```
<p props="v2r3">This paragraph is used only for V2R3.</p>
```

- **Link attributes** define the link relationships between elements. Linkend and Refid are link attributes for the L and XRef elements, respectively.

```
<P>A <L LINKEND="parahead">paragraph</L> is a chunk of information.
⋮
<P>See <XREF REFID="parahead"> for more information about paragraphs.</P>
⋮
<D ID="parahead">
```

- **Style attributes** define presentation characteristics of an element. In this next example, the OLTYPE attribute says this list should format as a step list.

```
<ol oltype="step">
<li>Do this.</li>
<li>then that.</li>
</ol>
```

Property and Class Definition

Properties such as language, status, or classification can be associated with elements and are defined by using property attributes. Also, elements can inherit properties from other elements.

The PropDef element allows you to define one set of properties that can apply to several elements.

The ClassDef element allows you to define element classes that enable processing functions such as creating a detailed glossary or bibliography, generating precise associative links, or automatically indexing certain kinds of information. Element classes can also be used to control the inheritance of element properties.

In many cases, element classes are defined for an entire collection of documents by someone responsible for designing the information in the collection, such as an information designer or planner. If you are working on information for which element classes have been defined, you need to know the class names, the affected elements, and their intended use.

For more information about defining properties, see “Chapter 19. Property and Class Definition” on page 177.

Separation of Content and Style

The main intent with IBMIDDoc SGML markup is to separate the content from how it appears. The output styles are determined by style “gurus” so that all our documents look alike.

We need to write clearly and consisely; the formatters take care of how the information appears. We don’t need to worry that a second-level heading is in the proper type face and highlighting.

IBMIDDoc Markup Considerations and Rules

This section describes markup considerations for ending elements and omitting tags. It also lists IBMIDDoc markup rules.

Markup and SGML Editors

All discussions in this book about entering tags and other markup are in the context of using a non-SGML text editor to create IBMIDDoc files. With a text editor, minimizing typing is useful, and IBMIDDoc does what it can to keep typing to a minimum. However, you should create and edit IBMIDDoc files with editors that support SGML, like Adept or Frame+SGML.

With an SGML editor, you make selections from menus rather than typing in tags. Thus, minimizing typing is not an issue with SGML editors.

The markup shown in this book is usually the minimum markup required. However, SGML editors often insert omissible tags for elements. Also, SGML editors often insert an optional attribute name when you enter a value for the attribute. Thus, when you request a view that shows the markup in an SGML editor, you can see tags that you did not select or, if you use a text editor, that you do not need to type.

Ending an Element

In SGML markup, an element is ended either by an end tag or by another element that cannot be directly contained by the first element. Elements that contain nothing have no end tags, and some elements have optional end tags.

The paragraph element has an optional end tag. Because a paragraph cannot directly contain another paragraph, one paragraph automatically ends when another paragraph begins, regardless of whether a paragraph end tag is used.

It is almost never wrong to use an end tag. The exceptions are:

- When an element may never have content. Such elements are called **Empty** elements. XRef is an example of an empty element.
- When attributes which force the element to be empty are specified on the element. All elements have several possible such attributes. CONLOC is an example of such a special attribute. LitData's OBJ attribute is another example of a special attribute.

In both of these examples, the attributes are content references. They point to other elements by ID, or to other entities by name. The content of the elements or entities that are the target of the content reference are used at the point where the content reference is made.

Migration Note

Bookmaster's Artwork tag has a name attribute which behaves the same way as described for these special attribute content references.

To determine whether an end tag is required or optional for a particular element, check the description for that element.

Omitted Tags and Implied Elements

In SGML, some start and end tags can be omitted and the corresponding elements can be implied. The omission makes it easier to create IBMIDDoc documents if you are using a text editor. Remember, however, that you may see error messages about the implied elements. The following division element markup demonstrates how omitted tags work. D contains two main elements, DProlog and DBody. DProlog contains TitleBlk and Title which contain the title of the division, and DBody contains the content of the division. You can type the following markup, in which DProlog, Title, and DBody are all automatically implied:

```
<D>Using IBMIDDoc
<P>IBMIDDoc is IBM Information Development's
implementation of the SGML standard for IBM documentation.
```

If you typed the complete markup, it would look like this:

```
<D>
  <DPROLOG>
    <TITLEBLK>
      <TITLE>Using IBMIDDoc</TITLE>
    </TITLEBLK>
  </DPROLOG>
  <DBODY>
    <P>IBMIDDoc is IBM Information Development's
      implementation of the SGML standard for IBM documentation.
    </P>
  </DBODY>
</D>
```

Start tags can be omitted only for required elements. Because TitleBlk, Title, DProlog, and DBody are required elements on the D element, you can omit the start and end tags. For each element with an optional title, you must explicitly enter the TitleBlk start and end tags. To determine whether an element is required in a particular context, check the description for the parent element.

Markup Rules

General markup rules for IBMIDDoc are as follows. This first set are good general rules:

- Always use the appropriate markup to identify a document element. For example, do not use a paragraph tag to create a blank line. If you use markup incorrectly, the output might appear satisfactory when processed; however, if the document is processed with a different processing program from the one you are using, the results may be unsatisfactory.
- Use an SGML comment to indicate strange or interesting markup. This not only helps you when you wonder why you did something, it also helps the poor soul that has to take over your document when you get promoted and move to other assignments. You can also include a comment in a declaration; it is delimited by double hyphens (--), as follows:

```
<!ENTITY product "IBMIDDoc" --This comment describes the entity.-->
<!ENTITY introfile SYSTEM "xxx10int.ide" --This comment describes
the entity and is too long to fit on one line.-->
:
:
<!--This is a comment by itself.-->
```

Do not use double hyphens within a comment because they end the comment.

- To “comment out” a section (so it’s hidden but not deleted), use the marked-section keyword IGNORE. See “Marked sections” on page 7 for an example.

These rules are automatically enforced by any SGML editor; you typically do not have to worry about them:

- Specify elements in the right order. Elements that occur only once in a document must be coded in the order shown in the syntax descriptions.
- Define all entities at the beginning of your document.
- For a multiple-word attribute value or for an attribute value that contains blanks or special characters, enclose the value within single or double quotation marks, as shown here:

```
<PH STYLE="bold italic">
This should be bold and italic
</PH>
```

If an attribute value contains apostrophes, use double quotation marks, as follows:

```
<P XREFTEXT="operator's tasks">tasks
```

If an attribute value contains one kind of quotation marks (single or double), use the other kind of quotation marks, as follows:

```
style='color="cyan white" bold monospace'

style="color='cyan white' bold monospace"
```

If a single-word attribute value does not contain special characters, quotation marks can be used but are not required.

- Do not hyphenate words at the end of an input line.
- You can omit the start tag only for a required element.
- You can use empty end tags as a shorthand way of ending the last element started, as shown for the Phrase element in this example:

```
<P>Phyto-daemon is a example of <PH>Generation-X-speak</>.
<P>This is another paragraph.
```

Phrase-Like and Paragraph-Like Elements

In IBMIDDoc, Phrase-Like (%PhLike) is used to refer to all phrase-like elements. Most %PhLike elements are valid anywhere that plain text is valid. There are a few exceptions to this rule where a few elements have very specific content rules.

Paragraph-Like (%PLike) Elements include the IBMIDDoc elements that are directly containable by division elements.

%DivLike Elements include the IBMIDDoc elements that can be contained (in most cases) at the same hierarchical level as a Division element.

Element Groupings in IBMIDDoc

There are several groups of elements that are referred to using generic names as a shorthand technique in this book. These are called parameter entities.

Part 2. Using IBMIDDoc Markup

Chapter 2. Using basic IBMIDDoc elements to create a document	17	Chapter 6. Cross-referencing	51
IBMIDDoc Document Structure	17	+ Referencing a figure	52
Creating an IBMIDDoc Document	17	+ Referencing a table	53
+ Creating the body of your document	17	+ Referencing a list item	53
+ Creating divisions (D element)	18	+ Referencing anything at all	54
+ Creating paragraphs (P element)	18	+ Controlling the form of cross references	55
+ Deciding which elements to use	19		
+ Creating a heading hierarchy	20	Chapter 7. Creating IBMIDDoc Tables	57
+ Division prologs	21	+ IBMIDDoc Table Markup Concepts	57
+ Division introductions	21	+ Creating simple tables	58
Using parts to organize your chapters	22	+ Specifying table column widths	58
		+ Table captions and descriptions	59
Chapter 3. All kinds of lists	23	+ Page, column, and line-wide tables	60
+ Unordered lists	23	+ Splitting tables between pages	61
+ Simple lists	24	+ Affecting how a table appears	61
+ Ordered lists	24	+ Defining the Column Specifications	64
+ Checkoff ordered lists	25	+ Defining Rows and Entries	65
+ Customer setup lists	25	+ A Few Simple Table Examples	65
+ Continuing ordered lists	25	+ A Simple Table	65
+ Definition lists	26	+ A Simple Table with More Options	66
+ Parameter lists	28	+ A Simple Table with a Table Header and IBMIDDoc Elements	66
+ Message and code lists	29	+ A Complex Table with Row and Column Spans	67
+ Overriding the message list subheadings	31	+ A Complex Table Header	68
+ Compacting lists	32	+ Adding footnotes to a table	68
+ Grouping list items	32		
+ Separating or bridging list items	33	Chapter 8. The document structure of an IBMIDDoc document	71
		About the IBMIDDoc tag	71
Chapter 4. Highlighting, Citing, Noting, and Quoting	35	+ Getting in style, the document style, that is	71
+ Highlighting	35	+ Setting the IBM copyright	72
+ Simple title citations	37	+ Setting the security classification	73
+ Notes	37	+ Setting page numbering to sequential or folio-by-chapter	73
+ Note lists	38	+ Creating multiple volumes for a book	73
+ Footnotes	38	+ Controlling generated chapter, part, and appendix titles	74
+ Quotes and excerpts	39	+ Specifying the language of the document	74
+ Labeled boxes	40	+ Bookmarks for PDF tables of contents	75
+ The perils of processing: Attention, caution, and danger	40	+ Line justification for DBCS languages	75
+ Annotations	41	About the prolog	75
+ Qualifying information	41	+ Document title	76
+ Trademarks	42	+ Document number	76
		+ Author and Address	77
Chapter 5. Examples, artwork, and multimedia	43	+ Date	77
+ Just plain lines	43	+ Improving the searching of PDF books	77
+ Examples of computer output	44	+ Other prolog elements	77
+ Literal text data	45	+ Adding to the front or back cover (CoverDef)	78
+ Including artwork in documents	45	+ Using CopyRDefs	78
+ Creating graphic links	46	+ Using IBMProdInfo	79
Figures	47	+ Using Property Definitions (PropDefs)	79
+ Figure captions and descriptions	48	+ PropDefs and Common Property Values	79
+ Character graphics	48	+ Limiting the Scope of PropDef Definitions	80
+ Screens	49	+ Using PropDefs for Conditional Processing	81
+ Math formulas	50	+ Using LDescs and Nameloc	82
		+ Using GLDefs	83

Using BibEntryDefs.	84	+ Citations	120
+ Front matter (FrontM)	85	Automatically generating a bibliography	121
+ Notices and Edition notices	86	+ Defining library entries	121
+ Other notices	86	Linking BibEntry elements and other documents	122
+ Table of contents	86	An example of using BibEntry and BibEntryDefs	122
+ List of figures	87		
+ List of tables	87	+ Chapter 14. Program Syntax Definitions	125
+ The preface	87	+ Defining the syntax diagram	125
+ Summary of changes	87	+ The Syntax element	127
+ Special sections	88	+ The Group element	128
+ IBM Safety text	88	+ The KWD (keyword) element	130
About back matter (BackM)	88	+ The VAR (variable) element	130
+ Using appendix	88	+ The OPER (operator) element	131
Using glossary	89	+ The SEP (separator) element	131
Using bibliography (Bibliog)	89	+ The Delim (delimiter) element	131
+ Using part number index (PNIndex)	89	+ The RepSep (repeat separator) element	132
+ Using Index	90	+ The FRAGMENT and FRAGREF (fragment reference) element	133
+ Using reader's comment form (RCF)	90	+ Syntax Notes	134
		+ Syntax Phrases	135
Chapter 9. Revision Elements and Marked Notes	91	+ Examples of Syntax Definitions and Markup	136
+ Using Revisions	91	+ Example 1: A simple syntax definition	136
+ Defining Revisions in the RevDefs Element	91	+ Example 2: A simple syntax definition that repeats	136
+ Indicating Revisions in the Document Markup	92	+ Example 3: A more complex syntax definition	137
+ Marking text for deletion	93	+ Example 4: A variation on Example 3	137
+ Creating Collections of Marked Notes	93	+ Example 5: A syntax definition showing a fragment and significant blanks	138
+ Using the Mark Element	94	+ Example 6: A syntax definition with automatic fragmenting	139
+ Defining Marked Actions and Classes	94		
+ Using the MkNote Element	94		
+ Generating a Collection with MarkList Element	95		
+ A Marked Notes Markup Example	95		
		+ Chapter 15. Developing Programming Language Reference Materials	141
Chapter 10. Indexing	97	+ The Structure of a Language Element Reference Section	141
+ Structuring a basic index	98	+ Describing Your Reference Section	142
+ Basic index tagging	99	+ Describing the language element	144
+ Placement of index tags	99	+ Example of a Simple Language Element Reference Section	144
+ Position method	99	+ DISHDEF defining a dish	147
+ Cross referencing index entries	100	+ EVALUATE evaluate nutrition, cost, or preparation time	148
+ Where to put index entries	101		
+ Defining index entries (central indexing)	102		
+ Creating index entries by cross-indexing	102		
+ Defining See and See-also references	103		
+ Generating the index	105		
+ Creating a master index	105		
		Chapter 16. Defining Modular Information	151
Chapter 11. All about linking	109	Examples of Using Modular Information	152
+ Linking 101	109		
+ Creating links within a document	109	+ Chapter 17. File, text, and character entities and reusing information.	155
+ Linking to another document	110	+ File and text entities	155
+ Citation link to an IBMIDDoc document	111	+ Special characters	156
+ Linking to an HTML (or web) document	111	+ Reusing elements from an object library	166
+ Linking to headings in a PDF document	112	+ Reusing attributes in the CONLOC reference	168
Linking to an IPF document	112	+ Cross-referencing items that use CONLOC	169
Chapter 12. Glossaries	115	+ Chapter 18. Conditionally including information	171
Defining Terms	116	+ Property-Based Retrieval	171
+ Separating letter groups in a glossary	116	+ The Props Attribute	171
Defining Classes for Terms	116	+ Setting the properties to true or false	172
		+ Specifying boolean properties	173
Chapter 13. Bibliographies and citations	119	+ Retrieval alternatives	174
+ Identifying books and documents	119	+ Using Marked Sections	175
+ Using title citations	120		

+ Controlling SGML Delimiter Recognition . . .	176
Chapter 19. Property and Class Definition . . .	177
Defining Element Properties	177
Defining Element Properties Directly	177
Defining Element Properties by Linking	178
Defining Element Properties Using Inheritance	179
Defining the Security Classification from an Element's Children	179
Defining Reusable Sets of Element Properties	179
Defining Element Classes	179
Property-Reference Links	181
Chapter 20. Creating maintenance analysis procedures.	183
MAP 0010: Baby Johnny is crying	184
MAP 0020: The Steak is Frozen	185
Using ProcEntry for Entry Requirements	185
Using ProcStep and ProcCmdnd to Describe Each Step	185
Using DecisionPnt for Outcome-Dependent Action Descriptions	186
Using RefKeys to Refer to Labels in a Graphic	186
Using ProcExit to Complete a Procedure or Sub-Procedure	187
Procedure Markup Examples	187
Starting the Procedure	187
Describing the Entry Point for the Procedure	187
Entering the Procedure Steps	187
Exiting the Procedure.	188
Controlling Procedure Output Styles	188
+ Chapter 21. Creating parts catalog lists.	191
+ Assembly 1: Bicycle	192
+ Markup source	192
+ Creating the heading for a component list	192
+ Developing the component list	193
+ Including comments in the component list	194
+ Cross-referencing part assemblies and component + lists	194
+ Assembly 2: Wheel, front	195
+ Keeping track of assemblies and parts	195
+ Getting an assembly list	195
+ Getting a part number index	196

Chapter 2. Using basic IBMIDDoc elements to create a document

This section describes the placement and use of divisions and other division-like elements in your document. The elements discussed in this chapter include the following:

- Body
- D, division
- P, paragraph
- Title
- TitleBlk, title block
- Part

IBMIDDoc Document Structure

The IBMIDDoc DTD defines the rules of structure and containment for all IBMIDDoc elements, and the attributes that can be used on these elements.

At the document level, IBMIDDoc documents can contain a:

- Prolog element
- FrontM (front matter) element
- Body element
- BackM (back matter) element

Not all of these elements are required in an IBMIDDoc document. When you use an SGML editor, the editor interprets the DTD rules for the correct structure and containment rules for IBMIDDoc, and enforces these rules when you are authoring.

As long as the rules (sometimes called context checking) are active, an SGML editor will only present the IBMIDDoc elements that are valid in the context in which you are editing. An SGML editor will not, for example, allow you to insert a P element directly within another P element.

While there are many aspects to creating an IBMIDDoc document, let's first focus on creating a simple one.

Creating an IBMIDDoc Document

Within the IBMIDDoc element, a IBMIDDoc document must have a Body element, which must contain a division or division-like element. We'll look at these basic elements now, and look at other IBMIDDoc elements in the chapters that follow.

Creating the body of your document

The Body element contains the body of the document. This is where you put the chapters for your document. The body can contain any number of D, LERS, MSGList, Proc, and Part elements. In the example that follows, the Body element contains two division (D) elements. Because the divisions are all contained at the same level, each is a chapter (so this is a simple document with two chapters).

```

<ibmiddoc>
<body>
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
</dbody></d>
<d>
<dprolog><titleblk>
<title>Another little chapter</title>
</titleblk></dprolog>
<dbody>
<p>It was a dark and stormy night...</p>
</dbody></d>
</body>
</ibmiddoc>

```

Creating divisions (D element)

Most often, you will insert a D element after the Body element in your document. The first division in the document body is the first chapter. This is analogous to an <H1> tag in HTML. When you insert a D element, most SGML editors automatically insert the required sub-elements for the division. In IBMIDDoc, the elements that must be included in a D element are:

DProlog

The DProlog can contain a number of elements, but the only required elements are TitleBlk and Title, which contain the heading text for that division. Stitle is an optional element that indicates a shorted title. For first-level headings, use this Stitle to shorten the running foot. Subtitle is another optional element that does nothing in a book; it's element is defined, but it is not used.

DBody

The DBody element contains the text elements that comprise the content of the division; that is, the paragraphs, lists, and your golden prose.

When you create a first-level division in the document hierarchy, the text contained in the Title element is displayed as the chapter title. For example, this shows a sample chapter, first-level division:

```

<ibmiddoc>
<body>
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
</dbody></d>
</body>
</ibmiddoc>

```

Creating paragraphs (P element)

The element you will use most often is P for Paragraph. The P element contains a paragraph, that is, a block of text representing a single idea. A paragraph can contain other elements such as lists. Paragraphs should contain a single idea, and can contain many other elements. In IBMIDDoc, paragraphs cannot directly contain other paragraphs, but they can contain other elements that contain paragraphs.

Here's a sample of a paragraph, in case you missed the examples shown in previous topics:

```
<ibmddoc>
<body>
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
</dbody></d>
</body>
</ibmddoc>
```

When creating paragraphs, keep in mind that each paragraph (like many IBMDDoc elements) is a container. If you do not wish another element (a list or figure for example) to be contained by the current paragraph, you must enter that element after the end tag for the paragraph.

Deciding which elements to use

There is often more than one permissible way to markup the document content. However, with IBMDDoc, the intent of the markup is important. For example, you could mark up a list as an unordered list:

- LI elements
List items contain individual list items.
- LIBlk elements
List item blocks contain logical groupings of list items.
- Bridge elements
Bridge elements bridge two concepts.

Here's its markup:

```
<ul>
<li>LI elements
<p>List items contain individual
list items.</p></li>
<li>LIBlk elements
<p>List item blocks contain logical
groupings of list items.</p></li>
<li>Bridge elements
<p>Bridge elements bridge two
concepts.</p></li>
</ul>
```

On the other hand, you could markup up the same information using a definition list:

LI elements

List items contain individual list items.

LIBLK elements

List item blocks contain logical groupings of list items.

BRIDGE elements

BRIDGE elements bridge two concepts.

Here's its markup:

```
<dl>
<dlentry><term>LI elementS</term>
<defn>List items contain individual list items.</defn>
```

```

</dentry>
<dentry><term>LIBLK elements</term>
<defn>List item blocks contain logical groupings of
list items.</defn>
</dentry>
<dentry><term>BRIDGE elements</term>
<defn>BRIDGE elements bridge two concepts.</defn>
</dentry>
</dl>

```

While either way is acceptable and valid IBMIDDoc markup, consistency in deciding how to mark up your information is important to the successful exploitation of IBMIDDoc markup. You need to mark up information according to its intent. Decide which IBMIDDoc markup best describes the type of information you are containing, and use that markup consistently in your information.

IBMIDDoc allows you to separate the markup from the final presentation. *You should not* mark up information so that it will “look good” a certain way in:

- a PostScript or PDF file
- an HTML set of files
- an IPF panel

If you are consistent in how you mark up your information, the resulting formatted output, for any target medium, will be treated consistently in that medium. This consistency contributes to our customer’s satisfaction with the information.

Creating a heading hierarchy

You create subheadings in your document by creating a heading hierarchy. For example, a division may be a chapter title (1st-level heading), a topic (2nd-level heading), a subtopic (3rd-level heading), and so forth. Nested divisions define this division hierarchy. Nested divisions are divisions that are contained within a division. The level of the headings produced is determined by the nesting. At this point in the book, the previous division is a third-level division.

Here’s the markup for two nested divisions; the bold shows the second-level division:

```

<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
<d>
<dprolog><titleblk>
<title>My teeny topic</title>
</titleblk></dprolog>
<dbody>
<p>Here's the beginning of a topic.</p>
</dbody></d>
</dbody></d>

```

If you need more the 6 levels of divisions, an editor might say you have “heading-itis”.

Migration Note: All other contained divisions will be treated like subheadings are treated in BookMaster. However, unlike headings in BookMaster, IBMIDDoc

divisions are automatically arranged according to their hierarchical position in the markup. Each contained division is handled at a lower heading level, so to speak.

Division prologs

After you enter the title, you can enter a number of optional elements in the division prolog, including:

- Approvers
- Authors
- BibEntryDefs, bibliography entry definitions
- CopyrDefs, copyright definitions
- CritDates, critical dates
- GlDefs, glossary definitions
- IBMProdInfo, IBM product information
- IdxDefs, Index definitions
- LDescs, Link descriptions
- Owners
- ProdInfo, product information
- PropDefs, property definitions
- QualifDefs, qualification definitions
- RevDefs, revision definitions

When you use these items in a division prolog; they take effect on that division and any nested divisions. To have these items affect the whole document, put them in the prolog. Here's a sample of a revision definition that affects this division; not the whole document:

```
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk>
<revdefs>
<rev id="v3r4" ident="use">
<date>June 5th</date>
<desc>Something happened...</desc>
</rev>
</revdefs></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
<p rev="v3r4">Something that changed on June 5th.
</p>
</dbody></d>
```

Division introductions

You can introduce the division's content with the DIntro element. This element is optional; you should usually have your first paragraph of the DBody introduce the division's content. Anyway, Dintro follows the DProlog element. The next example shows the DIntro element.

```
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dintro>
<p>My little division introductory sentence.</p>
</dintro>
<dbody>
<p>Here's the beginning of my chapter.</p>
```

Besides creating an introduction, you can also create a partial table of contents for the chapter. Here's the sample coding for a PTOC:

```
<d>
<dprolog><titleblk>
<title>Sample chapter heading</title>
</titleblk></dprolog>
<dintro>
<toc><gendtitle></toc>
</dintro><dbody>
<d>
<dprolog><titleblk>
<title>Next heading</title>
</titleblk></dprolog>
<dbody></dbody></d>
<d>
<dprolog><titleblk>
<title>Another heading</title>
</titleblk></dprolog>
<dbody></dbody></d>
</dbody></d>
```

Using parts to organize your chapters

IBMIDDoc includes the Part element, which you can use to divide your document into logical parts. This book has many divisions, but contains three parts (plus the appendixes). Parts do not affect the hierarchical ordering and numbering of divisions.

This example shows a sample book with 2 parts and 4 chapters:

```
<ibmidoc>
<body>
<part>
<dprolog><titleblk>
<title>Introduction</title>
</titleblk></dprolog>
<dbody>
<d>
<dprolog><titleblk>
<title>Salads of our neighborhood</title>
</titleblk></dprolog>
<dbody></dbody></d>
<d>
<dprolog><titleblk>
<title>Salads of the world</title>
</titleblk></dprolog>
<dbody></dbody></d>
</dbody></part>
<part>
<dprolog><titleblk>
<title>Recipies</title>
</titleblk></dprolog>
<dbody>
<d>
<dprolog><titleblk>
<title>Egg salad</title>
</titleblk></dprolog>
<dbody></dbody></d>
<d>
<dprolog><titleblk>
<title>Tuna fish salad</title>
</titleblk></dprolog>
<dbody></dbody></d>
</dbody></part></body>
</ibmidoc>
```

Chapter 3. All kinds of lists

Several types of list elements are explained in this chapter, including:

- UL, unordered (see “Unordered lists”) and simple (see “Simple lists” on page 24)
- OL, ordered (see “Ordered lists” on page 24)
- DL, definition (see “Definition lists” on page 26)
- ParmL, parameter (see “Parameter lists” on page 28)
- MsgList, message (see “Message and code lists” on page 29)

In addition, we discuss other things that are often used in lists:

1. List items can be compacted (see “Compacting lists” on page 32)
2. LiBlk, list item block; these allow you to group related list items into information blocks (see “Grouping list items” on page 32)
3. Bridges; these are for transitions between list items (see “Separating or bridging list items” on page 33)

Unordered lists

Unordered lists are used when the items in the list are fairly long, maybe even many paragraphs, but you don’t want to imply any particular sequence (as you would with an ordered list). The default appearance for an unordered list is as a bulleted list. Here’s an example of an unordered list:

- This is an item in an unordered list. To separate it from other items in the list, the formatter puts a bullet beside it.
- The paragraph that is contained in the LI element is part of the list item which contains it.

This is the contained paragraph.
- This is a separate list item in our unordered list.

Here is the IBMIDDoc markup for the unordered list in the previous example.

```
<ul>
<li>This is an item in an unordered list. To separate
it from other items in the list, the formatter puts
a bullet beside it.</li>
<li>The paragraph that is contained in the LI element
is part of the list item which contains it. <p>This
is the contained paragraph.</p></li>
<li>This is a separate list item in our unordered
list.</li>
</ul>
```

Many IBMIDDoc elements can contain lists. If you do not want the list to be contained in the element that precedes it, be sure to end the preceding element before starting the list element.

The example that follows illustrates an unordered list that is not contained by the paragraph element that immediately precedes it.

```
<p>
Text of paragraph.
</p>
<ul>
<li>Abbrev</li>
```

```
<li>Abstract</li>
<li>Bibliog</li>
<li>Appendix</li>
<li>Glossary</li>
</ul>
```

You can also use ULTYPE=CHECKOFF on an unordered list to create an unordered checkoff list. For a sample of an ordered checkoff list; see “Checkoff ordered lists” on page 25.

Simple lists

Simple lists are just what you’d think they are; they have no dingbat.¹For example, here’s a simple list:

```
bread
butter
cheese
bananas
```

A simple list starts with an unordered list, then you set the style attribute to “simple”:

```
<ul style="simple">
<li>bread</li>
<li>butter</li>
<li>cheese</li>
<li>bananas</li>
</ul>
```

Ordered lists

An ordered list contains information that must be listed in a specific sequence. This is often a list of steps which must be performed in a certain order, such as a recipe or tearing apart a PC. An ordered list looks like this:

1. Cream butter and sugar together until fluffy.
2. Beat in egg yolks one at a time.
3. Add nutmeg, cinnamon, and vanilla, and mix thoroughly. The batter should be smooth and glossy and stream off the spoon in ribbons.
4. Fold in beaten egg whites.

Do not overmix; the batter should be light and fluffy.

(I’m getting hungry.) The corresponding IBMIDDoc markup is as follows:

```
<ol>
<li>Cream butter and sugar together until fluffy.</li>
<li>Beat in egg yolks one at a time.</li>
<li>Add nutmeg, cinnamon, and vanilla, and mix thoroughly.
The batter should be smooth and glossy and stream
off the spoon in ribbons.</li>
<li>Fold in beaten egg whites.
<p>Do not overmix; the batter should be light and fluffy.</p></li>
</ol>
```

1. dingbat. (1) Printing: Any typographical ornament not further specified. (2) Old TV Shows: What Archie Bunker would call his wife, Edith.

Checkoff ordered lists

Sometimes when describing procedures, you need to give your reader a checkoff space as an aid to ensure that they perform every step. The checkoff list is an ordered list; you get it adding the OLTYPE=CHECKOFF attribute to your OL tag. For example:

- ___ 1. Verify that air pressure is in normal range.
- ___ 2. Verify that fuel level is in safe zone.
- ___ 3. Verify that water valve is in open position.

Here's the coding:

```
<ol oltype="checkoff">  
<li>Verify that air pressure is in normal range.</li>  
<li>Verify that fuel level is in safe zone.</li>  
<li>Verify that water valve is in open position.</li>  
</ol>
```

Customer setup lists

Customer setup lists use the word "Step" in the list item, together with the number. These lists also use the OLTYPE attribute on the ordered list tag (OL), with a value of STEP. For example:

- Step 1. Open the carton.
- Step 2. Remove the top layer of packing material.
- Step 3. Take the stuff out of the box.
- Step 4. Give the box to the kids to play with, while you proceed with the next step.

Here's its coding:

```
<ol oltype="step">  
<li>Open the carton.</li>  
<li>Remove the top layer of packing material.</li>  
<li>Take the stuff out of the box.</li>  
<li>Give the box to the kids to play with,  
while you proceed with the next step.</li>  
</ol>
```

You can also have checkoff-setup lists. These lists also use the OLTYPE attribute with a value of CHECKOFFSTEP. For example:

- ___ Step 1. Fold along dotted line C.
- ___ Step 2. Insert tab B into slot A.
- ___ Step 3. Throw these instructions out the window.

Here's its coding:

```
<ol oltype="checkoffstep">  
<li>Fold along dotted line C.</li>  
<li>Insert tab B into slot A.</li>  
<li>Throw these instructions out the window.</li>  
</ol>
```

Continuing ordered lists

Sometimes, you may have lists that need to continue around table cells or even from division to division. The ordered list (OL) tag has the attributes SEQ, ID, and SEQID to allow you to have an ordered list continue from where a previous list

left off. For example, here's a continued checkoff list :

Formatted Example

- ___ 1. Open the carton.
- ___ 2. Remove the top layer of packing material.
- ___ 3. Take the stuff out of the box.

The first list is ended — honest, this is not a bridge. The list then continues:

- ___ 4. Fold along dotted line C.
- ___ 5. Insert tab B into slot A.

End of Formatted Example

Here's its markup:

```
<ol seq="start" oltype="checkoff" id="swingsets">
<li>Open the carton.</li>
<li>Remove the top layer of packing material.</li>
<li>Take the stuff out of the box.</li>
</ol>
<p>The first list is ended — honest, this is
not a bridge. The list then continues:</p>
<ol seqid="swingsets" seq="end" oltype="checkoff">
<li>Fold along dotted line C.</li>
<li>Insert tab B into slot A.</li>
</ol>
```

Definition lists

Definition lists are a particular kind of list you can use when you want to pair a term or phrase with a description of it.

Here's a formatted example of a definition list:

gopher

A burrowing rodent that feeds on roots of plants.

lawn

Gopher highway.
Can be identified by dinner-plate-sized mounds of dirt where grass used to be.

agapanthus

Lovely flowering plant, the roots of which are the preferred food of gophers.

If your flourishing agapanthus suddenly keels over, it means a gopher has had a feast.

Here's its coding:

```
<dl>
<dlentry><term>gopher</term>
<defn>A burrowing rodent that feeds on roots of plants.
</defn>
</dlentry>
<dlentry><term>lawn</term>
<defn>Gopher highway. <p>Can be identified by dinner-plate-sized
mounds of dirt where grass used to be.</p></defn>
</dlentry>
<dlentry><term>agapanthus</term>
<defn>Lovely flowering plant, the roots of which are
```

```

the preferred food of gophers. <p>If your flourishing
agapanthus suddenly keels over, it means a gopher
has had a feast.</p></defn>
</dentry>
</dl>

```

You can use the TERMWIDTH attribute to determine the indentation size of the definition list. The valid choices are: small (.5 inch, the default), medium (1 inch), large (2 inches), and 1 (1-character) and 2 (2-characters).

Small Here's a sample of the small setting.

Medium Here's a sample of the medium setting.

Large Here's a sample of the large setting.

1 Here's a sample of the 1-character setting.

12 Here's a sample of the 2-character setting.

Definition lists can also have headings; for example:

Setting Description

Low A good setting for simmering soups.

Medium After the water has boiled, use this setting for cooking the spaghetti.

High Use this setting to get water boiling fast.

This is done with the TermHd and DefnHd tags. Here's the source:

```

<dl><termhd>Setting</termhd>
<defnhd>Description</defnhd>
<dentry><term>Low</term>
<defn>A good setting for simmering soups.</defn>
</dentry>
<dentry><term>Medium</term>
<defn>After the water has boiled, use this setting
for cooking the spaghetti.</defn>
</dentry>
<dentry><term>High</term>
<defn>Use this setting to get water boiling fast.
</defn>
</dentry>
</dl>

```

You can also group terms together using the DLBlk (definition block) tag. You can use DLBlk to create logical groups within a long definition list, or use two DLBlks with a Bridge between them to highlight some relationship between groups of entries. For example:

Cat A house pet that purrs when happy.

Dog A house pet that wags its tail when happy.

Fish A house pet with scales that swims.

Turtle A house pet with scales that swims and walks slowly.

Here's the source:

```

<dl>
<dlblk>
<dentry><term>Cat</term>
<defn>A house pet
that purrs when happy.</defn></dentry>

```

```

<dentry><term>Dog</term>
<defn>A house pet that wags
its tail when happy.</defn></dentry>
</d1blk>
<d1blk>
<dentry><term>Fish</term>
<defn>A house pet
with scales that swims.</defn></dentry>
<dentry><term>Turtle</term>
<defn>A house pet with
scales that swims and walks slowly.</defn></dentry>
</d1blk>
</d1>

```

Parameter lists

Parameter lists are used in programming documentation when you have to explain the elements of the programming syntax. Here's an example of a parameter list:

KEYWORD = DEFAULT | VALUE

This is the description of the parameter above. It could go on for many pages, if necessary. (Of course, that means we have a very complicated parameter to describe.)

KEYWORD2 = {ABC | XYZ}

[KEYWORD3 = GGG]

This description applies to the two parameters above. Often in examples of programming syntax, it is necessary to use symbols for the brackets and braces.

KEYWORD3

Here's a term that uses the syntax phrase (SYNPH); it allows you to use the same items as a syntax diagram.

Parameter lists are similar to definition lists. They involve three elements: ParmL (parameter list), Term, and Defn (definition). Term and Defn are used with other elements for the same function, including glossary and definition lists, among others. When doing parameter lists for programming syntax, you will also need to use these elements:

- PK (programming keyword)
- PV (programming variable)
- SYNPH (syntax phrase), with KWD (keyword), VAR (variable), and DELIM (delimiter)

The Term elements assume that what you are entering is a required or optional programming keyword. For a default programming keyword or programming variable, edit the attributes for these PK or PV elements, and set the OPTREQ attribute value to DEF.

The IBMIDDoc markup for the example parameter list is shown in the example that follows.

```

<parm>
<parm><term>KEYWORD = <pk optreq="DEF">DEFAULT</pk>|VALUE
</term>
<defn>This is the description of the parameter above.
It could go on for many pages, if necessary. (Of course,
that means we have a very complicated parameter to
describe.)</defn>
</parm>

```

```

+ <parm><term>KEYWORD2 = &lbrk;ABC|XYZ&rbrk;</term>
+ <term>&lbrk;KEYWORD3 = GGG&rbrk;</term>
+ <defn>This description applies to the two parameters
+ above. Often in examples of programming syntax, it
+ is necessary to use symbols for the brackets and braces.
+ </defn>
+ </parm>
+ <parm><term><synph><kwd>KEYWORD3</kwd></synph></term>
+ <defn>Here's a term that uses the syntax phrase (SYNPH);
+ it allows you to use the same items as a syntax diagram.
+ </defn>
+ </parm>
+ </parml>

```

| You can use the TERMWIDTH attribute to determine the indentation size of the
 | parameter list. The valid choices are: small, medium, and large. You can use the
 | TERMWIDTH attribute to determine the indentation size of the definition list. The
 | valid choices are: small (.25 inch, the default), medium (.5 inch), large (1 inch), and
 | 1 (1-character), and 2 (2-character).

+ **Small**

+ Here's a sample of the small setting.

+ **Medium**

+ Here's a sample of the medium setting.

+ **Large**

+ Here's a sample of the large setting.

| **Message and code lists**

| If you don't have any messages or codes to worry about, just skip this section and
 | go on.

| Both message and code lists use the MsgList element. A code list documents
 | numeric values that have specific meaning, for example error codes. A message list
 | documents text messages, which may each have a message number. Each entry in
 | a message or code list is contained in a MSG element.

| For the code list, use the Code element for the numeric value. For the message list,
 | use a MsgNum and MsgText element. If the message has no number, use just the
 | MsgText element. If the message text has a variable in it, you use the MV element
 | for the message variable. Use the MV element in both the message text and in any
 | descriptive text.

| After you have entered the code or the message number and the text, you use the
 | MsgItem element to contain the information about the message, according to the
 | class of the information. IBMIDDoc has a number of predefined classes of message
 | item information. These pre-defined classes have a generated message subheading.
 | These classes are:

Class	Default subheading text
<i>author-defined</i>	Author defined class using the MsgItemDef element (currently not supported by the output formatters).
DEST	Destination
XPL or EXPLANATION	Explanation
MODULE	Module
NUMBYTES	Number of Error Bytes
ORESP	Operator Response
PRESP	Programmer Response

	PROBD	Problem Determination
	SEVERITY	Severity
	SPRESP	System Programmer Response
	SYSACT	System Action
	URESP	User Response

| This is a sample message list:

	DJI7832E	This message is issued when no data set of the name <i>file-name</i> is found.		User Response: Search high and low for the data set.
	Explanation:	The processor could not locate the data set named <i>file-name</i> .		This message has no number
	Severity:	8		Explanation: This message has no message number; only text. These are really insidious because it makes finding the message very hard.
	Problem Determination:	You would appear to have a problem.		

| Here is the markup:

```
| <msglist>
| <msg>
| <msgnum>DJI7832E</msgnum>
| <msgtext>This message is issued when no data set of
| the name <mv>file-name</mv> is found.</msgtext>
| <msgitem class="xpl">
| <p>The processor could not locate the data set named <mv>
| file-name</mv>.</p>
| </msgitem>
| <msgitem class="severity">
| <p>8</p>
| </msgitem>
| <msgitem class="probd">
| <p>You would appear to have a problem.</p>
| </msgitem>
| <msgitem class="uresp">
| <p>Search high and low for the data set.</p>
| </msgitem>
| </msg>
| <msg>
| <msgtext>This message has no number</msgtext>
| <msgitem class="xpl">
| <p>This message has no message number; only text.
| These are really insidious because it makes finding
| the message very hard.</p>
| </msgitem>
| </msg>
| </msglist>
```

| Code lists are just the same, except that you use the Code element instead of MsgNum and MsgText.
| Code lists still use MsgItem elements. Here is a short code list:

| **123**

| **Explanation:** This is a simple code.

| Here's its coding:

```
| <msglist>
| <msg><code>123</code>
| <msgitem class="xpl">
| <p>This is a simple code.</p>
| </msgitem>
| </msg>
| </msglist>
```

Overriding the message list subheadings

The text generated in association with `MsgItem` classes (XPL, URESP, and the rest) is not always suitable to every type of document containing these lists. For instance, you might want to use the subheading “Cause” or “Reason” instead of “Explanation”, and “Recovery” instead of “User Response”. Some people prefer “Severity Code” to “Severity”. Some places don’t have system programmers, but do have administrators or supervisors.

To meet all of these requirements and still preserve some order in the chaos, the `MsgItemDef` element can be used to override the default text that corresponds to the `MsgItem` classes. With these attributes you can specify the text that you want printed when the tag is used.

Here is a sample message list with changed headings. The heading are overridden for this list only; because the `MsgItemDef` tags are within the message list.

A12 **Closet full: Insufficient storage to proceed.**

Why: There are too many clothes in the closet.

What to do: Remove some clothes from the closet and restart.

Here is its coding:

```
<msglist>
  <msgitemdef classname="xpl"><title>Why</title></msgitemdef>
  <msgitemdef classname="uresp"><title>What to do</title>
</msgitemdef>
  <msg>
    <msgnum>A12</msgnum>
    <msgtext>Closet full: Insufficient storage to proceed.
  </msgtext>
    <msgitem class="xpl">
      <p>There are too many clothes in the closet.</p>
    </msgitem>
    <msgitem class="uresp">
      <p>Remove some clothes from the closet and restart.
    </p>
    </msgitem>
  </msg>
</msglist>
```

Note that when you use the class attribute, you should pick a value that is close in intent to the original meaning of the tag. That is, if you want to print “Reason”, use the XPL class; don’t use one — for instance, NUMBYTES — that is totally unrelated to the text you are printing. Do this because you may have future applications for this text, such as extracting from a text data base all messages and their explanations. If you don’t preserve the meaning of these tags, these future applications won’t be possible.

To globally override the message headings, use the `MsgItemDef` tags in the document prolog. For example:

```
<prolog>
  ...
  <propdefs>
    <msgitemdef classname="xpl"><title>Why</title></msgitemdef>
    <msgitemdef classname="uresp"><title>What to do</title>
  </msgitemdef>
  </propdefs>
  ...
</prolog>
...
```

```

| <msglist>
| <msg>
| <msgnum>A12</msgnum>
| <msgtext>Closet full: Insufficient storage to proceed.
| </msgtext>
| <msgitem class="xpl">
| <p>There are too many clothes in the closet.</p>
| </msgitem>
| <msgitem class="uresp">
| <p>Remove some clothes from the closet and restart.
| </p>
| </msgitem>
| </msg>
| </msglist>

```

Compacting lists

Compact specifies that you do not want a blank line between each list item. Compact lists are only available in Xyvision, BookMaster, HTML, and IPF. Documents written in Xyvision automatically have a half a line space. Compact applies only to the space between list items and not to any space between paragraphs within a list item.

To specify a list as compact, use the LineSpace attribute. For example:

```
<ul linespace="compact">
```

The lists you can compact are: DL, OL, UL, GL, Msglist, Parml, and Notel.

Grouping list items

You can use list item block (LIBlk, DLBlk, and ParmBlk) elements to contain groups of similar items. LIBlk is for list items, DLBlk is definition list items, and ParmBlk is for parameter list items. In the example that follows, hardware and software are grouped into blocks of list items.

For example:

1. 1 GIG SCSI-2 Hard Disk
2. 32 MB RAM
3. 128-Bit 8MB VRAM Video
4. 21-Inch Monitor
5. Great Word Processor
6. Best Multimedia App
7. Voice Mail

Here's its markup:

```

<ol>
<liblk>
<li>1 GIG SCSI-2 Hard Disk</li>
<li>32 MB RAM</li>
<li>128-Bit 8MB VRAM Video</li>
<li>21-Inch Monitor</li>
</liblk>
<liblk>
<li>Great Word Processor</li>
<li>Best Multimedia App</li>
<li>Voice Mail</li>
</liblk>
</ol>

```

Processing Note

For Xyvision processing, if you want to have all the text in the LIBLK kept on the same page, use the attribute `style="xpp:(keep)"`. Be cautious when using this feature. If the text does not fit on a page, you may get a formatting error. Remove the style or shorten the content to have the pages print correctly.

Separating or bridging list items

Sometimes in an ordered list you want to break the list for some explanatory material and then resume the numbering where you left off. You do this with the Bridge element.

Suppose you wanted to do this:

1. Saute the shallots and chopped mushrooms until the shallots are tender and the liquid from the mushrooms has cooked away.
2. Brown the sausage and add to the mushroom mixture.

The above may be prepared several hours in advance and refrigerated. Then, 30 minutes before serving time, finish the dish

3. Mix one can of tomato sauce with the mushroom and sausage mixture and bring to a slow simmer.
4. Add the heavy cream and immediately pour into a casserole.
5. Pop into 350-degree oven for 15 minutes.

Here's its markup:

```
<ol>
<li>Saute the shallots and chopped mushrooms until
the shallots are tender and the liquid from the mushrooms
has cooked away.</li>
<li>Brown the sausage and add to the mushroom mixture.
</li>
<bridge><p>The above may be prepared several hours in
advance and refrigerated. Then, 30 minutes before
serving time, finish the dish</p></bridge>
<li>Mix one can of tomato sauce with the mushroom
and sausage mixture and bring to a slow simmer.
</li>
<li>Add the heavy cream and immediately pour into
a casserole.</li>
<li>Pop into 350-degree oven for 15 minutes.</li>
</ol>
```

Chapter 4. Highlighting, Citing, Noting, and Quoting

IBMIDDoc provides several different ways to highlight text and contains many ways to create notes, annotations, and footnotes. This chapter introduces each of these.

These element types include:

- Phrases (see “Highlighting”)
- Citations (see “Simple title citations” on page 37)
- Notes (see “Notes” on page 37)
- Note lists (see “Note lists” on page 38)
- Footnotes (see “Footnotes” on page 38)
- Quotes and long quotes (see “Quotes and excerpts” on page 39)
- The Perils of Processing (see “The perils of processing: Attention, caution, and danger” on page 40)
- Labeled boxes (see “Labeled boxes” on page 40)
- Author notes (see “Annotations” on page 41)
- Trademarks (see “Trademarks” on page 42)
- Qualifications (see “Qualifying information” on page 41)

Highlighting

You’ve already seen many examples in this book of things that are highlighted. By highlighting, we mean emphasizing the text by setting it in a different font or perhaps underscoring it. Many phrase styles are supported. They are mapped as `<PH style=attribute>`. The style attribute values include:

- `base`
- **`bold`**
- *`italic`*
- ***`bold italic`***
- `underlined`
- ^{`superscript`}
- _{`subscript`}
- `monospaced`
- `SMALLCAPS`. Note that YOU need to do the uppercase conversion yourself. This is because not all languages do proper uppercase conversion of lowercase letters.
- **`underlined bold`**
- *`underlined italic`*
- ***`underlined bold italic`***
- `UNDERLINED SMALLCAPS`

These values are only valid on the PH tag’s style attribute. Here’s a sample:

Formatted Example

Hey there! This is very important! Don't go out in the rain without your galoshes!

End of Formatted Example

Here's its markup:

```
<ph style="Bold Italic">Hey there!</ph>
This is <ph style="Underlined Bold">very</ph>
<ph style="Bold">important</ph>! Don't go out in the
<ph style="Italic">rain</ph> <ph style="Underlined Bold Italic">
without your galoshes</ph>!
```

Other elements included for denoting values and phrases are shown in Table 1.

Table 1. Phrase types

Description	Markup	Result
APL	<apl>APL</apl>	<i>APL</i>
binary (bin)	<bin>0101</bin>	B'0101'
character (char)	<char>character</char>	"character"
decimal (dec)	<dec>123</dec>	123
example phrase (xph)	<xph>example</xph>	example
hexadecimal (hex)	<hex>10FE</hex>	X'10FE'
marked deletion (md)	<md rev="rev1">marked deletion</md>	marked deletion
message variable (mv)	<mv>message variable</mv>	<i>message variable</i>
number with specified base (num)	<num base="3">120</num>	120
octal (oct)	<oct>013 736</oct>	O'013 736'
reference key (refkey)	<refkey>reference key</refkey>	reference key
term	<term>term</term>	term

You can also nest highlighted phrases — that is, put one kind inside another. The formatter causes the nested highlighting to inherit the highlighting of the previous style. In the following example, the italic and underlined text are also bold because the whole sentence is marked bold:

Speak *softly* and carry a BIG stick.

Here's its markup:

```
<ph style="Bold">Speak <ph style="Italic">softly</ph>
and carry a <ph style="Underlined">BIG stick
</ph></ph>.
```

Simple title citations

While IBMIDDoc provides extensive bibliographic markup (see “Chapter 13. Bibliographies and citations” on page 119), sometimes you just need a simple inline title citation. For this you use the CIT element (and some others). For example, here’s a reference to a non-IBM book:

Formatted Example

Huckleberry Finn, by Mark Twain, is a most excellent book.

End of Formatted Example

Here’s its markup:

```
<cit><bibentry><doctitle><titleblk><title>Huckleberry
Finn</title></titleblk></doctitle></bibentry></cit>,
by Mark Twain, is a most excellent book.
```

Here’s a reference to an IBM book:

Formatted Example

The *BookMaster User’s Guide* is the book to emulate.

End of Formatted Example

Here’s its markup:

```
The <cit><ibmbibentry><doctitle><titleblk><title>
BookMaster User's Guide</title></titleblk></doctitle>
</ibmbibentry></cit> is the book to emulate.
```

Notes

The Note element contains a single note. For example:

Note: Thinking of a seashore, green meadow, or cool mountain overlook can help you to relax and be more patient.

Here’s its coding:

```
<note><notebody>Thinking of a seashore, green meadow,
or cool mountain overlook can help you to relax and
be more patient.</notebody></note>
```

If you want the word “Note” to be something else, use the TITLE element; for example:

Tip: Don’t sit under the apple tree with anyone else but me.

Here’s its coding:

```
<note><title>Tip</title>
<notebody>Don't sit under the apple tree with anyone else but me.
</notebody>
</note>
```

+ Note lists

+ The NoteList element contains an ordered list of notes. The list is ordered because
+ there is usually a priority to the notes. For example:

+ **Notes:**

- + 1. Make a To Do list
- + 2. Prioritize sensibly
- + 3. Avoid interruptions where possible
- + 4. Check on your progress toward monthly goals
- + 5. Plan for the next work week
- + 6. Do something for the fun of it
- + 7. Spend some quality time with your pet

+ Here's its coding:

```
+ <notelist>  
+ <li>Make a To Do list</li>  
+ <li>Prioritize sensibly</li>  
+ <li>Avoid interruptions where possible</li>  
+ <li>Check on your progress toward monthly goals</li>  
+ <li>Plan for the next work week</li>  
+ <li>Do something for the fun of it</li>  
+ <li>Spend some quality time with your pet</li>  
+ </notelist>
```

+ Groups of notes can be organized into blocks using the LIBlk element.

+ You can also use your own title instead of "Notes" by adding a Title. For example:

+ **Watch out for these:**

- + 1. things that go bump in the night
- + 2. green eggs and ham

+ Here's is markup:

```
+ <notelist><title>Watch out for these</title>  
+ <li>things that go bump in the night</li>  
+ <li>green eggs and ham</li>  
+ </notelist>
```

+ You can compact, bridge, and group list items together. This is done in the same
+ way as ordered lists; see "Chapter 3. All kinds of lists" on page 23.

+ Footnotes

+ The FN element is used to annotate text with notes that are part of the narrative
+ content of the document, but that are not appropriate for inclusion inline with the
+ document text. The information contained in the FN element is associated with its
+ containing element.

+ **Formatted Example**

+ There's a footnote² around here somewhere.

+ **End of Formatted Example**

+ Here's its coding:

+ `<p>There's a footnote<fn>While some folks do not like`

+ `footnotes; they sometimes contain a nugget of priceless`

+ `lore. Did you know IBMIDDoc's grandmother was named`

+ `ISIL?</fn> around here somewhere.</p>`

+ You can also define a footnote and use it in multiple places. First, define the

+ footnote (FN tag) and give it an ID attribute. Then, use the FN tag with a REFID

+ attribute to refer to that footnote.

+ **Formatted Example**

+ Here's a footnote that is used more than onceHere's a sentence³ that uses several

+ footnotes³.

+ **End of Formatted Example**

+ Here's its coding:

+ `<p>`

+ `<fn id="multiple">Here's another little footnote.</fn>`

+ `Here's a sentence<fn refid="multiple">`

+ `that uses several footnotes<fn refid="multiple">.`

+ `</p>`

+ Quotes and excerpts

+ IBMIDDoc provides for two different kinds of quotations — inline quotations and

+ excerpts (which we call “long quotations” although it really has nothing to do with

+ the length). For simple inline quotations, use the Q element. These can be nested.

+ For example, this contains two quotes, one inside the other. The formatter knows

+ to automatically switch from double quotes to single quotes (in hardcopy anyway).

+ **Formatted Example**

+ George said; “She said ‘Yes!’ to me.”

+ **End of Formatted Example**

+ Here's its coding:

+ `George said; <q>She said <q>Yes!</q> to me.</q>`

+ For long quotations or excerpts, use the LQ element. Who remembers this quote

+ from History class:

+ The only thing we have to fear is fear itself.

2. While some folks do not like footnotes; they sometimes contain a nugget of priceless lore. Did you know IBMIDDoc's grandmother was named ISIL?

3. Here's another little footnote.

Here's its coding:

```
<lq>The only thing we have to fear is fear itself.
</lq>
```

Labeled boxes

Labeled boxes are a special style of paragraph block. For example:

Here's my cute little box

Here's something that I'm really proud of.

Here's its coding:

```
<pblk style="lblbox">
<title>Here's my cute little box</title>
<p>Here's something that I'm really proud of.</p>
</pblk>
```

Beware of over-using these; and of trying to put too much information into them.

The perils of processing: Attention, caution, and danger

These elements are used to contain information about situations that can dangerous to people, equipment, or data.

Attention

Use an Attention notice to indicate the possibility of damage to a program, device, system, or data.

Warning

Use an Attention notice to indicate the possibility of damage to a program, device, system, or data.

Caution

Use a Caution notice to call attention to a situation that is potentially hazardous to people because of some existing condition. For example, you might use a Caution notice to warn about the hazard of paper cuts when a fresh ream of paper is opened.

Danger

Use a Danger notice to call attention to a situation that is potentially lethal or extremely hazardous to people. For example, after a computer side panel is removed, exposed high-voltage wires might be lethal.

Attention: Here's a way to get someone's attention.

CAUTION:

Watch out for these!

DANGER

Really watch out for these!

Here are their codings:

```
<attention>Here's a way to get someone's attention.</attention>
<caution>Watch out for these!</caution>
<danger>Really watch out for these!</danger>
```

Annotations

The Annot element is used to contain comments about the content of its containing element. These comments can be notes to reviewers, other writers, editors, vendors, and so forth. Annotations do not contain comments you want to appear in a final draft of your document. Annotation content is not part of the narrative text of your document.

Annot has NO formatting associated with it. All formatting comes from the markup within the annotation body.

Processing Note

The Xyvision formatter does not print annotations under any circumstances. BookMaster hides the content by default, unless you specify this runtime option:

```
sysvar(A yes)
```

Migration Note

Do not use the Annot element to comment out information. Because Annot is an element in the document hierarchy, it cannot span structures in the document.

If you want to selectively print annotations, each Annot element should be contained in a marked section or use a PROPS value as shown below.

```
<P>Remove the cover from the system unit by unscrewing the tabs on the
rear of the unit.
<ANNOT PROPS="hide">
<TITLE>>System Test Note</TITLE>
<ANNOTBODY>
<P>
Please advise us of any discrepancies in the installation
instructions in this section.
</P>
</ANNOTBODY>
</ANNOT>
</P>
```

Migration Note

Any ANNOT tags used to comment out information in Bookmaster will be presented within a labeled box. You can delete these or convert them to SGML comments.

Qualifying information

When you have to qualify information as applying to a particular product or system in a book about multiple products or systems, there are several techniques you can use. One, of course, is simply to say, "If you are using Model 9, then....". Another method, for major differences, is to put some qualification in the section heading. Still another method is to use a formatting convention such as the one provided by the QualifDef (qualifying information definition) element and the QUALIF (qualifying information) attribute. This is suitable only for qualification at

the level of a paragraph or list item or greater. It is not suitable for single-phrase qualifications, or for qualifications of many pages.

You begin by defining your qualification in the document's prolog, using the QualifDef and Qualif elements. For example, this specifies two qualifications, one for Windows 99 and another for OS/2.5:

```
<qualifdefs>
  <qualif id="win99" ident="use">
    <title>Windows/99</title>
    <desc>Windows/99 information</desc>
  </qualif>
  <qualif id="os25" ident="use">
    <title>OS/2.5</title>
    <desc>OS/2.5 information</desc>
  </qualif>
</qualifdefs>
```

Here are sample paragraphs for each type of qualification:

Windows/99

This operating system is great for home use.

End of Windows/99

OS/2.5

This operating system is the business-oriented, industrial-strength operating system.

End of OS/2.5

Here is their markup:

```
<p qualif="win99">This operating system is great for
home use.</p>
<p qualif="os25">This operating system
is the business-oriented, industrial-strength operating system.</p>
```

QUALIFs cannot be nested. Also, the text must be short enough to fit within the column along with the words "end of," plus the blanks, corners, and at least a little of the line. This can be a tight fit in a double-column layout. If you use the Qualif inside a boxed figure or a ruled table, the corners may overlay the rules.

+ Trademarks

The TM tag identifies the trademark terms in your source by surrounding the trademark term or phrase. This tag has attributes that are not translated; they contain no "MRI". The TM attributes contain information for the author. The TMType attribute creates the appropriate trademarking character after the term or phrase. The ID Workbench files IDDIRTM.LST or IDTMSCAN.LST list the trademarks and the attributes needed for the TM tag. Use the Adept or Frame+SGML editor to insert these tags and attributes. Currently, the trademark symbols are not supported in HTML format.

For more information about trademarks, see the *ID Workbench Getting Started and User's Guide*.

Chapter 5. Examples, artwork, and multimedia

In IBMIDDoc, all non-text objects are considered multimedia objects, including graphics. This chapter explains how to use these objects in IBMIDDoc. The elements discussed in this chapter include:

- Lines (see “Just plain lines”)
- Xmp (see “Examples of computer output” on page 44)
- LitData (see “Literal text data” on page 45)
- MMObj (see “Including artwork in documents” on page 45)
- CGraphic (see “Character graphics” on page 48)
- Screen (see “Screens” on page 49).

Just plain lines

Use the Lines element to contain text that has record ends, or boundaries, that need to be preserved when presented to the document user.

The LINES element allows you to control where lines break. That is, within the content of the LINES element, IBMIDDoc will end each output line at the same point where you ended the input line in the markup. In SGML terms, the record ends must be respected. This occurs whether you include text characters within the Lines element, or if you reference an entity that contains text characters or a graphic.

You can include information within LINES by either of the following:

- The Lines tag contains text characters that should be presented “as-is”.
- Using the OBJ attribute to name an entity containing the text or other character data to be presented

The next example illustrates using a Lines element to contain unflowed text:

a partridge in a pear tree
two turtledoves
three French hens
four calling birds
five golden rings
six geese a-laying
seven swans a-swimming
eight maids a-milking
nine ladies dancing
ten lords a-leaping
eleven pipers piping
twelve drummers drumming

Here's its markup:

```
<LINES>  
a partridge in a pear tree  
two turtledoves  
three French hens  
four calling birds  
five golden rings  
six geese a-laying
```

```
|
|       seven swans a-swimming
|       eight maids a-milking
|       nine ladies dancing
|       ten lords a-leaping
|       eleven pipers piping
|       twelve drummers drumming
| </LINES>
```

```
|
| This next example shows how you would code the Lines tag to use the OBJ
| attribute to insert a file named "lines.txt":
| <lines obj="samplelines">
```

```
|
| Here's the declaration:
| <!ENTITY linestext SYSTEM "lines.txt" ndata linespec>
```

```
+
+ The lines typically have a space before and after them, except at the start of a page,
+ column, or table entry.
```

| Examples of computer output

```
|
| The Xmp element typically contains an example of computer input or output.
| Sometimes you may have a very long example — possibly running for several
| pages. In this case, you have to tell IBMIDDoc that it is okay to break the example
| at any point after a specified number of lines have printed. You do this with the
| KEEP attribute. This is illustrated in the example that follows.
```

```
| <XMP STYLE='BKM:(KEEP="10")'>
| 10 LET A = B
| 20 IF A GT C THEN GO 40
| 30 LET A = C
| 40 PRINT A, C
| </XMP>
```

```
|
| You can also scale the examples incase you have a super-wide listing. In
| BookMaster and XyVision output, the scaling can be computed. For BookMaster
| you specify the width of the example in characters and use the value
| SCALE=AUTO. XyVision reads the BKM width. For example, this shows how to
| code the overrides for a 132-column listing and have the scaling automatically
| computed:
| style=bkm:(scale=auto width=132 keep=20)
```

```
|
| As with Lines, you can also use the OBJ attribute to include the example. In the
| example that follows, the OBJ attribute is the entity name sampcprg. This entity is a
| small sample C program named sampcprg.c. The XMPs element uses the content of
| this entity as its content.
```

```
|
|       :
|       :
| <!ENTITY  sampcprog SYSTEM "sampcprg.c" NDATA C>
|       :
| <XMP OBJ="sampcprg">
```

```
+
+ The example typically has a space before and after, except at the start of a page,
+ column, or table entry.
```

Literal text data

Literal data can be used to contain special information or code in which SGML markup is not recognized. It may also refer to the content of other files whose content will not be processed as SGML markup. Examples of this type of data include character translation, special code pages, and samples of programming code.

For example, a sample C++ program could be contained in the LitData element:

```
<xmp>
<LITDATA>
// testprog.cpp
#include <iostream.h>
int main(void)
{
    ...
}
</LITDATA>
</xmp>
```

This same program could be referenced using the OBJ attribute that refers to an entity that contains the program, as shown in the next example.

```
<!ENTITY testprg SYSTEM "testprg.c" ndata c>
:
<FIG>
<CAP>A basic C++ program.
<LITDATA OBJ="TESTPRG">
</FIG>
```

Including artwork in documents

In most cases, the artwork you include will be constructed using a graphic/image tool such as CorelDraw or Photoshop. It will be merged with the text during processing for the output device. All you have to do is specify the file type of the artwork that you want to include in your document; the formatter does the rest.

You use the MMObj and ObjRef elements to contain artwork such as images, vector graphics, encapsulated PostScript, or video clips. The processing and presentation systems in use determine the types of multimedia objects that are supported.

In order to use graphic objects in your SGML markup, you must declare them as entities using an entity declaration. Use the notation "graphics" on the declaration.

```
<!ENTITY bike system "bike.gif" ndata graphics>
```

In some cases, certain graphic formats are supported for in-line viewing during your editor session. Use the file extension GIF, JPG, TIF, or EPS (if your EPS file has a TIFF header) when you declare your graphic, and the editor will display the artwork.

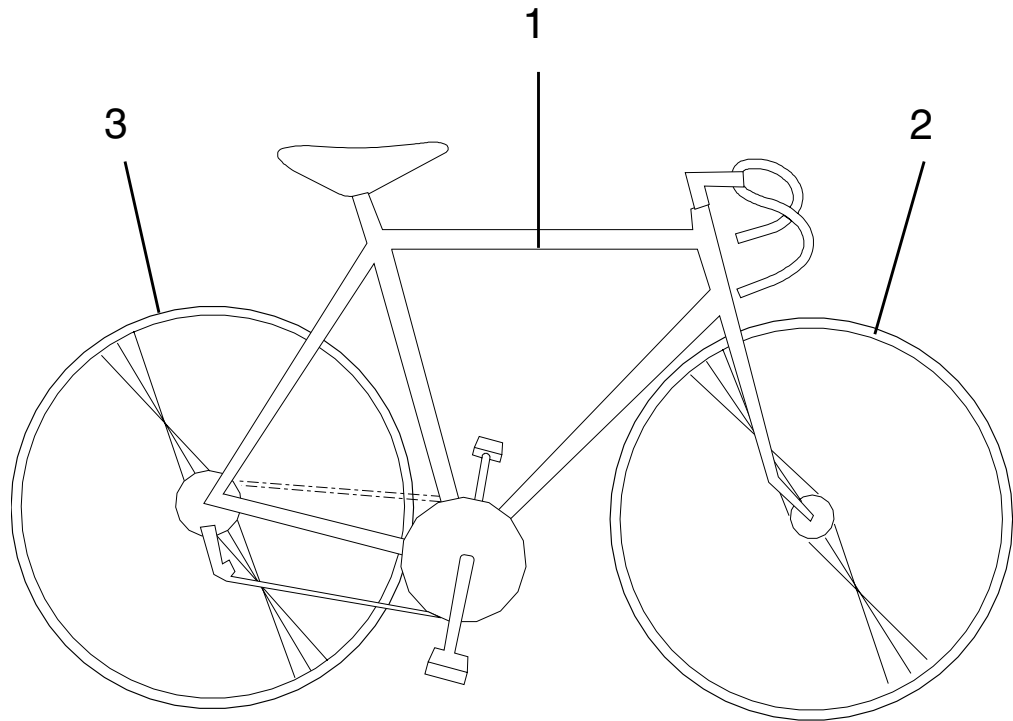
In the example that follows, the entity is defined first, and it is referred to later in the document by the OBJ attribute on the ObjRef element contained in the MMObj element. Here's the declaration and the markup for an illustration of a bike:

```

<!ENTITY bike system "bike.gif" ndata graphics>
...
<MMOBJ>
<OBJREF OBJ="bike">
<TEXTALT>This is a two-wheeled bicycle.</TEXTALT>
</MMOBJ>

```

And here's the bike:



The MMOBJ element also contains a TextAlt element. The TextAlt element contains a text description that is presented as an alternative to the artwork.⁴ The text appears in HTML when the user's mouse hovers over the artwork. So be sure to type something meaningful in the TextAlt; don't be embarrassed by entering something like "Fred, is this really true?" and then having that go out the door and on to a website.

Normally a single space precedes the artwork. You can use the setting `placement=runin` to have the artwork put inline in a sentence. You can also use `placement=runin` to have the artwork appear in the margin.

Creating graphic links

To create a graphic link for RTE, IPF, or HTML, you need to use the MMOBJLnk tag. The Linkend attribute specifies the link ID. The AreaDef element is not used (that is, you cannot create hotspots with it; the entire graphic becomes a link).

Here is an example that will make the graphic part1 link to the ID "newdiv":

4. Some say a picture is worth a thousand words; well, this is where you put the thousand words to express your picture to those that cannot view pictures.

```

<mmobj><objref obj="part1">
<mmobjlink linkend="newdiv">
<areadef coords="1 100">
</areadef></mmobjlink>
<textalt></textalt>
</mmobj>

```

Figures

Figures typically contain examples, text, or artwork; so it makes sense to talk about them here.

You can choose to have the figure formatted within the column or formatted the full width of the page. You can choose to have it set off with rules across the page, put in a box, or formatted with no frame at all. You can give it an identifier, which will allow you to make cross references to it (which we'll cover in a later chapter). You can give the figure a caption, which will also cause it to be listed in the figure list, if you have one. You can extend the figure caption with a figure description, too (only the caption itself goes in the figure list). And, if you have a very large figure, you can have it split into pieces with a caption that says "Part x of y" on each piece.

With BookMaster hardcopy output, you can either have it placed exactly where you entered it, or you can allow BookMaster to move it so as to fill out a page (the figure "floats" to the top or bottom — you say which — of the next column or page).

Here's a sample, simple figure:

```

Here are some lines
in the sample, simple figure.

```

Here's its markup:

```

<fig style="bkm:(place=inline width=column)">
<lines>Here are some lines
in the sample, simple figure.</lines>
</fig>

```

The values in the style attribute set to figure as column-wide, and inline; that is, the figure does not float to fill in a page with text.

Your figure can also have a caption or description.

To have a full-page figure, you specify this style override:

```

<fig style="bkm:(place=inline width=page)">
<lines>Here are some lines in a page-wide figure.</lines>
</fig>

```

You can choose to add a box around your figure, or to have lines or rules appear before and after your figure. You add these with the FRAME attribute. For example:

```

<fig frame="rules">

```

Or:

```

<fig frame="box">

```

Figure captions and descriptions

Your figures can have a short caption and an optional longer description. The caption can appear in a figure list at the beginning of your book. A good place for the figure list is after your table of contents. If your figure has a caption, you should also give it an ID; some processes (like BookMaster) complain about figures that have captions but are missing the ID. Put the ID on the fig tag, not on the caption. The caption should be entered like a heading, without ending punctuation.

Here's another sample figure. This one is page-wide and includes a caption:

Here are some lines
in the sample, simple figure.

Figure 4. Here's a sample, page-wide figure

Here's its markup; note the FIG tag has an identifier:

```
<fig id="samplefig" style="bkm:(place=inline width=page)">
<cap>Here's a sample, page-wide figure</cap>
<lines>Here are some lines
in the sample, simple figure.</lines>
</fig>
```

Here's another sample figure with both a caption and a description. You enter a description like a sentence, with punctuation. Note that the caption source has no punctuation; if any is needed, it is added by the formatter.

Here are some lines
in the sample, simple figure.

Figure 5. Here's a sample figure with a caption and description. This figure has a description. Note that descriptions have punctuations like sentences.

Here's its markup:

```
<fig id="samplefigdesc" style="bkm:(place=inline width=column)">
<cap>Here's a sample figure with a caption and description
</cap>
<desc>This figure has a description. Note that descriptions
have punctuations like sentences.</desc>
<lines>Here are some lines
in the sample, simple figure.</lines>
</fig>
```

Character graphics

Use the CGraphic element to contain a graphic created using character graphics, such as box and line characters.

Note: Ensure the cgraphics are external file entities. Having them inline increases the chances of data corruption.

CGraphic will often use the OBJ attribute to reference an entity that contains the actual cgraphic markup characters. The NOTATION attribute value is always LINESPEC.

```
<!ENTITY ILLUS01 SYSTEM "ILLS01.CHG" ndata linespec>
:
<CGRAPHIC OBJ="ILLUS01">
```

[illegible]

1
1
1
1
1

11

1

11

1
1
1

1

11

— 1 —

11

Math formulas

Mathematical formulas are contained by the Formula element. Only formulas created using the Script Mathematical Formula Formatter (SMFF) are supported at this time. This is only supported for BookMaster hardcopy output.

```
<FORMULA NOTATION="smff">  
integral from 0 to infinity of d x  
</FORMULA>
```

Chapter 6. Cross-referencing

At last we're going to tell you what all those ID attributes are for!

IBMIDDoc manages cross references to headings, figures, tables, items in ordered lists, and footnotes, indeed, to any spot in a document. Each of these types of cross references is done with the ID (identification) and REFID (reference identification) attributes — ID on the thing pointed to, and REFID on the thing doing the pointing.

Also, when you take your BookMaster source and create an online document, IBMIDDoc uses the REFID and ID attributes to set up hypertext links between cross references within and between documents. See "Chapter 11. All about linking" on page 109 for more information about interdocument cross-reference linking.

The ID (identification) attribute is a way of giving something a name that IBMIDDoc can use in providing a reference to that thing. You've already seen cases in this book of cross references to headings, although you may not have realized it at the time. For example, this is a cross reference to a heading:

Formatted Example

See "Chapter 3. All kinds of lists" on page 23.

End of Formatted Example

To get the cross reference, two things have to be done. First of all, the heading that we referred to was entered like this:

```
<d id="lst">
<dprolog><titleblk>
<title>All kinds of lists</title>
</titleblk></dprolog>
```

The "lst" is a name we made up. It can be any combination of letters and numbers as long as it is no more than 64 letters and numbers. The first character must be a letter; and the other characters can be a period (.) or a dash (-). You can use uppercase or lowercase letters. Each ID in the document must be unique. The other thing you have to do is enter the cross reference itself, using the XREF tag with a REFID attribute. It would look like this:

See <xref refid="lst">.

Because we used the same value for the REFID attribute as for the ID attribute of the heading we are referring to, IBMIDDoc knows which one we want. Because it also knows what page it is on, it supplies that, too. If the heading had been on the same page as the reference, IBMIDDoc would know that also, and it would not give the page number.

The target of a cross-reference should be to an ID on the outer container (for example, D, MSG, LE, FIG, TABLE) and not the title text or caption text.

Since online BookManager books don't have actual pages, cross references will refer to topic numbers. For online HTML and other information, the references just contains the heading.

Cross references can go in either direction; that is, the thing being referred to can come before or after the cross reference.

It is a good idea to pick descriptive names for your ID attributes; if you name things "chap1", "chap2", and so on, you will find that when you update the source file and insert, delete, and rearrange material, your names will be more confusing than useful in trying to keep track of what is going on.

There are times when you want to control the form of cross references using the FORM attribute, as described in "Controlling the form of cross references" on page 55.

You can use XREF to reference any heading level and to reference anything. These next topics show cross references to common elements:

- "Referencing a figure"
- "Referencing a table" on page 53
- "Referencing a list item" on page 53
- "Referencing anything at all" on page 54

Referencing a figure

To reference a figure, ensure the figure has a caption. Here's a sample figure that we're going to reference in a bit:

Formatted Example

A figure that is going to have a cross reference must also have a caption.
Figure 6. Captioned figure for cross reference

End of Formatted Example

Here's the markup for our little figure above:

```
<fig id="littlefig">
<cap>Captioned figure for cross reference</cap>
<p>A figure that is going to have a cross reference
must also have a caption.</p>
</fig>
```

Now we can code and XREF like this:

See <xref refid="littlefig"> for a sample of a figure reference.

and the result is this:

Formatted Example

See Figure 6 for a sample of a figure reference.

End of Formatted Example

Referencing a table

Cross referencing a table is just like referencing a figure. The table must have a caption. Here's a sample table:

Formatted Example

Table 2. Captioned table for cross reference

A table that is going to have a cross reference...	
	... must also have a caption.

End of Formatted Example

Here's the markup for our little table above:

```
<table pgwide="0" id="sampletable">
<cap>Captioned table for cross reference</cap>
<tgroup cols="2">
<colspec colname="col1">
<colspec colname="col2">
<tbody><row>
<entry colname="col1">A table that is going to have
a cross reference...</entry>
<entry colname="col2"></entry>
</row><row>
<entry colname="col1"></entry>
<entry colname="col2">... must also have a caption.
</entry></row></tbody></tgroup></table>
```

Now we can code and XREF like this:

See <xref refid="sampletable"> for a sample of a table reference.

and the result is this:

Formatted Example

See Table 2 for a sample of a table reference.

End of Formatted Example

Referencing a list item

Another common phenomenon in the books we produce is the cross reference to an item in an ordered list. To do this, we put an ID attribute on the LI (list item) tag for the item we want to point to, and use an XREF tag with a REFID attribute to do the pointing. Here's a sample of some new tax instructions from the U.S. Department of Treasury:

Formatted Example

1. If the amount on line 37 is greater than the lesser of lines 5 and 6, go to step 5 on page 54.
2. Enter the sum of line 5 and line 37. If this exceeds your total annual income before deductions, go to step 6 on page 54.

3. If line 32 less the difference of lines 73 and 74 on Schedule C is greater than line 36 plus line 17 of Schedule A and you are under 65 years of age, go to step 1 on page 53, where you will be in a loop until you are 65.
4. Use table 30-C to compute the number of bathrooms in your house and enter on line 56.
5. Enter the root-mean-square of line 14.
6. Sign form and mail with remittance.

End of Formatted Example

Here's its markup:

```
<ol>
<li id="ageloop">If the amount on line 37 is greater
than the lesser of lines 5 and 6, go to
step <xref refid="squareit">.</li>
<li>Enter the sum of line 5 and line 37. If this exceeds
your total annual income before deductions, go to
step <xref refid="mailit">.</li>
<li>If line 32 less the difference of lines 73 and
74 on Schedule C is greater than line 36 plus line
17 of Schedule A and you are under 65 years of age,
go to step <xref refid="ageloop">, where you will
be in a loop until you are 65.</li>
<li>Use table 30-C to compute the number of bathrooms
in your house and enter on line 56.</li>
<li id="squareit">Enter the root-mean-square of line
14.</li>
<li id="mailit">Sign form and mail with remittance.
</li>
</ol>
```

Referencing anything at all

Although we've given you a lot of ways to create cross references, there are times when none of those ways exactly meets your requirements. So IBMIDDoc has IDs on all its tags, which allow you to identify any spot in your document, by page number, and to refer to it from another place.

A tag with an ID attribute has no effect on the formatting of the text around it, but to get the result you want, it should be placed in the same places that are good for index entries. These are described in "Where to put index entries" on page 101. You can also specify some text that is associated with this tag, using the XREFTEXT (cross-reference text) attribute on that tag. If there was no XREFTEXT, then the XREF gives you just the page number of the tag. If there was XREFTEXT, then XREF prints that text, followed by "on page" and the page number.

For example, here's a paragraph we want to reference:

Here's my little paragraph that I want to reference.

Here's its markup, including xref text:

```
<p id="samplepara" xreftext="My little paragraph">
Here's my little paragraph that I want to reference.
</p>
```

Then, to reference that, we would code the following:

See <xref refid="samplepara"> for a small bit of information.

+ And we would get this result:

+ See “My little paragraph” on page 54 for a small bit of information.

+ **Controlling the form of cross references**

+ The “normal” form of a cross reference, as shown in the examples in the preceding
+ sections, is not always exactly suitable to your needs. IBMIDDoc has a FORM
+ attribute on the XREF tag that allows you to control what is generated. Table 3
+ shows the supported values of the FORM attribute for hardcopy output, for the
+ typical things you cross-reference.

+ *Table 3. XREF forms for hardcopy output*

Referenced item	Form=				
	Normal ¹	Full	Text	Location	Number
Heading	“Chapter 1. Title text” [on page 3]	“Chapter 1. Title text” on page 3	Chapter 1. Title text	3	Chapter 1
Figure	Figure 1 [on page 3]	Figure 1 on page 3	Figure 1	3	1
Table	Table 1 [on page 3]	Table 1 on page 3	Table 1	3	1
List item	1 [on page 3]	1 on page 3	1	3	1
Anything (with xref text)	“Anything” [on page 3]	“Anything” on page 3	Anything	3	
¹ The [on page 3] only appears if the referenced item is on a different page.					

Chapter 7. Creating IBMIDDoc Tables

IBMIDDoc supports the CALS table elements, with a few modifications for IBM-specific usage. This section describes fundamental IBMIDDoc table concepts, and gives examples of markup for several tables.

Migration Note

For BookMaster tables that include DVCF text that will be reused across a number of documents, or will be output in many different formats, use the IBMIDDoc modular information elements instead of table elements.

IBMIDDoc Table Markup Concepts

You can use IBMIDDoc to create a wide variety of tables. From simple tables:

Table 4. Simple table

A	B	C
D	E	F

To complex tables:

Table 5. Complex table

A	B	C
	D	E
F	G	

You can select several different combinations of rules and framing. You can have a table heading and a table caption. You can specify the alignment and formatting of text in your table. You can combine several table definitions within a single table. But you don't have to use all of the power of table markup every time you want to create a table. You can get handsome tables with fairly simple markup. So we'll take things one step at a time, beginning with simple table markup and moving on to the more advanced stuff later.

IBMIDDoc tables are contained in a Table element. The Table element then can contain TGroup elements. In most cases, you'll assign values to the TGroup's attributes that define the structure and layout of the table.

Usage Note

In most cases, you will probably use your graphical table editor to create a table. The graphical table editor does not currently support the TFOOT element. You can manually add this element when the rest of your table is complete.

Creating simple tables

Tables consist of **cells**, arranged in **rows**. Here is a simple example:

Table 6. A simple example

Row 1, Cell 1	Row 1, Cell 2	Row 1, Cell 3; here's a little more text than the other cells have
Row 2, Cell 1	Row 2, Cell 2	Row 2, Cell 3

Each little box in this table is a cell. This table consists of two rows, with three cells in each. You can, of course, have more or fewer rows and cells, so this basic form can take care of a lot of your table requirements. A table must have at least one row (in addition to any headings, footings, and captions) to produce any output, and each row must have at least one cell.

The markup for this table is quite simple:

```
<table><cap>A simple example</cap>
<tgroup cols="3">
  <colspec colname="col1">
  <colspec colname="col2">
  <colspec colname="col3">
  <tbody>
    <row>
      <entry colname="col1">Row 1, Cell 1</entry>
      <entry colname="col2">Row 1, Cell 2</entry>
      <entry colname="col3">Row 1, Cell 3; here's a little more text
      than the other cells have</entry>
    </row>
    <row>
      <entry colname="col1">Row 2, Cell 1</entry>
      <entry colname="col2">Row 2, Cell 2</entry>
      <entry colname="col3">Row 2, Cell 3</entry>
    </row>
  </tbody>
</tgroup>
</table>
```

We began the table with the TABLE tag and put in a caption with CAP. Next comes TGROUPE with the number of columns in the table, and COLSPECs indicating the column names. The table content starts with the TBODY tag. We then started specifying rows and cells with the ROW and ENTRY tags. The text within each cell is formatted the same way regular body text is formatted (we'll look later at how to redefine this.) Finally, we ended the table with the proper ending tags. It's that easy.

Specifying table column widths

If you want your columns to have different widths, like this:

Table 7. Simple table with different column widths

a	b	c
---	---	---

You use the COLSPEC tag and the COLWIDTH attribute, like this:

```
<table><cap>Simple table with different column widths
</cap>
<tgroup cols="3">
  <colspec colnum="1" colname="col1" colwidth="1*">
  <colspec colnum="2" colname="col2" colwidth="1*">
```

```

+      <colspec colnum="3" colname="col3" colwidth="2*">
+      <tbody>
+      <row>
+      <entry colname="col1">a</entry>
+      <entry colname="col2">b</entry>
+      <entry colname="col3">c</entry>
+      </row>
+      </tbody>
+      </tgroup>
+      </table>

```

In simple table markup, cell widths are the same as the corresponding column widths, so COLWIDTH really specifies the cell widths. Omitting the COLWIDTH from all the COLSPECS causes each column to have the same width. In our example, the Xyvision formatter makes the whole table as wide as the text column (this is the initial setting) and calculates the table columns based on that.

Of course, you won't always want all your table columns to be the same width. The COLWIDTH attribute and the asterisks tell the formatter that we want the table divided proportionally; we don't care what the exact width turns out to be. So the formatter decides for us. You can specify 1*, 1*, and 2* if you want the first and second columns to have the same width, and to have last column be twice as wide as the first or second column.

Table captions and descriptions

Your tables can have a short caption and an optional longer description. The caption can appear in a table list at the beginning of your book. A good place for the table list is after your table of contents. If your table has a caption, you should also give it an ID; some processes (like BookMaster) complain about tables that have captions but are missing the ID. Put the ID on the table tag, not on the caption. The caption should be entered like a heading, without ending punctuation.

Here's a sample table with a small caption:

Table 8. Sample table caption

my little
sample table

Here's its markup:

```

+      <table pgwide="0" id="tablesample">
+      <cap>Sample table caption</cap>
+      <tgroup cols="1">
+      <colspec colname="col1">
+      <tbody>
+      <row>
+      <entry colname="col1">my little</entry>
+      </row>
+      <row>
+      <entry colname="col1">sample table</entry>
+      </row>
+      </tbody>
+      </tgroup>
+      </table>

```

Here's another sample table with both a caption and a description. You enter a description like a sentence, with punctuation. Note that the caption source has no punctuation; if any is needed, it is added by the formatter.

Table 9. Sample table caption. This table shows little remarkable information. You need to read between the lines.

my little
sample table

Here's its markup:

```
<table pgwide="0" id="tablesampldesc">
<cap>Sample table caption</cap>
<desc>This table shows little remarkable information.
You need to read between the lines.</desc>
<tgroup cols="1">
<colspec colname="col1">
<tbody>
<row>
<entry colname="col1">my little</entry>
</row>
<row>
<entry colname="col1">sample table</entry>
</row>
</tbody>
</tgroup>
</table>
```

Page, column, and line-wide tables

You use the PGWIDE attribute on the TABLE tag to control the width of a table.

PGWIDE=0

(zero) makes the table column-wide

pgwide=0	

PGWIDE=1

(one) makes the table page-wide.

pgwide=1	

PGWIDE=2

(two) makes the table as wide as the current text line.

pgwide=2	

In BookMaster, tables default to page wide. In Xyvision, tables default to column wide.

If you need to have wide tables in BookManager BOOKs, use the DWIDTH attribute to specify a wide display width. The normal setting is 75:
style=bkm:(dwidth=100)

Complex tables are NOT supported in RTF nor IPF. Complex tables are tables whose contents include lists, definition lists, etc. or tables with complex

column/row spanning. If you are creating RTF or IPF output, you should avoid making use of the complex features that the Adept table editor enables for you.

Splitting tables between pages

In BookMaster hardcopy, tables do not split. This may cause a BookMaster error and the second part of the table will not have a caption. Use this override on your large tables so they split in BookMaster without errors:

bkm:(split=yes)

In Xyvision, tables always split. You should accept this default due to possible translation center impacts (expansion space in rows). If you can't bear to split your table, you can use this override:

bkm:(split=no)

Affecting how a table appears

The Table tag allows several attributes and settings to control how your table appears. These are all available in the Xyvision PostScript formatter. The other transforms support some of these settings, but not all of them. Experiment with your desired output formats to determine what you can use.

- To make the table have a frame, rules, or nothing, the FRAME attribute can be set to:

all Rules appear on all four sides; the table is boxed.

frame=all	

bottom

frame=bottom	

none

frame=none	

sides

frame=sides	

top

frame=bottom	

topbot

frame=bottom	

- To control the vertical rules in the table, use the COLSEP attribute.

0 (zero) makes the table have no column separators

colsep=0	
No vertical rules appear	

1 (one) makes the table have column separators

colsep=1	
	The vertical rules appear

- To control the horizontal rules in the table, use the ROWSEP attribute.

0 (zero) makes the table have no row separators

rowsep=0	
No horizontal rules appear	

1 (one) makes the table have row separators

rowsep=1	
	The horizontal rules appear

- To rotate a table, set the ORIENT attribute to LAND.

port Use this for normal, portrait-oriented tables.

orient=port	

land Use this to turn your table on it's side (landscape) for hardcopy.
PGWIDE is ignored (it is assumed to be full-page).

orient=land	

Defining the Column Specifications

A typical set of ColSpec attributes for a simple table is shown in the example that follows.

Migration Note

At this time, the graphic table editor does not handle a mixture of proportional and fixed COLWIDTH values in the same table. You should use proportional COLWIDTHs until this issue is resolved.

```
<TABLE FRAME="ALL">
<TGROUP COLS="4" COLSEP="1" ROWSEP="1" ORIENT="PORT" PGWIDE="0">
<COLSPEC COLWIDTH="68*">
<COLSPEC COLWIDTH="127*">
<COLSPEC COLWIDTH="195*">
<COLSPEC COLWIDTH="66*">
:
:
```

The ColSpec attributes include:

COLNUM=*col_number*

This value indicates the number of the column.

COLNAME=*col_name*

Specifies the column name. This name is referenced by other table elements.

ALIGN=**LEFT** | **RIGHT** | **CENTER** | **JUSTIFY** | **CHAR**

This attribute specifies the horizontal positioning of text in the column.

LEFT

specifies left alignment (the default).

RIGHT

specifies right alignment.

CENTER

specifies center alignment.

JUSTIFY

specifies that the text should be justified.

CHAR

specifies the character that is used for alignment.

CHAROFF=*number*

Specifies the character offset for Entry elements in a column.

COLWIDTH=*measure*

Specifies a fixed, proportional, or mixed measure for the column width.

Migration Note

At this time, mixed measures are not supported. You should use proportional measures.

COLSEP=**0** | **1**

This attribute's value specifies that the internal column rules should be:

- to the right of each cell's content (1)
- not displayed at all (0)

Note that BookMaster does not allow rules on only one side of a column.

ROWSEP 0 | 1

This attribute's value specifies that the internal row rules should be:

- below each Entry element that ends a row (1)
- not displayed at all (0)

Note that BookMaster does not allow rules on only one side of a row.

Defining Rows and Entrys

Rows are defined using the Row element. Each Entry element contained in a Row element occupies the consecutive column, from left to right. The two Row attributes you will use are VALIGN (vertical alignment) ALIGN (horizontal alignment). Unless the ROWSEP attribute is specified, the Row inherits the ROWSEP value specified on the Table or TGroup element.

```
<TABLE FRAME="ALL">
<TGROUP COLS="3" COLSEP="1" ROWSEP="1">
  <COLSPEC COLWIDTH="152*">
  <COLSPEC COLWIDTH="152*">
  <COLSPEC COLWIDTH="152*">
  <TBODY>
    <ROW>
      <ENTRY VALIGN="TOP" ALIGN="LEFT">ROW 1, CELL 1</ENTRY>
      <ENTRY VALIGN="TOP" ALIGN="LEFT">ROW 1, CELL 2</ENTRY>
      <ENTRY VALIGN="TOP" ALIGN="LEFT">ROW 1, CELL 3;
      HERE'S A LITTLE MORE TEXT THAN THE OTHER CELLS HAVE.</ENTRY>
    </ROW>
    :
  
```

To have unformatted text in a table, insert the LINES tag, then insert your cell content within the lines.

A Few Simple Table Examples

Let's look at a couple of simple tables. This section includes the IBMIDDoc markup, and an approximation of the resulting formatted output.

A Simple Table

Let's look at a simple IBMIDDoc table

Table 10. A simple example

Row 1, Cell 1	Row 1, Cell 2	Row 1, Cell 3; here's a little more text than the other cells have
Row 2, Cell 1	Row 2, Cell 2	Row 2, Cell 3

Here's its markup:

```
<table frame="all" pgwide="0">
<cap>A simple example</cap>
<tgroup cols="3" colsep="1" rowsep="1">
  <colspec colname="col1" colwidth="1*">
  <colspec colname="col2" colwidth="1*">
  <colspec colname="col3" colwidth="1*">
  <tbody>
    <row>
      <entry valign="top">Row 1, Cell 1</entry>
      <entry valign="top">Row 1, Cell 2</entry>
```

```

+      <entry valign="top">Row 1, Cell 3; here's a little
+      more text than the other cells have</entry>
+    </row>
+  </row>
+  <entry valign="top">Row 2, Cell 1</entry>
+  <entry valign="top">Row 2, Cell 2</entry>
+  <entry valign="top">Row 2, Cell 3</entry>
+ </row>
+ </tbody>
+ </tgroup>
+ </table>

```

A Simple Table with More Options

Now let's take a similar table and add another column.

Table 11. Another simple table

Row 1, Cell 1	Row 1, Cell 2	Row 1, Cell 3; here's a little more text than the other cells have	Row 1, Cell 4
Row 2, Cell 1	Row 2, Cell 2	Row 2, Cell 3	Row 2, Cell 4

Here's its markup:

```

+ <table frame="all" pgwide="0">
+   <cap>Another simple table</cap>
+   <tgroup cols="4" colsep="1" rowsep="1" style="BKM:(cols='1* 2* 3* 1*')">
+     <colspec colname="col1" colwidth="1*">
+     <colspec colname="col2" colwidth="2*">
+     <colspec colname="col3" colwidth="3*">
+     <colspec colname="col4" colwidth="1*">
+   <tbody>
+     <row>
+       <entry valign="top">Row 1, Cell 1</entry>
+       <entry valign="top">Row 1, Cell 2</entry>
+       <entry valign="top">Row 1, Cell 3; here's a little
+       more text than the other cells have</entry>
+       <entry valign="top">Row 1, Cell 4</entry>
+     </row>
+     <row>
+       <entry valign="top">Row 2, Cell 1</entry>
+       <entry valign="top">Row 2, Cell 2</entry>
+       <entry valign="top">Row 2, Cell 3</entry>
+       <entry valign="top">Row 2, Cell 4</entry>
+     </row>
+   </tbody>
+ </tgroup>
+ </table>

```

A Simple Table with a Table Header and IBMIDDoc Elements

Now let's take a similar table and add a THead element and some IBMIDDoc Phrase elements with STYLE attribute specifications.

Table 12. Another sample table

Col #1	Col #2	Col #3	Col #4
Row 1, Cell 1	1. Row 1 2. Cell 2	Row 1, Cell 3; here's a little more text than the other cells have	Row 1, Cell 4
Row 2, Cell 1	Row 2, Cell 2	Row 2, Cell 3	Row 2, Cell 4

```

+
+ Here's its markup:
+
+ <table frame="all" pgwide="0">
+ <cap>Another sample table</cap>
+ <tgroup cols="4" colsep="1" rowsep="1">
+ <colspec colname="col1" colwidth="1*">
+ <colspec colname="col2" colwidth="2*">
+ <colspec colname="col3" colwidth="3*">
+ <colspec colname="col4" colwidth="1*">
+ <thead>
+ <row>
+ <entry valign="top" rowsep="1">Col #1</entry>
+ <entry valign="top" rowsep="1">Col #2</entry>
+ <entry valign="top" rowsep="1">Col #3</entry>
+ <entry valign="top" rowsep="1">Col #4</entry>
+ </row>
+ </thead>
+ <tbody>
+ <row>
+ <entry valign="top">Row 1, Cell 1</entry>
+ <entry valign="top"><ol>
+ <li>Row 1</li>
+ <li>Cell 2</li>
+ </ol></entry>
+ <entry valign="top">Row 1, Cell 3; here's a little
+ more text than the other cells have</entry>
+ <entry valign="top">Row 1, Cell 4</entry>
+ </row>
+ <row>
+ <entry valign="top">Row 2, Cell 1</entry>
+ <entry valign="top">Row 2, Cell 2</entry>
+ <entry valign="top"><ph style="italic">Row 2, Cell
+ 3</ph></entry>
+ <entry valign="top">Row 2, Cell 4</entry>
+ </row>
+ </tbody>
+ </tgroup>
+ </table>

```

A Complex Table with Row and Column Spans

Now let's take a similar table and add NAMEST, NAMEEND, and MOREROWS attribute values to change the look of the table. NAMEST and NAMEEND specify the spanning of columns, and MOREROWS specifies the spanning of rows.

Table 13. Complex table example

Row 1, Cell 1		Row 1, Cell 2
Row 1, Cell 3	Row 1, Cell 4	

```

+
+ Here's its markup:
+
+ <table frame="all" pgwide="0" id="complt"><cap>Complex table example</cap>
+ <tgroup cols="3">
+ <colspec colname="col1" colwidth="77*">
+ <colspec colname="col2" colwidth="100*">
+ <colspec colname="col3" colwidth="119*">
+ <tbody>
+ <row>
+ <entry namest="col1" nameend="col2">Row 1, Cell 1
+ </entry>
+ <entry colname="col3" morerows="1">Row 1, Cell 2</entry>
+ </row>
+ <row>
+ <entry colname="col1">Row 1, Cell 3</entry>
+ <entry colname="col2">Row 1, Cell 4</entry>

```

```
+ </row>
+ </tbody>
+ </tgroup>
+ </table>
```

+ **A Complex Table Header**

+ Here's a complex table header. As in the previous example, NAMEST, NAMEEND, and MOREROWS were used to combine the heading cells.

±

Head 1	Head 2		
	Sub 1	Sub 2	Sub 3
a	b	c	d

+ Here's it's markup:

```
+ <table frame="all" pgwide="0">
+ <tgroup cols="4">
+   <colspec colname="col1">
+   <colspec colname="col2">
+   <colspec colname="col3">
+   <colspec colname="col4">
+   <thead>
+     <row>
+       <entry colname="col1" morerows="1" align="center">Head 1</entry>
+       <entry namest="col2" nameend="col4" align="center">Head 2</entry>
+     </row>
+     <row>
+       <entry colname="col2" align="center">Sub 1</entry>
+       <entry colname="col3" align="center">Sub 2</entry>
+       <entry colname="col4" align="center">Sub 3</entry>
+     </row>
+   </thead>
+   <tbody>
+     <row>
+       <entry colname="col1">a</entry>
+       <entry colname="col2">b</entry>
+       <entry colname="col3">c</entry>
+       <entry colname="col4">d</entry>
+     </row>
+   </tbody>
+ </tgroup>
+ </table>
```

| **Adding footnotes to a table**

| While you cannot have footnotes within a table using the FN tag, you can use superscripts and a note list to have the same affect. For example, here's a table with some sample notes:

|

Sample ¹	And another ²
Notes: 1. The first table note 2. And another table note.	

| Here is its coding:

```
| <table pgwide="0">
| <tgroup cols="2">
|   <colspec colname="col1">
|   <colspec colname="col2">
```

```

|      <tbody>
|      <row>
|      <entry colname="col1">Sample<ph style="superscript">1</ph></entry>
|      <entry colname="col2">And another<ph style="superscript">2</ph></entry>
|      </row>
|      <row>
|      <entry namest="col1" nameend="col2">
|      <notelist>
|      <li>The first table note</li>
|      <li>And another table note.</li>
|      </notelist></entry>
|      </row>
|      </tbody>
|      </tgroup>
|      </table>

```

Chapter 8. The document structure of an IBMIDDoc document

This section illustrates the order and usage of elements in creating a document. The high-level elements include:

- The IBMIDDoc tag itself (“About the IBMIDDoc tag”)
- Prolog (“About the prolog” on page 75)
- FrontM, front matter⁵ (“Front matter (FrontM)” on page 85)
- Body (already covered previously, see “Creating the body of your document” on page 17)
- BackM, back matter (“About back matter (BackM)” on page 88)

This example gives an high-level view of the structure of an IBMIDDoc document:

```
<ibmidoc>
<prolog>
...
</prolog>
<frontm>
...
</frontm>
<body>
<d>
...
</d>
</body>
<backm>
...
</backm>
</ibmidoc>
```

About the IBMIDDoc tag

The IBMIDDoc tag contains information about the whole document. This includes the document style, language used, security classification, and page-numbering information.

(sec ibmsec ibmcopyr language resmat and all the stuff currently at the ARB level)

Class Classif Company Copyr Language MLSprefix Sec

Getting in style, the document style, that is

IBMIDDoc has a number of built-in styles. To use a style other than the default, set the IBMIDDoc tag’s DocStyle attribute to one of these following values:

IBM8X11

8-1/2 by 11 inch style. Replaces BookMaster style IBMXAGD.

IBM7X9

7 by 9 inch style. Replaces BookMaster style IBMXGGD.

IBM2COL

8.5x11 style (2 column layout)

5. Some folks call this “don’t matter”, because customers seldom read it – do you read a preface?

IBMCD

4.75x4.75 style (for CD Jewel Case booklets)

IBMREFC

Reference cards (3-5/8x9in.).

IBM5X8

5.5x8.5 style (for hardware).

IBM4X6

4.25x6.25 style (for hardware).

IBM8X5

5.5x8.5 landscape style (for hardware).

IBM9X7

7x9 landscape style.

If you create a PDF from this style, the pages may switch between landscape and portrait presentation in Adobe Acrobat Reader or Exchange. Add the following lines to your PostScript file before distilling it to prevent this from occurring:

```
/currentdistillerparams where {pop}
{userdict /currentdistillerparams {1 dict} put} ifelse
/setdistillerparams where {pop}
{userdict /setdistillerparams {pop} put} ifelse
<< /AutoRotatePages /All >> setdistillerparams
```

IBMLAND

Printer System's landscape books. (Not for BookMaster)

IBMXAGD

User Guides (8.5x11in., A4); old BookMaster style.

IBMXARF

Reference (8.5x11in., A4); old BookMaster style.

IBMXGGD

Summary Guides (7-3/8x9in.); old BookMaster style.

TIV7X9

7x9 style for Tivoli

TIV8X11

8.5x11 style for Tivoli

OBIPORT

5.5x8.5 style (for Options by IBM)

OBIWWA6P

4.25x5.75 style (for Options by IBM)

SMALLFLG

3.625x8.5 style (for hardware)

Setting the IBM copyright

The dates for the IBM copyright are taken from the IBMCopyr attribute. You typically enter either a single year or two years separated by a comma. Here's the setting for a brand new book, published for the first time in 1999:

```
<ibmiddoc ibmcopyr="1999">
```

Here's the setting for a book that was originally published in 1985, and was last published in 2000:

| <ibmiddoc ibmcopyr="1985, 2000">

| The date is printed in the edition notice; see “Notices and Edition notices” on
| page 86.

| **Setting the security classification**

| Sometimes your document needs to be confidential. To make it so, set the
| IBMIDDoc tag’s IBMSec attribute to IC.

| **Setting page numbering to sequential or folio-by-chapter**

| Page numbering by chapter (known as folio-by-chapter) is a technique of
| numbering where the chapter number (or in the case of appendixes, the appendix
| letter) is prefixed to the page numbers, the figure numbers, and the table numbers.
| Thus, the fifth page in chapter 3 is numbered 3-5, the second figure in chapter 7 is
| numbered 7-2, and the fourth table in appendix E is numbered E-4. The page
| numbers, figure numbers, and table numbers are reset to 1 at the beginning of each
| chapter.

| Adding the PageNumber=FBC attribute to the IBMIDDoc tag gives you
| folio-by-chapter page numbering.

| If your book has more than one volume, see “Creating multiple volumes for a
| book”.

| **Creating multiple volumes for a book**

| Sometimes you have a book that is just huge. To have the book printed, you need
| to divide it into volumes. To do this, you need to add the MULTIVOL=Index-Folio
| attribute to the IBMIDDoc tag. You can use either sequential or folio-by-chapter
| page numbering; see “Setting page numbering to sequential or folio-by-chapter”.

| The Index-Folio setting adds X- as a prefix for the page numbers in the index and
| starts the page numbering from 1.

| The remaining steps to create a set of multiple volume books are as follows:

- | 1. You format your book as one large book, to get the table of contents,
| cross-reference, and index page numbers correct.
- | 2. If your multi-volume books also have different cover pages and possibly order
| numbers, you will need to create mini-documents with that cover information.
- | 3. You then use Adobe Acrobat to separate your large document into pieces;
| adding the separate pieces to the mini-documents.

| A typical set of volumes would contain:

- | 1. Volume 1
 - | a. cover
 - | b. edition notice
 - | c. table of contents, figure list, table list (for entire set)
 - | d. chapter 1 through 10, for example
 - | e. index (for entire set)
- | 2. Volume 2
 - | a. cover
 - | b. edition notice

- c. table of contents, figure list, table list (for entire set)
- d. chapter 11 through 20, for example
- e. index (for entire set)

Controlling generated chapter, part, and appendix titles

Sometimes you may want to change the way “Chapter” or “Appendix” is added to your headings. The IBMIDDoc tag attributes PartPrefix ChapPrefix AppPrefix have attributes that allow you to control the form of this generated text. The have the basic form show below:

text

This is the default. This outputs: “Part 1.” for part headings, “Chapter 1.” for chapter headings, or “Appendix A.” for appendix headings.

numonly

This omits the word from the number. This outputs: “1.” for part headings, “1.” for chapter headings, or “A.” for appendix headings.

text

This omits the word and the number from the heading. Only the heading text itself appears.

Specifying the language of the document

The Language attribute specifies the language in which a document is written. It also specifies how the IBMIDDoc-generated text should be printed.

The valid values for the Language attribute on IBMIDDoc element are:

- ENGLISH, en_US, or USENGLISH
- UKENGLISH or en_GB
- DUTCH or nl_NL
- GERMAN or de_DE
- ITALIAN or it_IT
- FRENCH or fr_FR
- SPANISH or es_ES
- PORTUGUESE or pt_PT
- DANISH or da_DK
- FINNISH or fi_FI
- NORWEGIAN or no_NO
- SWEDISH or sv_SE
- CFRENCH or fr_CA
- BFRENCH or fr_BE
- BDUTCH or nl_BE
- BPORTUGUESE or pt_BR
- CENGLISH or en_CA
- ICELANDIC or is_IS
- SGERMAN or de_CH
- SFRENCH or fr_CH
- SITALIAN or it_CH
- KOREAN or ko_KR
- TCHINESE or zh_TW

- SCHINESE or zh_CN
- JAPANESE or ja_JP
- CATALAN or ca_ES
- TURKISH or tr_TR
- GREEK or el_GR
- POLISH or pl_PL
- CZECH or cs_CZ
- SLOVAK or sk_SK
- HUNGARIAN or hu_HU
- CROATIAN or hr_HR
- SLOVENIAN or sl_SI
- RUSSIAN or ru_RU
- ROMANIAN or ro_RO
- BULGARIAN or bg_BG
- ESTONIAN or et_EE
- LATVIAN or lv_LV
- LITHUANIAN or lt_LT
- MACEDONIAN or mk_MK
- SERBIAN or sr_SP

Bookmarks for PDF tables of contents

Adobe Acrobat PDF documents can have bookmarks generated that match the document's table of contents. This creates a very usable method of navigating the PDF file. You specify the creation of these bookmarks for Xyvision-formatted PostScript and PDF files this way:

```
<ibmddoc style="xpp:(bookmarks)">
```

This is now the default setting; you no longer need to code this entry.

Line justification for DBCS languages

This is used only for DBCS (double-byte character set) languages. While this is not supported at this time, you can have it in the source. This is used for left and right justification of text; the preferred format for DBCS languages. You specify the justification control this way:

```
<ibmddoc style="xpp:(justify)">
```

The default is to not justify; the values `nojustify` and `ragged` indicate that.

About the prolog

The prolog is where we put information (marked up with special tags) about the whole document. For instance, the title, the author's name and address, and so forth.

Here is a simple prolog:

```
<prolog>
<ibmbibentry><doctitle><titleblk>
<title>My Cute, Little Document</title>
</titleblk></doctitle>
<ibmdocnum>SC99-1234-01</ibmdocnum>
<authors>
```

```

<author><person>
<name>Fred Mertz</name>
<address>East Overshoe, SD</address>
</person></author>
</authors>
</ibmbibentry>
</prolog>

```

Not all the tags you can use in the prolog are shown here. We'll discuss the ones shown first and then tell you about the others. The prolog itself does not cause anything to be printed; instead, it is a place to collect information that will be used in other places — for example, on the draft title page when the TIPAGE value is used.

Document title

The document's title is contained in the IBMBibEntry and Title elements. If the document title is short, you enter it with the Title element alone. If you want multiple lines in your title, you use the Title element and insert your lines as you want them formatted; putting a carriage return after each line to be split.

So a short title would be entered like this:

```
<title>Tom Sawyer</title>
```

whereas a long title would be entered like this:

```
<title>The Do's and Don'ts of
Caring for Your Fruit Bat</title>
```

If your title is very long, you may also want to use the STITLE element to get a short title (used in some document styles for the even-page running foot). For example:

```
<title>The Do's and Don'ts of
Caring for Your Fruit Bat</title>
<stitle>Fruit Bat User's Guide</stitle>
```

IBMBibEntry contains all of the bibliographic information for the document. It can contain the following:

- Authors
- DocTitle
- FileNum
- IBMDocNum
- IBMPartNum
- ISBN
- PublicID
- Publisher
- RetKey
- CoverDefs ("Adding to the front or back cover (CoverDef)" on page 78)

Document number

You enter the document number, if you have one, with the IBMDocNum element. For example:

```
<ibmdocnum>SC99-1234-01</ibmdocnum>
```

The last two number are called the "dash-level"; this is always a two-digit number.

Author and Address

The Author and Address elements contain the author's name and address information. The address is entered using as many lines as needed, surrounded by Address and its end tag. The text for each address line must be on a single line. For example:

```
<authors>
<author><person>
<name>Fred Mertz</name>
<address>
127 East Main Street,
East Overshoe, SD <postalcode>59134</postalcode>
</address>
</person></author>
</authors>
```

Date

The CritDates element contains the date of your document. For example, this specifies a date of September 9th, 1999:

```
<critdate>
<date>September 9th, 1999</date>
<desc>Date of publishing.</desc>
</critdate>
```

Improving the searching of PDF books

Several prolog items will help in the focused searching of Adobe Acrobat PDF books. Ensure you have these items; they are passed through to the PDF document's "Information" dialog:

IBMIDDoc Element	PDF Document Information field
Library title (or stitle): document title (or stitle)	Title
Desc in IBMBibEntry	Subject
IBM, Tivoli, or blank (from company attribute on IBMIDDoc)	Author
Retkey in IBMBibEntry	Keywords
"XPP"	Creator
(blank -- automatic by Distiller)	Producer
(current date)	Created
(blank -- automatic by Distiller)	Modified

Other prolog elements

The Prolog contains all tracking and control information for the document. Prolog can contain a number of elements, including:

- Approvers (similar to Authors)
- BibEntryDefs ("Chapter 13. Bibliographies and citations" on page 119)
- CopyrDefs ("Using CopyRDefs" on page 78)
- GLDefs ("Using GLDefs" on page 83)
- IBMProdInfo, IBM product information ("Using IBMProdInfo" on page 79)
- IDXDefs ("Chapter 10. Indexing" on page 97)
- LDescs ("Chapter 11. All about linking" on page 109)

- Maintainer (“Using reader’s comment form (RCF)” on page 90)
- MasterIndexInfo (“Creating a master index” on page 105)
- ObjLib (“Reusing elements from an object library” on page 166)
- Owners (similar to Authors)
- ProdInfo
- PropDefs
- QualifDefs, qualification definitions (“Qualifying information” on page 41)
- RevDefs defines the revisions and marks that can be used in the document (“Defining Revisions in the RevDefs Element” on page 91).

Adding to the front or back cover (CoverDef)

You use the CoverDefs element to define cover artwork for your book’s front and back covers. It is contained in the IBMBIBENTRY; the title for your book. For example:

```
<!entity front1 system "front1.eps" ndata graphics>
<!entity back1 system "back1.eps" ndata graphics>
...
<prolog><ibmbibentry><doctitle><titleblk>
<title>Sample Cover</title>
</titleblk></doctitle>
<coverdef>
<frontcover><mmobj><objref obj="front1">
<textalt>System/X cover artwork</textalt>
</mmobj></frontcover>
<backcover><mmobj><objref obj="back1">
<textalt>System/X back cover artwork</textalt>
</mmobj></backcover>
</coverdef>
</ibmbibentry></prolog>
```

You can also add text to the front cover. Inside the FrontCover tag, insert the PBLK tag and any content that you want to appear on the cover. For example:

```
<frontcover>
<pblk style="l1lbox"><title>Notice</title>
<p>
The IBM License Agreement for Machine Code is included in this book.
Carefully read the agreement. By using this product you agree to
abide by the terms of this agreement and applicable copyright laws.
</p>
</pblk>
</frontcover>
```

Using CopyRDefs

If you have copyrighted material from some other company, you need to enter that company’s information in the document prolog.

The CopyRDefs element contains the primary copyright information for the document. Use the CopyR element to specify this primary copyright information.

The primary CopyR element must have an ID attribute. This ID is referred to on the IBMIDDOC element using the Copyr attribute, as shown in the example that follows.

```
<IBMIDDOC COPYR="ibmprim">
<PROLOG>
:
<COPYRDEFS><COPYR ID="ibmprim">
```

```

        <P>IBM Corporation
          1994, 1995
          All Rights Reserved</P></COPYR>
    </COPYRDEFS>

    :
    </PROLOG>

    :
    </IBMIDDOC>

```

Using IBMProdInfo

The IBMProdInfo element contains the IBM-specific product information about the product described in the document.

- ProdName, product name
- Version
- Release
- ModLvl, modification level
- IBMPgmNum, IBM program number
- IBMFeatNum, IBM feature number

For example:

```

    <ibmprodinfo>
      <prodname>System/36</prodname>
      <version>2</version>
      <release>3</release>
      <modlvl>1</modlvl>
      <ibmprognum>223-3330</ibmprognum>
    </ibmprodinfo>

```

Using Property Definitions (PropDefs)

PropDefs contains elements that define properties that can be used by other elements in your document. These properties apply to elements contained within the document or division with which the property definitions are associated.

All property definition elements can be used in PropDefs. These elements include:

- ClassDef (“Defining Element Classes” on page 179)
- PropDef (“Defining Element Properties” on page 177)
- LersDef (“Chapter 16. Defining Modular Information” on page 151)
- ModInfoDef (“Chapter 16. Defining Modular Information” on page 151)
- MsgItemDef (“Message and code lists” on page 29)
- PropGroup

PropDefs and Common Property Values

Use the PropDef element to define common property values. In the absence of ELETYPES or ID attributes, the property specified applies to all elements. These common properties are specified on common attributes. Examples of such common attributes are:

- Props
- Status
- Style

Limiting the Scope of PropDef Definitions

The global effect of PropDef definitions can be limited by specifying PROPNAME and ELETYPES. Scoping can also be limited by using a different set of PropDefs in each DProlog, instead of having only one set in the Prolog of the document.

For example, specifying ELETYPES='UL' on a PropDef element causes the other properties specified on the same PropDef element to apply to all UL elements.

The PROPNAME attribute with a value of P001 provides a name to refer to when you want to apply the particular property definition to a specific element in your document using the PROPSRC attribute.

If both ELETYPES and PROPNAME are specified, the properties specified on the PropDef element apply to all of the specified element types, and may be referred to by the PROPNAME value.

If you wish that all figures be boxed figures, you can use the property definition described in the examples that follow.

```
⋮
<PROLOG>
⋮
<PROPDEFS>
  <PROPDEF PROPNAME="wider" ELETYPES="fig"
    STYLE="BKM:(width=page place=inline frame=rules)">
  </PROPDEF>
</PROPDEFS>
```

All Figure elements will now be framed.

In the next example, the PropDef has an ID of P001, and can be referenced by any element where such properties are valid.

```
⋮
<PROLOG>
⋮
<PROPDEFS>
  <PROPDEF PROPNAME="P001" ELETYPES="fig"
    STYLE="BKM:(width=page place=inline frame=rules)">
  </PROPDEF>
</PROPDEFS>
```

The document markup for a figure that includes PowerPC artwork would look like the example that follows.

```
<FIG ID="Unit" PROPSRC="P001">
  <FIGCAP>The IBM PowerPC CPU</FIGCAP>
  <MMOBJ>
    <OBJREF OBJ="ppcfig">
    ⋮
  </MMOBJ>
</FIG>
```

For hardcopy documents, you may have some column-wide figures. Instead of specifying the override on each column-wide figure, define your figure PROPDEF tags like this. The first PROPDEF (without the propname) sets the default for all figures; the second PROPDEF sets the column-wide override.

```
<propdef eletypes="fig" style="bkm:(place=inline)">
</propdef>
<propdef propname="colfig" eletypes="fig" style="bkm:(width=column place=inline)">
</propdef>
```

To get a column-wide figure, you specify "COLFIG" on the PROPSRC attribute of that figure. For example:

```
<fig propsrc="colfig">
```

By default, the BookMaster output process places these at the top of the next page. All other output processes place them inline. Use the INLINE override. For example:

```
style="bkm:(width=column place=inline)"
```

For wide figures in BookManager BOOKs, you may need to use the DWIDTH attribute. For example:

```
style="bkm:(width=column place=inline dwidth=100)"
```

For more information about using PropDefs, see "Chapter 19. Property and Class Definition" on page 177.

Using PropDefs for Conditional Processing

Properties may be used to include or exclude information. This is called property-based retrieval.

For example, you can define a complex set of properties for doing conditional processing using PropDef, and then use those values for other elements by referring to the PropDef element, rather than having to explicitly specify those attributes on each element.

In the example that follows, a PropDef element is used to define the properties for including either RS6000 or PowerPC information.

```
<!DOCTYPE IBMIDDOC PUBLIC "-//ISBN 0-933186::IBM//DTD IBMIDDoc//EN"
"ibmiddoc.dtd"
<IBMIDDOC COPYR="ibmprim"><PROLOG>
</PROLOG>
<PROPDEFS>
  <PROPDEF PROPNAME="ppc" PROPS="POWERPC #AND #NOT RS6000"
    <DESC>This PropDef will be referred to when PowerPC information is
      to be processed.</DESC>
  </PROPDEF>
  <PROPDEF PROPNAME="rs6" PROPS="RS6000 #AND #NOT POWERPC"
    <DESC>This PropDef will be referred to when RS6000 information is
      to be processed.</DESC>
  </PROPDEF>
</PROPDEFS>
</PROLOG>
<BODY>
<D PROPSRC="PPC">
  <DPROLOG>
  <TITLEBLK>
    <TITLE>Installing a 128-Bit Video Card in the IBM PowerPC</TITLE>
  </TITLEBLK>
  </DPROLOG>
  <DBODY><P>This procedure describes how to insert the 128-Bit
    video card in the IBM PowerPC.</P>
  <PROC>
    <TITLEBLK><TITLE>Installing a 128-Bit Video Card</TITLE></TITLEBLK>
    <PROCENTRY>
      :
    
```

```

        </PROC>
      </DBODY>
    </D>

    :
  <D PROPSRC="RS6">
    <DPROLOG>
      <TITLEBLK>
        <TITLE>Installing a 128-Bit Video Card in the RS6000</TITLE>
      </TITLEBLK>
    </DPROLOG>
    <DBODY><P>This procedure describes how to insert the 128-Bit
      video card in the RS6000.</P>
    <PROC>
      <TITLEBLK><TITLE>Installing a 128-Bit Video Card</TITLE></TITLEBLK>
      <PROCENTRY>

      :
    </PROC>
    </DBODY>
  </D>

  :
</BODY>

  :
</IBMIDDOC>

```

Using LDescs and Nameloc

LDescs contains elements to describe links and the locations used in these links.

Contained location elements can:

- provide an indirect reference to another SGML element. This protects the link(s) from changes made to the target element.
- associate a single ID with multiple elements
- support references to other documents, and to elements in other documents
- support references to non-SGML information.

For example, to link to multiple elements in the same document, you can use the Nameloc element to contain a list of IDs that are used to identify the information topic. In the example that follows, the LINKEND attribute refers to the Nameloc element with the ID=DDInfo. The elements in the document that have the ID=aboutDan and ID=REHero are linked to any link element which references the ID DDInfo using the LINKEND attribute.

```

<IBMIDDOC>
  <PROLOG>
    <IBMBIBENTRY>
      <DOCTITLE>
        <TITLEBLK><TITLE>Our Saturday Heroes</TITLE>
      </TITLEBLK>
      </DOCTITLE>
    </IBMBIBENTRY>
  <LDESCS>
    <NAMELOC ID="DDInfo">
      <NMLIST>AboutDan REHero</NMLIST>
    </NAMELOC>
  </PROLOG>
  <BODY>
    <D>
      <DPROLOG>
        <TITLEBLK><TITLE>Movie Serials</TITLE>

```

```

</TITLEBLK>
</DPROLOG>
<DBODY>
  <P>The <L LINKEND="ddinfo">Dan Danger serial hero</L>
    was very popular in the 1940s.</P>
  <D>
    <DPROLOG>
      <TITLEBLK><TITLE>Male Heros</TITLE>
      </TITLEBLK>
    </DPROLOG>
    <DBODY>
      <P ID="aboutdan">Looking back, we now see Dan Danger as the
        quintessential Saturday morning serial hero.</P>
      <D>
        <DPROLOG ID="REHero">
          <TITLEBLK><TITLE>Heroes and Villains</TITLE>
          </TITLEBLK>
        </DPROLOG>
        <DBODY><P>More stuff about heroes....</P>
        </DBODY>
      </D>
    </DBODY>
  </D>
</DBODY>
</BODY>
</IBMIDDOC>

```

Using GLDefs

Use GLDefs to contain GLEntry elements that can be used by reference from anywhere in your document. For more information about using GLEntry, see “Chapter 12. Glossaries” on page 115.

In the next example, terms are defined in GLDefs in the Prolog, and referred to by CONLOC reference within the document.

```

<PROLOG>
  ⋮
  <GLDEFS>
    <GLENTRY>
      <TERM ID="mainec">Maine Coon</TERM>
      <DEFN>A friendly and gentle breed of cat.</DEFN>
    </GLENTRY>
    <GLENTRY>
      <TERM ID="ragdoll">Rag Doll</TERM>
      <DEFN>A gentle breed of cat that may be even
        more docile than the Maine Coon.
      </DEFN>
    </GLENTRY>
  </GLDEFS>
</PROLOG>
<BODY>
  <D>
    <DPROLOG>
      <TITLEBLK>
        <TITLE>Movie Serials
        </TITLE>
      </TITLEBLK>
    </DPROLOG>
    <DBODY>
      <P>The <L LINKEND="ddinfo">Dan Danger serial hero</L>
        was very popular in the 1940s. Dan's sidekick was a
        <TERM CONLOC="mainec"> named Elvis.</P>
    </DBODY>
  </D>
</BODY>

```

```

:
</IBMIDDOC>

```

Using BibEntryDefs

BibEntryDefs contains BibEntry, LibEntry, IBMBibEntry, and IBMLibEntry elements.

An IBMIDDoc document requires a IBMBibEntry element, which contains the bibliographic information about the document.

BibEntryDefs is an optional Prolog element that can contain a variety of bibliographic entries that can be referred to throughout the document, or to build a bibliography by reference.

In the example that follows, BibEntryDefs contains several IBMBibEntry elements. The first IBMBibEntry is used to contain information about the containing document. The other IBMBibEntry elements contain entries for other referenced documents, and an IBMBibEntryDef that uses the CONLOC attribute to get its content from the containing document's IBMBibEntry. The example also includes an IBMLibEntry element.

```

<PROLOG>
<IBMBIBENTRY ID="BOOK0">
  <DOCTITLE>
    <TITLEBLK><TITLE>IBMIDDoc USER'S GUIDE</TITLE></TITLEBLK>
  </DOCTITLE>
  <AUTHORS>
    <AUTHOR>
      <PERSON>
        <NAME>RICK DENNIS</NAME>
        <ADDRESS>
          <INTERNET>rickd@nando.net</INTERNET>
          <INTERNET>rickd@rtptnotes.raleigh.ibm.com</INTERNET>
        </ADDRESS>
      </PERSON>
    </AUTHOR>
  </AUTHORS>
  <IBMDOCNUM>SH21-0783-02</IBMDOCNUM>
</IBMBIBENTRY>
:
<BIBENTRYDEFS>
  <IBMBIBENTRY ID="BOOK1">
    <DOCTITLE>
      <TITLEBLK><TITLE>IBMIDDoc MIGRATION GUIDE</TITLE>
    </TITLEBLK>
    </DOCTITLE>
  </IBMBIBENTRY>
  <IBMBIBENTRY ID="BOOK2">
    <DOCTITLE>
      <TITLEBLK><TITLE>IBMIDDoc REFERENCE GUIDE</TITLE></TITLEBLK>
    </DOCTITLE>
  </IBMBIBENTRY>
  <IBMBIBENTRY ID="BOOK3">
    <DOCTITLE>
      <TITLEBLK><TITLE>IBMIDDoc TUTORIAL </TITLE></TITLEBLK>
    </DOCTITLE>
  </IBMBIBENTRY>
  <IBMLIBENTRY ID="IDDOCLIB">
    <LIBRARY>
      <TITLEBLK><TITLE>IBMIDDOC</TITLE></TITLEBLK>
    </LIBRARY>
    <PUBLISHER>
      <CORPNAME>IBM</CORPNAME>
    </PUBLISHER>
  </IBMLIBENTRY>
</BIBENTRYDEFS>

```

```

        <CONTAINEDDOCS BIBIDS="BOOK1 BOOK2 BOOK3">
    </IBMLIBENTRY>
</BIBENTRYDEFS>
</PROLOG>

```

For more information about IBMIDDoc bibliographic elements, see “Chapter 13. Bibliographies and citations” on page 119.

Front matter (FrontM)

The front matter contains the title page, notices (such as the edition notice), the preface, the summary of changes, the table of contents, the table list, and the figure list. The elements that can be used in FrontM include:

- EdNotices (see “Notices and Edition notices” on page 86)
- TOC (see “Table of contents” on page 86)
- FigList (see “List of figures” on page 87)
- TList (see “List of tables” on page 87)
- Preface (see “The preface” on page 87)
- SOA (see “Summary of changes” on page 87)
- Abbrev (abbreviations), Abstract, Legend, and others (see)
- D, divisions
- IBMSafety (see “IBM Safety text” on page 88)
- Safety (see “IBM Safety text” on page 88)

The FrontM Style attribute can specify the following values using the Display keyword. You specify these in any combination.

TIPAGE

Causes a draft title page to appear (this should not be used for final camera-ready output).

COVER

Causes a cover, inside title page, and back cover page to appear.

SPINE

Causes the spine to appear after the back cover. The spine contains the IBM Logo, Library, Title, and Version number.

OLDSPINE

Causes the spine to appear after the back cover. The spine Includes the IBM Logo, Library, Title, Version number, and document number.

NORECYCLE

Prevents the recycle logo from appearing on the back cover. Removes the recycled paper logo and text from the back cover of a US English document

REGLOGO

This uses a registered logo on the back cover even when the PrtLoc element is used in the document.

For example, this causes a draft title page, the cover, inside cover, back cover, and spine to be output:

```
<FRONTM STYLE="display='TIPAGE COVER SPINE'">
```

Notices and Edition notices

Anything you want to put on the back of the title page are known collectively as “notices”. Some documents have only an edition notice, which goes at the bottom of the back of the title page; others have notices in addition to the edition notice.

The edition notice involves the EDNotice tag and the Title tag. The Title tag is used immediately following the EDNotice tag, and specifies the text of the heading for the edition notice. A sample edition notice might look like this:

```
<ibmddoc ibmcopyr="1996, 1999">
...
<ednotices><title>First Edition (June 1997)</title>
<p>This edition applies to the IBMDDoc language,
Version 4.2, and to all subsequent releases
and modifications until otherwise indicated in new
editions.</p>
</ednotices>
```

Look at the page following this book’s title page to see what IBMDDoc does with the edition notice. The EDNotice end tag brings in the copyright line, if the IBMCopyr attribute was specified on your IBMDDoc tag (or if the COPRNOTE tag is used).

Other notices

If you have other things to put on the back of the title page besides the edition notice, put them all within a NOTICES tag and its matching end tag before the EDNotice tag (if there is one). It might look like this:

```
<notices><pblk style="lblbox"><title>Note</title>
<p>Before using this information, be sure to read
the general information under <xref refid="notices">.
</p>
<p>This manual was produced using IBMDDoc SGML, the
Adept editor, and processed for print and online using
the ID Workbench.</p>
</pblk></notices>
<ednotices>
```

The NOTICES tag (with its end tag) is actually allowed anywhere in your document. If you put it before the EDNotice tag, the notice associated with it appears on the back of the title page.

Table of contents

The TOC contains the table of contents the document. You can choose to use the GendTitle, for which the title text is generated automatically, or you can enter your own title for this special division by using the Title element.

Typical markup for a TOC:

```
<toc><gendtitle></toc>
```

A TOC with a heading you’ve specified:

```
<toc><titleblk><title>Here's what's in my cool, little booklet</title></titleblk></toc>
```

You can control which heading levels appear in the table of contents by using the MAXTOC attribute on the IBMDDoc tag. For example, the default for the style IBM8X11 is to show headings in the table of contents to heading level 3. Specifying the following will include divisions to heading level 4 to appear:

```
<ibmddoc maxtoc="4" docstyle="ibm8x11">
```

You can also control which headings appear in a table of contents by using the TOC attribute on the Division or other heading tag.

List of figures

Use the FigList element to contain a list of figures that appear in the document. You can choose to use the GendTitle, for which the title text is generated automatically, or you can enter your own title for this special division by using the Title element.

Typical markup for a FigList:
<figlist><gendtitle></figlist>

List of tables

Use the TList element to contain a list of tables that appear in the document. You can choose to use the GendTitle, for which the title text is generated automatically, or you can enter your own title for this special division by using the Title element.

Typical markup for a TList:
<tlist><gendtitle></tlist>

The preface

Use the Preface element to contain explanatory or preparatory information about the document. As with other FrontM elements, you may use the GendTitle element or provide a unique title by using the TitleBlk element. Enter the preface text using the same rules you follow when creating any other division.

```
<preface>
  <specdprolog><gendtitle></specdprolog>
  <dbody>
    <p>This manual...</p>
  </dbody>
</preface>
```

If you don't want to use the generated title, you can enter another title within a TitleBlk element.

```
<preface>
  <specdprolog><titleblk>
    <title>About this book</title>
  </titleblk></specdprolog>
  <dbody>
    <p>This manual...</p>
  </dbody>
</preface>
```

Summary of changes

Use the SOA element to contain a list or description of the important information that has been changed or added since the last revision of the document. SOA is valid in FrontM (recommended) and BackM. For example:

```
<soa>
  <specdprolog><titleblk><title>What's new and different
  </title></titleblk></specdprolog>
  <dbody>
    <p>Changes since the last edition include...</p>
  </dbody>
</soa>
```

Special sections

There are a number of special sections that often occur in publications, typically in either the front matter or the back matter. IBMIDDoc recognizes these special sections (as well as the preface, which we've already covered):

Special Section	Tag
List of Abbreviations	ABBREV
Abstract	ABSTRACT
Bibliography	BIBLIOG
Legend	LEGEND

IBM Safety text

Use IBMSafety to contain any IBM-specific safety concerns or issues that are addressed in your information.

```
<FRONTM>
  <IBMSAFETY SPEC="AUTO">
    <GENDTITLE>
  </IBMSAFETY>
</FRONTM>
```

Note: Not supported by Xyvision.

About back matter (BackM)

The BackM element can contain Appendices, the bibliography, glossary, index, part number index, and Divisions.

The Xyvision and BookMaster transforms provide a part separator for the back matter when the body of the document contained a PART tag. If you want to suppress the automatically-generated part separator, use the following coding on the BACKM tag:

```
<backm style="xpp:(nopart)">
```

Using appendix

The Appendix element contains divisions that contain appendix information. Appendix is valid in BackM.

You must enter titles for the appendixes. For example:

```
<BACKM>
  <APPENDIX>
    <D>
      <DPROLOG>
        <TITLEBLK>
          <TITLE>Whantoozler Tuning Parameters</TITLE>
        </TITLEBLK>
      </DPROLOG>
      <DBODY>
        <P>There are many settings that you can adjust to
          improve Whantoozler performance.
        </P>
      </DBODY>
    </D>
  </APPENDIX>
</BACKM>
```

Using glossary

Use the Glossary element to contain a list of the glossary terms for the document. You can choose to use the GendTitle, for which the title text is generated automatically, or you can enter your own title for this special division by using the Title element. Glossary should contain a GL element, which can contain an explicit list of GLEntry elements, or can use the AUTO value on the SPEC attribute. The AUTO value on the SPEC attribute causes the GL to contain a list of all the GLEntry elements in the document. AUTO is the default value for the SPEC attribute, and is the usual way to create a list of GLEntry elements in a GL.

```
<BACKM>
<GLOSSARY>
  <SPECDPROLOG>
    <GENDTITLE>
  </SPECDPROLOG>
  <DBODY>
    <GL> ... </GL>
  </DBODY>
</GLOSSARY>
</BACKM>
```

See “Chapter 12. Glossaries” on page 115 information about glossary.

Using bibliography (Bibliog)

Use the Bibliog element to contain a list of documents related to the document. Bibliog is valid in both FrontM and BackM.

Bibliog should contain the BibList element, which can contain an explicit list of BibEntry elements, or can use the AUTO value on the SPEC attribute. The AUTO value on the SPEC attribute causes the BibList to contain all of the BibEntry elements in the document. AUTO is the default value for the SPEC attribute, and is the usual way to create a list of BibEntry elements in a BibList.

You can choose to use the GendTitle, for which the title text is generated automatically, or you can specify the title of the Bibliog (“Related Publications”) using the TitleBlk elements.

```
<bibliog>
<specdpolog><gendtitle></specdpolog>
<dbody>
<biblist><bibentry><doctitle><titleblk><title>My Nice
Book</title></titleblk></doctitle></bibentry>
<bibentry><doctitle><titleblk><title>Your Nice Book
</title></titleblk></doctitle></bibentry>
</biblist>
</dbody></bibliog>
```

See “Chapter 13. Bibliographies and citations” on page 119 for more information.

Using part number index (PNIndex)

A Part Number Index can be automatically generated by including the PNIndex element in the BackM element. In most cases, you will use the GendTitle element to include the system-defined title for the PNIndex.

```
<PNINDEX><GENDTITLE></PNINDEX>
```

For more information on these, see “Chapter 21. Creating parts catalog lists” on page 191.

Using Index

The Index element content is normally generated automatically at processing time from the index tags you've sprinkled throughout the source.

```
<INDEX><GENDTITLE></INDEX>
```

For more information about creating indexes, see "Chapter 10. Indexing" on page 97.

The Index should be the last item that has content in a document. Only the reader's comment form should follow the index.

Using reader's comment form (RCF)

A Reader Comment Form can be automatically generated by including the RCF element in the BackM element. In most cases, you will use the GendTitle element to include the system-defined title for the RCF. There should no longer be a "Contacting IBM" section before the RCF; that now belongs in the preface.

For the RCF to be generated, you need to specify the MAINTAINER element information in the prolog of your document:

```
<maintainer>  
<corp>  
<corpname>IBM Corporation</corpname>  
<address>ATTN: Dept 542  
3605 HWY 52 N  
Rochester, MN  
<postalcode>55901-9986</postalcode>  
<phone equip="fax">1-800-555-1212</phone></address>  
</corp>  
</maintainer>
```

In the back matter, you need to include the RCF element; for example:

```
<backm>  
<rcf><gendtitle></rcf>  
</backm>
```

Chapter 9. Revision Elements and Marked Notes

IBMIDDoc has a slick way of indicating changes in your document. Use RevDefs, Rev, and Mark elements to track revisions in your documentation. These elements are valid in the Prolog or DProlog elements of your document. You can also use a marked deletion (MD) element to indicate some text you are going to remove.

Revision and Marked Notes elements include:

- Rev, define a revision level
- RevDefs, define several revisions
- REV attribute, assign a revision level to an element
- MD, indicate words or phrases to be removed; the text is formatted and struck-through.

The following Mark elements are not yet fully supported; they only work for BookMaster output processing.

- Mark
- MkAction
- MKClass
- MKDesc
- MKNote
- MarkList

Using Revisions

In order to define revisions, you must place a Rev element in a RevDefs element. The RevDefs can be in your document prolog to affect the whole document; or in a division DProlog to affect only that division. The Rev element defines the specifics for a single revision that can be used throughout the document or division. You can have more than one revision in a document. This is useful for multiple drafts. Normally, in hardcopy, a revised text is indicated by a vertical bar to the left of the text. You can set different characters for your different revisions. Any element that contains new, changed, or deleted information can refer to the ID attribute value on the Rev element to denote why and how the information has changed.

Defining Revisions in the RevDefs Element

The RevDefs element contains several Rev elements. Each Rev defines a revision, describes the reason for the revision, and states if the revision should be used or ignored during document processing. If the Rev is to be used, the character specified on the char attribute will be printed in the margin to the left of the changed information. If the revision is ignored, no special characters will appear beside the information.

To begin, get to the prolog of your document and insert a RevDefs element; then insert a Rev element. Pick an identifier for your REV tag that makes sense. Suggestions include:

- v4r5 — this indicates version 4, release 5
- r2m1 — this indicates release 2 modification 1; or 2.1

Then, set the IDENT attribute to USE; this enables the revision. If you want to turn off the revisions, set IDENT to IGNORE.

If you want a special character to indicate the revision, type that one character into the CHAR attribute. Normally, a vertical bar is used.

Here is a simple revision definition; only one revision is defined in the document's prolog:

```
<prolog>
...
<revdefs>
<rev id="v4r5" ident="use">
<date>9/9/99</date>
<desc>First draft for v4r5</desc>
</rev>
</revdefs>
...
</prolog>
```

Here is a more complicated set of revision definitions. "v4r5" and "v4r5d2" are enabled, with characters "|" and "+" being used respectively. The other revision for "v4r4" is defined, but set to ignore.

```
<revdefs>
<rev id="v4r5" ident="use">
<date>9/9/99</date>
<desc>First draft for v4r5</desc>
</rev>
<rev id="v4r5d2" code="+" ident="use">
<date>9/10/99</date>
<desc>Second draft for v4r5</desc>
</rev>
<rev id="v4r4" ident="ignore">
<date>9/9/98</date>
<desc>V4R4 changes</desc>
</rev>
</revdefs>
```

Indicating Revisions in the Document Markup

To indicate that text is new or changed; you use the REV attribute on the element that contains the text. The REV attribute refers to the REV elements defined in the RevDefs. The revision characters start with the beginning tag, and continue through to the ending tag.

Using the "v4r5" revision definition from "Defining Revisions in the RevDefs Element" on page 91, the middle list item is changed:

- something old
- something new (or changed)
- + something borrowed

Here's its markup:

```
<ul>
<li>something old</li>
<li rev="v4r5">something new (or changed)</li>
<li>something borrowed</li>
</ul>
```

If you need to revise a section that is part of another revision, IBMIDDoc allows you to nest the revisions; that is, to place revisions inside other revisions. When the formatter encounters the REV attribute for your new revision, it stops printing the

character associated with the old revision and starts printing the character you assigned to the new revision. Then, when the formatter encounters the end tag for your new revision, it resumes printing the character associated with the old revision.

For example, if the whole list was changed for "v4r5", then the middle item was added for the second draft; you would want something like this:

- something old
- something new (or changed)
- something borrowed
- something blue

Here's its coding:

```
<ul rev="v4r5">  
<li>something old</li>  
<li rev="v4r5d2">something new (or changed)</li>  
<li>something borrowed</li>  
<li>something blue</li>  
</ul>
```

The revision markup doesn't evaluate the dates of the revision definitions, it works on nested position. It's totally possible to make a small change first and then make a larger change that encompasses the first change. The inner revision will still show its change bar. Watch out for these; you might need to delete the nested Rev attributes if they should really be part of outer revision.

Marking text for deletion

Sometimes, as part of a revision, you may want to indicate that some text has been deleted, but still leave that text in the document for your reader's convenience. If you precede the text to be deleted with the MD (marked deletion) tag and follow it with the matching end tag, the formatter overstrikes the text with a horizontal line. The MD element is like a phrase. For example:

You may want to eliminate ~~repetitious~~ redundancies.

Here's its markup:

You may want to eliminate <md rev="v4r5">repetitious</md> redundancies.

Creating Collections of Marked Notes

To use marked notes in IBMIDDoc, you enter mark elements that define the marked note collection. These are contained in the RevDefs element, which is contained in the Prolog or DProlog elements. You must define at least one collection in order to use marked notes in your document. These collections of notes can be put in a number of forms, including a table.

Processing Note

Marked Notes currently only work in BookMaster output processing.

These are the elements you'll need to use to create a marked collection of notes:

- Mark
- MKAction

- MKClass
- MKDesc
- MkNote
- MarkList

Using the Mark Element

The Mark element names a marked collection of changes and specifies whether or not the other elements associated with this marked collection are processed when your document is formatted. Mark elements are contained in the RevDefs element, along with Rev elements.

A typical Mark element looks like the one in this example:

```
<mark id="mkv4r5" ident="use">
<desc>v4r5 marked message changes</desc>
</mark>
```

Defining Marked Actions and Classes

MkAction is used to define one or more actions that can be associated with marked notes. These actions can be used with any marked class. MkClass defines a marked note class within MkDesc. You can define as many classes as you need. These class codes are also used on the MarkList element to tell IBMIDDoc which class codes to make part of the marked notes table.

```
<propdefs>
<mkdesc>
<!--Define two classes for marked lists - notes and abends-->
<mkclass name="msg">Msg</mkclass>
<mkclass name="abend">Abend</mkclass>
<!--Define the actions for the changed info-->
<mkaction name="new">New</mkaction>
<mkaction name="change">Changed</mkaction>
<mkaction name="del">Deleted</mkaction>
<mkaction name="rep">Replaced</mkaction>
</mkdesc>
</propdefs>
```

Using the MkNote Element

The MkNote element identifies the actual text of your marked note. Several attributes are used on the MKNote element. These include:

CLASS

defines one or more mark classes to which the marked note belongs.

ACTION

defines one or more actions associated with the marked note.

MKIDS

contains the ID of one or more Mark elements.

ITEM

defines an identifying label for the note, such as a message number or error report.

The content of the tag displays in the description column of the marklist table.

```
<mknote class="msg" action="change" mkids="mkv4r5" item="IDW0012">Hi there!
</mknote>
```

Generating a Collection with MarkList Element

The MarkList element causes a table of marked collection notes to be generated. You can include any notes that you mark in the marked note list, and you can headings for the table. For example, the marked note list can be used to generate a definitive summary of changes. In addition, you can use marked notes to collect information about document content, notes to yourself or others, or references to certain locations in the document that you think will be very important to the reader.

The MarkList element generates a list of marked notes at the place in the document where the MarkList element is specified. Only notes of the specified classes, collections, and actions will be included in the generated list.

```
<marklist mkids="mkv4r5" classes="msg abend" actions="change del rep"
display="item action page desc" classhd="Msg" actionhd="Reason"
itemhd="Msg" lochd="Page" deschd="Message text">
```

A Marked Notes Markup Example

The example that follows illustrates how to use marked notes in IBMIDDoc.

```
<ibmiddoc docstyle="ibmxagd">
<prolog><ibmbibentry><doctitle><titleblk>
<title>My Marked Changes Document for Messages</title>
</titleblk></doctitle></ibmbibentry>
<propdefs>
<mkdesc>
<!--Define two classes for marked lists - notes and abends-->
<mkclass name="msg">Msg</mkclass>
<mkclass name="abend">Abend</mkclass>
<!--Define the actions for the changed info-->
<mkaction name="new">New</mkaction>
<mkaction name="change">Changed</mkaction>
<mkaction name="del">Deleted</mkaction>
<mkaction name="rep">Replaced</mkaction>
</mkdesc>
</propdefs>
<revdefs>
<rev id="revv4r5" ident="use">
<date></date>
<desc></desc>
</rev>
<mark id="mkv4r5" ident="use">
<desc>v4r5 marked message changes</desc>
</mark>
</revdefs>
</prolog>
<body>
<d>
<dprolog><titleblk>
<title>List of changed items</title>
</titleblk></dprolog>
<dbody>
<marklist mkids="mkv4r5" classes="msg abend" actions="change del rep"
display="item action page desc" classhd="Msg" actionhd="Reason"
itemhd="Msg" lochd="Page" deschd="Message text"></dbody>
</d>
<msglist>
<msg rev="revv4r5">
<msgnum>IDW0012</msgnum>
<msgtext>Hi there!</msgtext>
<msgitem class="xpl">
<p>This is a friendly message.
<mknote class="msg" action="change" mkids="mkv4r5" item="IDW0012">Hi there!
</mknote></p>
```

```

</msgitem>
</msg>
<msg rev="revv4r5">
<msgnum>IDW0013</msgnum>
<msgtext>Farewell!</msgtext>
<msgitem class="xpl">
<p>This unlucky message was removed.
<mknote class="msg"
action="del" mkids="mkv4r5" item="IDW0013">Farewell!
</mknote></p>
</msgitem>
</msg>
</msglist></body>
</ibmiddoc>

```

The resulting marklist table will look like the example that follows.

Msg	Reason	Page	Message text
IDW0012	Changed	1	Hi there!
IDW0013	Deleted	2	Farewell!

Chapter 10. Indexing

Creating an index using IBMIDDoc is somewhat like creating a table of contents (TOC). The index is built for you from items in the text you have tagged as index entries. Just as you use the TOC tag to indicate you want a table of contents, you use the INDEX tag in the back matter of your document to show you want the index included.

Indexes are similar to a TOC where a subject and page number are listed. However, the index can provide much more detail than a TOC. It provides a term, subterms, and sometimes synonyms (in the form of “see and see also” references) with page numbers indicating where detailed information can be found on the subject. Indexes are also sorted alphabetically for easy subject retrieval.

You can build an index by tagging the terms you think will be useful for the reader. Place the index tags at the point that the topic occurs to ensure the page references in the index will be correct whenever you format the document. The formatter will automatically create an index in alphabetical order and place the correct page number next to each entry in the index.

Table 14 illustrates the terminology we use in this chapter to describe the elements of an index.

Table 14. Terminology used in discussion of indexing

entry:	<u>appetizers</u> 102 ⋮
subject:	<u>bechamel sauce</u> 13
page references:	<u>cabbage</u> 58, 115 ⋮
primary entry:	<u>eggs</u> 106
secondary entry:	<u>souffles</u> 108
tertiary entry:	<u>chocolate</u> 112 ⋮
entry heading:	<u>meats</u> beef 60 ⋮
	poultry 75
“see also” reference:	<u>See also</u> <u>chicken, turkey</u> ⋮
page range:	<u>sauces</u> 12—14 ⋮
	white sauce
“see” reference:	<u>See</u> <u>bechamel sauce</u>

In this chapter we will discuss the levels of indexes, where to place index entries, how to define index entries, refer to index entries, the use of ss and see also references, and how to control and generate an index. But first we will discuss the basic index structure.

Structuring a basic index

A good index is an indispensable part of any document. This is especially true for reference documents. Because you don't usually read reference information from cover to cover, you need a way to be able to find specific bits of information you need.

To be a good, complete retrieval device, an index must do the following:

- Help readers find information within the document.
- Anticipate how readers will search for information.
- Serve the novice and the expert.
- Show how topics interrelate.
- Tell what the book contains.
- Cross-reference similar terms or concepts.

Before we talk about the actual tags, here are a few index development tips:

- Familiarize yourself with the content, organization, and objectives of the document before you start the indexing process.
- Analyze your audience. Who will be using the book? Are readers likely to be familiar with the book and the product? What will the reader already know?
- Ask yourself, "Does this topic contain information the reader will want to find?" If so, create at least one index entry for that topic.
- Develop an indexing worksheet for each section of your document. On it list major concepts or ideas, major terms defined, acronyms and abbreviations, restrictions and warnings, and cross-references to other information products. Use the worksheet to determine which topics should be main entries and which should be subentries. The worksheet also ensures that important information is not left out of the index.
- Be sure to use both the acronym or abbreviation and its "spelled-out version" as index entries if your document uses them.
- Ask yourself when looking at an index entry, "Are there any commonly used synonyms for this word?" If so, include them in your index as well.
- Make sure that each index entry has no more than two or three references. Use specific subentries to reduce the number of page references and give your reader a more precise pointer to the topic.

This chapter describes how to define and refer to index entries and how to generate an index. It also describes how to create index entries using cross-indexing and how to define see and see-also references.

The IBMIDDoc indexing elements include:

- I1, primary
- I2, secondary
- I3, tertiary
- IdxTerm, index term text
- IRef, index reference

- IdxDefs, index definitions
- Index, index placement

Basic index tagging

There are 3 levels of index entries: Primary (i1), Secondary (i2), and Tertiary (i3). The simplest kind of index entry is a primary entry. A primary entry is the major subject and should be a noun or noun phrase. A primary entry with a page number, is entered with the I1 tag, which says, “This is an index subject at the first level”. It would look like this:

```
<i1><idxterm>dessert sauces</idxterm></i1>
```

A primary index entry may or may not have page number listed. However, if the primary index entry does not have a secondary entry associated with it, the primary tag will automatically have a page number entered. There will be more on index entries and page numbers later in this chapter.

Most indexes run to more elegant structures with primary, secondary, and sometimes tertiary entries. The secondary entry narrows the primary entry into a more specific subject. It may or may not have a page reference. Secondary entries are arranged alphabetically in the index following the primary entry to which they apply. A secondary entry with a page number is entered with the I2 tag.

When you have a large number of subtopics under your primary entry, a secondary or tertiary tag improves index readability. If you’ve ever seen an index where most of the entries are primary and have page numbers, you know how difficult it can be to find the information you need. Using the I2 tag makes an entry stand out and directs the reader’s attention to a topic instead of a mass of numbers.

Tertiary entries are the third, even more specific level for the major topic. A tertiary entry always has a page reference. Tertiary entries are arranged alphabetically following the secondary entry to which they apply.

Placement of index tags

Indexing isn’t as easy as tossing an i1 tag here and an i2 tag there. Believe it or not, there are “rules” for index tagging unless you want an index with only primary index entries. As we discussed before, secondary and tertiary entries make an index more readable. So, unless you have a small index, you’ll want to add a few index levels.

There are two ways to associate secondaries with their primaries and tertiaries with their secondaries. One way is by their position in the source file. The other way is by creating cross references. We’ll tell you all about the position way first.

Position method

The position method has the secondary entries within the tags of the primary entry. Likewise, the tertiary entries are embedded in the secondary entries. The rule when using the position method is you cannot have the secondary entries listed outside the primary entry and the tertiaries cannot be outside the secondary entry. Here’s an example of the position method:

```
<i1><idxterm>dessert sauces</idxterm>  
<i2><idxterm>butterscotch</idxterm></i2>  
<i2><idxterm>hot fudge</idxterm>
```

```

<i3><idxterm>microwave method</idxterm></i3>
<i3><idxterm>stovetop method</idxterm></i3>
</i2>
<i2><idxterm>strawberry</idxterm></i2>
</i1>

```

Here is the formatted result:

```

dessert sauces
  butterscotch 12
  hot fudge
    microwave method 12
    stovetop method 12
  strawberry 12

```

You'll notice that the i3 entries, microwave and stovetop methods, are only listed under hot fudge. This is because the i3 tags are listed inside the i2 hot fudge tag. If you wanted the i3 tags to be under butterscotch, hot fudge, and strawberry, you would have to place the i3 tags inside each one of the i2 tags. So you would have microwave method and stovetop method listed three times each in this example. You'll also notice the page numbers are automatically placed in the formatted example. You don't need to print the document then add the page numbers. It's all done for you.

Cross referencing index entries

As you can see from the examples under "Position method" on page 99, repeating the entire structure of primary and secondary entries before each tertiary entry can be pretty tedious. For this reason, IBMIDDoc has the ID, I1ID, and I2ID attributes on the indexing tags to allow you to get at the structure with just the name you put on the ID.

The ID attribute identifies an index entry within an SGML document. IDs must be unique within a single document. IDs can be up to 64 characters long. IDs must start with an alphabetic character and can contain letters, numbers, dashes (-), or periods (.).

When you put an ID attribute on an I1 or I2 tag, the formatter "remembers" *that entry and any higher level entries associated with it.*

For example, if you had these entries:

```

<i1><idxterm>sauces</idxterm>
<i2 id="mayo"><idxterm>mayonnaise</idxterm>
<i3><idxterm>hard way</idxterm></i3></i2></i1>

```

This associates the ID "mayo" with both mayonnaise *and* sauces.

Then you can enter:

```

<i3 i2id="mayo"><idxterm>blender method</idxterm></i3>
...
<i3 i2id="mayo"><idxterm>food processor method</idxterm></i3>

```

and get exactly the same results as if you had coded this:

```

<i1><idxterm>sauces</idxterm>
<i2><idxterm>mayonnaise</idxterm>
<i3><idxterm>hard way</idxterm></i3></i2></i1>
...
<i1><idxterm>sauces</idxterm>

```

```

+      <i2><idxterm>mayonnaise</idxterm>
+      <i3><idxterm>blender method</idxterm></i3></i2></i1>
+      ...
+      <i1><idxterm>sauces</idxterm>
+      <i2><idxterm>mayonnaise</idxterm>
+      <i3><idxterm>food processor method</idxterm></i3></i2></i1>

```

When you use the reference attribute on the I2 and I3 tags, they pick up the specified level needed from the structure named with the reference name. (You can't use a reference on an I1 tag, because there are no "higher" levels.)

If you want to pick up *all* the levels (that is, you have an identical structure to the one named with the ID attribute), you should use the IREF tag. Because you are picking up all the levels, the IREF tag doesn't need a level indicator of its own. The IREF tag adds a page number to an existing structure.

So if we had many different ways of making mayonnaise with a blender, we could enter:

```

+      <i1><idxterm>sauces</idxterm>
+      <i2 id="mayo"><idxterm>mayonnaise</idxterm></i2></i1>
+      ...
+      <i1ref refids="mayo">
+      ...
+      <i1ref refids="mayo">

```

and we would get this result:

```

+      sauces
+      mayonnaise  20, 23, 26

```

You can even use both an ID and a reference on an I2 or I3 tag, to both pick up the higher level entries (the referenced entry) and then give this whole new structure a name (the ID). (You want to be careful not to confuse yourself, though.)

| Where to put index entries

| Your index entries can be entered just about anywhere; they don't cause any
 | variation in how the text around them is formatted. Here is a list of good places to
 | put index entries to ensure the proper page reference:

- | • Immediately following a heading, after the ending Dprolog tag.
- | • Following the first sentence of a paragraph.
- | • Following the first sentence of a list item or definition description.
- | • Immediately following an XMP tag or FIG tag (in particular, if you are indexing
 | a figure that is going to float, the index tag must be inside the figure).
- | • Immediately following the first ENTRY element of the first ROW of a TABLE,
 | for entries that point to an entire table (this is because the page on which the
 | table will begin isn't determined until the first ENTRY element is processed).

| For example, because the formatter keeps the first few lines of a paragraph on the
 | same page, using the index tags in these places ensures that the page reference
 | picked up for the entry is the same page on which the paragraph starts. If the
 | index entries were placed before the paragraph, they might be processed (and the
 | page number picked up) before the formatter discovers that it has to start a new
 | page for the paragraph.

It is a good idea to put index entries that don't have page references associated with them in one place in the front of your source document. If you scatter them through your document, you will have trouble finding them when you want to change them because the index itself won't give you a page number to help you find them.

And where NOT to put index entries: *Do not* put index entries in the following places:

- Do not place index entries anywhere before the PREFACE tag or between the BODY tag and the first division in the body. They can cause extraneous blank pages in your BookMaster-formatted document if they occur at either of these points.
- Don't put index entries in a table between a ROW tag and an ENTRY tag. This is also a Bookmaster-formatting problem. Put them in the Entry tag.
- Don't put index entries in the middle of a sentence, this raises heck with translation centers.

Defining index entries (central indexing)

You can define index entries in the document's prolog. This is often called "central" or "central-file" indexing.

Use the index definitions (IdxDefs) element to put some or all the index entry definitions in a central place, either the document prolog or a division prolog. This makes it easier for you to maintain your index IDs, since you can maintain them in a central place. Use I1 to define each primary index entry. In IdxDefs, any I2 (secondary) and I3 (tertiary) index entries must be contained in the I1 index entries. The following example shows I1 index entries in IdxDefs for "document structure" and "element":

```
<idxdefs>
<i1 id="ixdocstruct"><idxterm>document structure</idxterm>
<i1 id="ixelement"><idxterm>element</idxterm></i1>
</idxdefs>
```

No page number is associated with index entries located in the IDXDEFS. Use the IREF element or add lower level index entries to set reference points. For example:

```
<i2 refids="ixdocstruct">
<i2 ilid="ixelement"><idxterm>context</idxterm></i2>
```

Creating index entries by cross-indexing

There are many times in indexing when you want to associate the same set of subentries and page references with a group of primary index entries. This process is called "cross-indexing".

You can use the IdxTerm element multiple times to associate the same set of subentries with several I1-level index terms. For example, to associate the same set of I2 and I3 entries with both "cross indexing" and "indexing, cross", you can define an I1 index entry as follows:

```
<i1><idxterm>cross indexing</idxterm><idxterm>indexing, cross</idxterm>
<i2><idxterm>creating</idxterm>
<i3><idxterm>easy way</idxterm></i3></i2></i1>
```

You would get this result:

```

cross indexing
  creating
    easy way 12
...
indexing, cross
  creating
    easy way 12

```

In this example, “cross indexing” is the first index term specified and is considered to be the main index term for this entry.

Cross-indexed primaries do not have to have identical subentries. For example, suppose you want to index all custard pies under both “pies” and “custard pies”; all fruit pies under both “pies” and “fruit pies”; and a general discussion of pies under “pies” alone. To do that, use the following markup, the Index definitions are also used and are in the document’s prolog.

```

<prolog>
...
<idxdefs>
<i1 id="pies"><idxterm>pies</idxterm></i1>
<i1 id="custpies"><idxterm>custard pies</idxterm><idxterm>pies</idxterm></i1>
<i1 id="fruitpies"><idxterm>fruit pies</idxterm><idxterm>pies</idxterm></i1>
</idxdefs>
</prolog>
...
<i2 ilid="pies"><idxterm>general discussion</idxterm></i2>
...
<i2 ilid="custpies"><idxterm>coconut</idxterm></i2>
...
<i2 ilid="custpies"><idxterm>chocolate</idxterm></i2>
...
<i2 ilid="fruitpies"><idxterm>peach</idxterm></i2>
...
<i2 ilid="fruitpies"><idxterm>blueberry</idxterm></i2>

```

which gives results similar to this:

```

custard pies
  chocolate 2
  coconut 1
...
fruit pies
  blueberry 5
  peach 3
...
pies
  blueberry 5
  chocolate 2
  coconut 1
  general discussion 1
  peach 3

```

Defining See and See-also references

If you need see and see-also references in your index, use the SeeID and SeeText attributes with I1 or I2 elements. SeeID points to an index entry specified by an ID attribute.

SeeText points to text that you specify. Use SeeID whenever possible because it ensures that you are referring your reader to a real entry in the index. (You see a question mark in your cross-reference listing if a SeeID specifies an ID that does not exist.) This markup shows the SeeID attribute:

```
<il id="bech"><idxterm>bechamel sauce</idxterm></il>
<il seeid="bech"><idxterm>white sauce</idxterm></il>
```

Processing Notes:

1. The Xyvision formatter does not yet support SeeID and SeeText, so while you can code see and see-also references in your index, they will not appear in Xyvision-formatted output.
2. For a see reference to work correctly in HTML output, the I1 needs to be in the prolog. Any I1 in the body of the document is treated as an index link, giving you a see-also reference. Also, the ID needs to have an IREF in the body of the document.

IBMIDDoc determines whether the reference should be a see or a see-also reference. If “white sauce” has no page references of its own and no secondary entries, the index reference is a see reference, as follows:

```
bechamel sauce

white sauce
  See bechamel sauce
```

However, if “white sauce” has page references or other subentries, the index reference is a see-also reference, as follows:

```
bechamel sauce 29

white sauce 32
  See also bechamel sauce
```

SeeText works the same as SeeID, except that you supply the text you want for the reference, as follows:

```
<I1 seetext="cakes, cookies, pies"><IDXTERM>desserts</IDXTERM></I1>
```

With SeeText, you must ensure that the referenced entries (in this case, cakes, cookies, and pies) can all be found in your index.

If “desserts” has other references, the index entries appear as follows, with the see-also reference listed first in the subentries:

```
desserts
  See also cakes, cookies, pies
```

If both SeeID and SeeText are specified, only the SeeID is used.

If a SeeID points to the ID of a secondary or tertiary entry, as in the following example, IBMIDDoc constructs the full cross reference for you:

```
<I1><IDXTERM>sauces</IDXTERM><I2 ID="vinaig"><IDXTERM>vinaigrette</IDXTER
:
<I1 SEEID="vinaig"><IDXTERM>oil and vinegar dressing</IDXTERM>
```

The cross-reference looks similar to this:

```
oil and vinegar dressing
  See sauces, vinaigrette
:
sauces
  vinaigrette 83
```

When you use SeeID in one index entry to refer to another index entry that has several index terms defined, the “See” or “See also” text generated in the index shows only the main (first) index term.

Note: If a SeeID attribute points to an I1 or I2 element that specifies cross indexing (has multiple IdxTerm elements), the resulting see or see-also reference points only to the first (main) IdxTerm element. Because of that, you should select the main entry carefully for any index elements that specify cross indexing.

Generating the index

The INDEX tag, like the TOC tag in the front matter, shows that you want your index placed in the back matter. In the BACKM (back matter) section, insert an INDEX tag. Also insert a GENDTITLE tag. It generates the level 1 heading “Index” for you and then includes the sorted and formatted index.

It would look like this:

```
<backm>
<index>
<gendtitle>
</index>
</backm>
```

and that’s all there is to that.

You can override the index heading text like so:

```
<index>
<titleblk>
<title>My Cute, Highly Retrievable Index</title>
</titleblk>
</index>
```

To get an idea of what your final index will look like, just look at the index in this book; it was done using these tags.

Creating a master index

A master index incorporates the index entries from other documents and combines them into one central place for the user. The master index provides the name of the document and the page number where the information about the index entry can be found.

To create a master index for a set of documents, do the following:

1. Each of the documents that contribute to a master index needs to have the master index prefix specified in their prolog. In each contributing book, use a MasterIndexInfo element containing a MasterIndexPrefix element to specify the prefix code. For example, for a user guide, you might want to use the prefix USERGD; for a reference, you might want to use the prefix REF. For example:

```
<masterindexinfo>
<masterindexprefix>USERGD</masterindexprefix>
</masterindexinfo>
```

The prefix should be something short, generally less than 10 characters. We also recommend having no spaces. When an entry in the master index prints, they will look something like the following. This sample master index has three

books: USERGD for a user's guide, INTRO for an introduction, and PLAN for a planning guide. The page number after the prefix is the page number in the corresponding book.

configuring INTRO-2, USERGD-12
changing PLAN-34

deleting USERGD-39

2. To format a master index using Xyvision:

- a. This step is optional. The master index support allows you to link directly from a PDF of the master index to that page in the PDF of the contributing document. To do this, you can specify an ExternalFileName element in each contributing document to be contained in the master index. Specify only the file name of the document. Do not specify a file extension. This only works for Xyvision-formatted documents. For example, this specifies the name of this document is myusergd. If you will be placing the PDF files on an AIX[®] server, remember that the file names are case-sensitive.

```
<externalfilename>myusergd</externalfilename>
```

- b. Each of the documents to be contained in the master index needs to be formatted for PostScript using Xyvision. By specifying the master index prefix elements, the Xyvision formatter generates a PostScript file and a master index file (file extension MDX) for each document. The document must also have an Index tag, and be processed so that an index is generated (avoid the NOINDEX option).

- c. Once all of the contributing documents have been formatted and the master index files (MDX) have been created, you create a master index document that imbeds each of the individual master index files. You indicate that this is a master index document by coding a MasterIndex element containing a MasterIndexObj element for each MDX file to be included. Each MDX file needs to be declared; declare the MDX file as a "graphic" entity with a notation of "mindex".

- d. Format the master index file for PostScript using Xyvision to create the master index document.

- e. If you want, create Adobe Acrobat PDFs from the PostScript files for the master index and the contributing documents.

3. To format a master index using BookMaster[®]:

- a. Each of the documents to be contained in the master index needs to be formatted for PostScript using BookMaster. You will need to format the documents without using the ID Workbench. The master index files from BookMaster processing are not returned to the OS/2[®] IDWB client.

- b. Transform each contributing document to BookMaster.

- c. Transform the master index document to BookMaster.

- d. Upload each converted document, with its artwork, to VM and process them using IDPS. For each contributing book, specify the correct BookMaster master index options for IDPS to create the master index; either:

```
Master index ==> filename
```

or:

```
SYSVAR (M filename)
```

where *filename* is the name for the contributing document's master index file. This creates the file: *filename* DSMINDEX.

- e. For each contributing document, you will need to add an imbed command for the master index file in the following format. Add these lines just before the INDEX tag in the master index document.

```
.* set the name of the DSMINDEX file
.namefile name=filename cms='filename dsmmidx'
.* imbeds the index source
.im filename
```

This shows an example prolog for a contributing document. The ExternalFileName specifies the file name of this document: idfgsmst, without the file extension. The MasterIndexInfo and MasterIndexPrefix elements indicate the prefix is GSUG (the prefix used for this book).

```
<ibmbibentry><doctitle>
<library><titleblk>
<title>ID Workbench</title>
</titleblk></library>
<titleblk>
<title>Getting Started and User's Guide</title>
</titleblk></doctitle>
<externalfilename>idfgsmst</externalfilename>
</ibmbibentry>
<masterindexinfo>
<masterindexprefix>GSUG</masterindexprefix>
</masterindexinfo>
```

In the master index document, the contributing MDX master index files must be declared. This example shows two sets of declares. The "mindex" declares are for the master index files; the "sgmldoc" declares are for cross-book links (using the citations).

```
<!ENTITY instidx SYSTEM "idfinmst.mdx" ndata mindex>
<!ENTITY planidx SYSTEM "idfplmst.mdx" ndata mindex>
<!ENTITY gsugidx SYSTEM "idfgsmst.mdx" ndata mindex>
<!ENTITY inst SYSTEM "idfinmst.idd" ndata sgmldoc>
<!ENTITY plan SYSTEM "idfplmst.idd" ndata sgmldoc>
<!ENTITY gsug SYSTEM "idfgsmst.idd" ndata sgmldoc>
```

This shows a sample master index document:

```
<ibmiddoc>
<prolog><ibmbibentry><doctitle>
<library><titleblk>
<title>ID Workbench</title>
</titleblk></library>
<titleblk>
<title>Master Index</title>
</titleblk></doctitle>
</ibmbibentry>
<bibentrydefs>
<ibmbibentry docname="gsug" id="gsug"><doctitle><titleblk><title>
ID Workbench Getting Started and User's Guide</title></titleblk></doctitle>
</ibmbibentry>
<ibmbibentry docname="inst" id="inst"><doctitle><titleblk><title>
ID Workbench Workstation Installation Guide</title></titleblk></doctitle>
</ibmbibentry>
<ibmbibentry docname="plan" id="plan"><doctitle><titleblk><title>
ID Workbench Planning and Host Installation Guide</title></titleblk></doctitle>
</ibmbibentry></bibentrydefs>
</prolog>
<frontm style="display='cover'">
<toc><gendtitle></toc>
</frontm>
<body>
<d>
<dprolog><titleblk>
```

```

+      <title>Master Index Prefix Codes</title>
+      </titleblk></dprolog>
+      <dbody>
+      <dl>
+      <dlentry><term>GSUG</term>
+      <defn><cit bibid="gsug"></defn>
+      </dlentry>
+      <dlentry><term>INST</term>
+      <defn><cit bibid="inst"></defn>
+      </dlentry>
+      <dlentry><term>PLAN</term>
+      <defn><cit bibid="plan"></defn>
+      </dlentry>
+      </dl>
+      </dbody></d>
+      </body>
+      <backm>
+      <masterindex>
+      <specdprolog><gendtitle></specdprolog>
+      <masterindexobj obj="gsugidx">
+      <masterindexobj obj="planidx">
+      <masterindexobj obj="instidx">
+      </masterindex></backm>
+      </ibmidoc>

```

Chapter 11. All about linking

Hypertext links (we'll just call them links from now on) connect elements in one part of an online document to elements in another part of the same document or a separate online document.

Linking 101

Think of links as you would think of cross references in a printed document. For example, while reading about Henry David Thoreau in the encyclopedia, a reader comes across a reference to another topic: "See also *Ralph Waldo Emerson*". What does the reader do? Keeps a finger on the page that describes Thoreau and turns back to the new reference. The reader has just created a link from one part of the document to another.

In printed documents, a reader turns to related information. In online documents, IBMIDDoc creates a link to related information, and the online reader can then display that information. The way a reader selects a reference, that is, asks the online browser to display it, is usually by double-clicking on some highlighted text. Web browsers, Adobe Acrobat, and BookManager Read are the typical online browsers for which we create books or articles.

The following terms are used for the different types of links:

Cross-reference links

These are explicit links that IBMIDDoc creates between cross references that use the XREF tag and referenced information within one document.

Author-defined links

These are explicit links that you specify with the L (link) tag and others. These can be within a document or from one document to another.

Associative links

These are links that BookManager creates automatically. You do not need to specify them. They typically come from glossary terms

Implicit links

These links are derived from the structure of the markup. Tables of contents and index entries are examples of implicit links derived from the SGML markup structure.

Creating links within a document

Generic links support the identification of a "hot spot" as one anchor of the link, and the specification of a target as the other end of a link. Within the same document, you can use the XREF tag or the L tag to create links. The XREF tag uses the heading text or figure number, for example, as the "hot spot" text. If you want to use your own text for the link, perhaps for readability or to have the link fit better in a sentence, use the L (link) element. The content of the L element is the "hot spot" text, and the linkend attribute specifies the ID of the other anchor.

This next example shows how to make both an XREF tag and an L tag link to a heading. The L tag's LinkEnd attribute references the division's ID attribute.

```

<d id="xrefhyl">
<dprolog><titleblk>
<title>All about linking</title>
</titleblk></dprolog>
<dbody>
<p>Hypertext links (we'll just call them links from
now on) connect elements in one part of an online
document to elements in another part of the same document
or a separate online document. </p>
...
<p>Sometimes you need to <l linkend="xrefhyl">link</l> to
other topics.</p>
...
<p>See <xref refid="xrefhyl"> for ways of creating links.</p>

```

The XREF appears as a cross reference with a page number in a hardcopy document. In an online document, the heading text “All about linking” is highlighted and selectable. The L type of link has no representation at all in a hardcopy document. But, in an HTML, PDF, or BookManager online version, the text “link” is highlighted and selectable. When you select the link, the browser jumps to the division. Here’s how it appears:

Formatted Example

Sometimes you need to link to other topics.

See “Chapter 11. All about linking” on page 109 for ways of creating links.

End of Formatted Example

You can link to other elements in the same document using this same linking mechanism. The target of an explicit link should be to an ID on the outer container (for example, D, MSG, LE, FIG, TABLE) and not the title text or caption text.

Linking to another document

There are several classes of inter-document links to other documents:

- Linking to another IBMIDDoc document.
This type of link is interpreted based on the output being produced for the linking (from) document. The link produced in the output of the linking document assumes the same type of processing is done for the target document. So Xyvision documents produced from IBMIDDoc will link to the Xyvision output of the target document, HTML documents to other HTML documents, IPF documents to other IPF documents, and so forth.
- Linking to a specific output type of document.
This type of link specifies the type of document to be linked (for example, HTML or IPF). The type of output processing done to the linking document does not affect the type of the target, which remains the same.

Note: If the ID Workbench output transform application finds an ID that conforms to BookMaster’s ID rules (seven characters or less, no special characters, starting with an alphabetic character), it will preserve the ID when it transforms the SGML markup to BookMaster markup. This enables both cross-document links using BookManager and cross-document references between IBMIDDoc documents, and between IBMIDDoc documents and native BookMaster documents.

Citation link to an IBMIDDoc document

You use the CIT element to reference another document as a whole. If the bibliographic entry specifies an entity declaration for the bibliographic entry and uses the DocName attribute, a link to that other document is created. For BookManager to use this link properly, specification of IBMDocNum is also necessary.

This markup generates the appropriate cross-document link markup in Xyvision PDFs and BookManager.

```
<!ENTITY bk2ent SYSTEM "xdoclnk2.idd" NDATA sgmldoc>
...
<ibmbibentry docname="bk2ent" id="bk2">
<doctitle><titleblk><title>Target Document (XD0CLNK2)</title>
</titleblk></doctitle>
<ibmdocnum>SC41-0002</ibmdocnum>
</ibmbibentry>
...
<p>Title citation link: See the <cit bibid="bk2"> for this
information.</p>
```

Linking to an HTML (or web) document

A link to an HTML document (or location within an HTML document) is accomplished by referencing its URL. This is done by referencing, by ID, a notation location or NOTLOC element with a specified notation of URL which contains the URL.

Here is an example that will link to the main IBM web page.

```
<ldescs>
<notloc id="ibm" notation="url">http://www.ibm.com</notloc>
<ldescs>
</prolog>
...
<p>You should try linking to the
<l linkend="ibm">IBM home page</l>.</p>
```

Formatted Example

You should try linking to the IBM home page.

End of Formatted Example

When processing for HTML or Xyvision PDF output, the appropriate anchor markup is generated. When processing for other outputs, the URL is ignored.

Here is another example that will links to a PDF version of a book:

```
<ldescs>
<notloc id="gsugpdf" notation="url">
http://w3.rchland.ibm.com/projects/IDWB/documents/idfgsmst.pdf</notloc>
<ldescs>
</prolog>
...
<p>You should try linking to the
<l linkend="gsugpdf">PDF version of the IDWB Getting Started book</l>.</p>
```

Formatted Example

You should try linking to the PDF version of the IDWB Getting Started book.

End of Formatted Example

Linking to headings in a PDF document

We showed you in “Citation link to an IBMIDDoc document” on page 111 how to link to a Xyvision PDF document as a whole. How would you like to link to a specific heading, figure, or table within a Xyvision PDF book? Here’s how!

You find the ID of that heading, figure, or table on the target book, and set up your LDesc and NameLoc tags to point to those IDs. Then you make Links to those NameLoc definitions, and Xyvision and Acrobat do the rest.

For example, you have the following things you want to reference in a book named “fred.pdf”:

- Heading ID: barney
- Figure ID: betty
- Table ID: wilma

The declaration for the SGML document must have the name matching the PDF name; the extension can be IDD. The NameLoc tags set up the links that are used later:

```
<!ENTITY fred SYSTEM "fred.idd" NDATA sgml doc>
...
<l desc>
<name loc id="barneyintro" objtype="head">
<nmlist docname="fred">barney</nmlist>
</name loc>
<name loc id="bettyphoto" objtype="fig">
<nmlist docname="fred">betty</nmlist>
</name loc>
<name loc id="wilmachart" objtype="table">
<nmlist docname="fred">wilma</nmlist>
</name loc>
</l desc>
...
<p>Barney's hobbies are listed <l linkend="barneyintro">here</l>.
This is Barney's wife, <l linkend="bettyphoto">Betty</l>.
Wilma divides her time <l linkend="wilmachart">this way</l>.</p>
```

Linking to an IPF document

A direct link to an IPF book may be coded similar to this example:

```
<!ENTITY sctagent SYSTEM "SCTAGENT.INF" NDATA IPFINF>
...
<name loc id="ID907" objtype=book>
<nmlist nametype=entity>sctagent</nmlist>
</name loc>
...
<p>This paragraph links to an IPF online book.
See this <l linkend=ID907 style="IPF: (data='sctagent.inf'
object='view.exe' reftype=launch)">IPF topic</l> for more info.</p>
```

In the example, the NAMELOC only defines the ID referenced by the link. The entity declaration performs no function at all. This coding reflects the coding that should be used in the future when the need for the passthrough attributes has been eliminated.

This coding will generate the appropriate IPF code but does not produce usable Xyvisoin PDF, BookManager, or HTML code. Cross document (XREF) references are not supported in IPF.

For IPF, the citation element alone does not generate a cross-document link. The link must be coded with the appropriate IPF passthrough attribution to generate a launch-type link, launching the IPF viewer against the desired IPF file.

```
<!ENTITY bk2ent SYSTEM "xdoclnk2.idd" NDATA sgmldoc>
...
<nameloc id="lk1" objtype="BOOK">
  <nmlist nametype=entity>bk2ent</nmlist>
</nameloc>
...
<ibmbibentry docname="bk2ent" id="bk2">
  <doctitle><titleblk><title>Target document (XDOCLNK2) </title>
</titleblk></doctitle>
  <ibmdocnum>SC41-0002</ibmdocnum>
</ibmbibentry>
...
<p>Title citation link: See the <cit bibid="bk2" props="#not IPF">
<l linkend=lk1 props="IPF" style="ipf:(data='xdoclnk2.inf'
reftype='launch' object='view.exe')">
  Target document (XDOCLNK2)
</l>
for this information.</p>
```

Chapter 12. Glossaries

Glossaries are similar to definition lists, in that you pair terms with their definitions, using the Term and Defn (definition) element. You begin your glossary with the Glossary element, which generates a head level 1 with the generic heading "Glossary." The Glossary element, can have an ID attribute for cross-referencing. You can use your own title instead of "Glossary," so you might have "Definition of terms".

The glossary typically goes in the back matter, in the Backm element's content. Here are the typical tags for the glossary section:

```
<backm>
...
<glossary>
<specdprolog><gendtitle></specdprolog>
<dbody>
  <gl>...</gl>
</dbody>
</glossary>
...
</backm>
```

If you like, you can enter ordinary text after the DBody before you actually begin your glossary list with its entries. You start the glossary list with a GL tag and end it with its matching end tag. Within the glossary list, you use the GEntry, Term, and Defn tags to mark up the terms and their descriptions. The description can be many paragraphs. For example, the glossary in this book was entered in part like this:

binding edge. The edge of a page to be bound, stapled, or drilled.

Here's its markup:

```
<gl>
<gentry><term>binding edge</term>
<defn>The edge of a page to be bound, stapled, or drilled.
</defn>
</gentry>
</gl>
```

If your term has multiple definitions, just enter another set of Defn elements in the Gentry. For example:

cat. (1) cute, furry mammal that purrs when rubbed the right way (2) owner of the house in which it dwells, any people sharing the dwelling are the caretakers

Here's its markup:

```
<gl>
<gentry><term>cat</term>
<defn>cute, furry mammal that purrs when rubbed the
right way</defn>
<defn>owner of the house in which it dwells, any people
sharing the dwelling are the caretakers</defn>
</gentry>
</gl>
```

Defining Terms

Use the GLEntry element to define a term used in your document. GLEntry contains the glossary term and one or more Defn elements, each of which contains a definition for the term. You can define terms in the document prolog or in a glossary list.

Glossary entries for IPF output become divisions displayed in popup windows.

In your document, you can use a Termdef attribute on a Term element that points to the ID of the glossary term. In HTML, IPF, and Window's Help, this generates a link from the Term element to the term in the glossary.

Separating letter groups in a glossary

The retrievability of items in your glossary will be improved if you use the GLBlk (glossary block) elements for alphabetic groups of terms. For example:

A

aardvark. long-nosed doglike creature.

B

bat. flying mouse

Here's its markup:

```
<gl>
  <glblk><title>A</title>
  <glentry><term>aardvark</term>
    <defn>long-nosed doglike creature.</defn>
  </glentry>
</glblk>
<glblk><title>B</title>
  <glentry><term>bat</term>
    <defn>flying mouse</defn>
  </glentry>
</glblk>
</gl>
```

Defining Classes for Terms

You can also define classes of glossary terms, and assign properties to those classes, as shown in the following example, where the terms are defined as being in either class *odwords* or class *duckwords*:

```

:
:
<CLASSDEF ELETYPES="GLENTRY" CLASSNAME="odwords">
  <TITLE>OTHER D-WORDS</TITLE>
  <SEM>OTHER WORDS BEGINNING WITH A D</SEM>
</CLASSDEF>
<CLASSDEF ELETYPES="GLENTRY" CLASSNAME="duckwords">
  <TITLE>OTHER D-WORDS</TITLE>
  <SEM>WORDS ABOUT DUCKS</SEM>
</CLASSDEF>
:
:
<GL>
  <GLENTRY CLASS="odwords"><TERM>December</TERM>
    <DEFN>A month that is often cold and dreary.</DEFN></GLENTRY>
  <GLENTRY CLASS="odwords"><TERM>duty</TERM>
```

```

    <DEFN>What one must do in life.</DEFN></GENTRY>
<GENTRY CLASS="duckwords"><TERM>ducks and drakes</TERM>
    <DEFN>The game of skimming stones across water.</DEFN></GENTRY>
<GENTRY CLASS="duckwords"><TERM>ducky</TERM>
    <DEFN>Very well, as in <q>Just ducky, thanks.</q></DEFN></GENTRY>
<GENTRY CLASS="odwords"><TERM>Durango</TERM>
    <DEFN>City in Colorado.</DEFN>
    <DEFN>City somewhere else.</DEFN></GENTRY>
</GL>

```

For more information about defining classes, see “Chapter 19. Property and Class Definition” on page 177.

Chapter 13. Bibliographies and citations

This section introduces the IBMIDDoc bibliographic elements. These elements identify books and documents, and allow you to create citation references (and links) as well as traditional “back of the book” bibliographies.

IBMIDDoc has two sets of bibliographic elements.

- The BibEntry elements contain non-IBM bibliography information
- The IBMBibEntry elements contain IBM bibliographic information. IBMBibEntry elements should be used to describe all IBM documents.

In most respects, IBM and non-IBM bibliographic elements are the same. Unless otherwise noted, the information contained in this chapter which refers to a non-IBM-specific bibliographic element (for example BibEntry) also applies to both the IBM-specific bibliographic element (IBMBibEntry).

These elements use the bibliographic entries contained in the BibEntryDefs element by referring to the individual BibEntry’s ID.

- Cit — title citation
- BibList — bibliography list
- LibEntry — library entry

Identifying books and documents

You identify books and documents as bibliographic items using BibEntryDefs elements. The BibEntryDefs element can be used in a Prolog (for your whole document to use) or in a DProlog (for that division to use). BibEntryDefs contain one or more BibEntry elements, which contain individual bibliographic entries.

You can set up a file entity to contain a library of BibEntry elements, and then imbed that file in the BibEntryDefs. This is useful when all the books in your library use the same bibliographic information to create bibliographies.

Each BibEntry (or IBMBibEntry) element can contain extensive bibliographic information about a publication. Each BibEntry must contain a DocTitle element. It can also contain Author, Desc, Publisher, PrtLoc, DocNum, PartNum, ISBN, and PubID. An IBMBibEntry element can contain the same elements, plus two other elements, IBMDocNum and IBMPartNum, which contain IBM-specific publication information.

Here is the markup for defining two books. The IDs can be used to generate citations and bibliographies. The DOCNAME attributes point to an external entity that declares a book to cross-reference to, using a CIT (citation) tag.

```
<bibentrydefs>
<ibmbibentry docname="fruitbats" id="fruitybat"><doctitle>
<titleblk><title>The Care and Feeding of Fruit Bats
</title></titleblk></doctitle>
<ibmdocnum>ZZ99-9876-00</ibmdocnum>
</ibmbibentry>
<ibmbibentry docname="vampbats" id="vampbat">
<doctitle><titleblk><title>The Vampire Bat, a much
```

```

|      maligned creature</title></titleblk></doctitle>
|      <ibmdocnum>ZZ99-1234-00</ibmdocnum>
|      </ibmbibentry>
|      </bibentrydefs>

```

Here are the declarations for the two books:

```

|      <!ENTITY vampbats SYSTEM "vampbats.idd" ndata sgml doc>
|      <!ENTITY fruitbats SYSTEM "fruitbats.idd" ndata sgml doc>

```

Using title citations

The Cit element represents a citation of another document. The Cit can either refer to a BibEntry or LibEntry by ID, or include a BibEntry or LibEntry element. The example that follows is a Cit that references the books defined in “Identifying books and documents” on page 119:

See this book *The Care and Feeding of Fruit Bats* and that book *The Vampire Bat, a much maligned creature*, ZZ99-1234-00 for serious bedtime reading.

Here’s its markup:

```

|      See this book <cit bibid="fruitybat"> and that book
|      <cit bibid="vampbat" form="full"> for serious bedtime reading.

```

Here’s an example citation that is self-contained:

See these books for a good read and then a weird read: *Tom Sawyer* and *System/36: Concepts and Programmer’s Guide*

Here’s its markup:

```

|      See these books for a good read and then a weird read: <cit>
|      <bibentry><doctitle><titleblk><title>Tom Sawyer</title>
|      </titleblk></doctitle></bibentry></cit> and <cit>
|      <ibmbibentry><doctitle>
|      <library><titleblk><title>System/36</title></titleblk>
|      </library>
|      <titleblk><title>Concepts and Programmer's Guide</title>
|      </titleblk></doctitle></ibmbibentry></cit>

```

The default document style determines the appearance, or form, of the citation. You can specify the form of the Cit by using the FORM attribute. This allows you to specify that only the title or document number will be displayed. You can also use the FORM=FULL specification to cause the entire bibliographic entry to be displayed.

When LibEntry is specified in a Cit element, the LibEntry is collected for use in generated bibliography.

Citations

When the Cit element is used in IBMIDDoc, the link to the target is automatically generated at processing time. Citations must use bibliographic entries to define the target of the citation. If the bibliographic entry specifies an entity using the DOCNAME attribute, the citation may also be treated as a link as well as a citation by the document name of the target. All targets must be defined in a BibEntryDefs element in a Prolog, DProlog, or SpecDProlog element. A central file containing a master BibEntryDefs element with all of the IBMBibEntry and BibEntry elements for a product library can be referenced using an entity reference in your document.

The Cit element uses the BIBID attribute to reference the ID value of the target citation reference that is defined in the IBMBibEntry or BibEntry element contained in a BibEntryDefs element. The example that follows illustrates how to use these elements.

```
<!ENTITY fredbook SYSTEM "fred.idd" ndata sgmldoc>
...
<bibentrydefs><ibmbibentry docname="fredbook" id="fred">
<doctitle><titleblk><title>Phred's Guide to Phishing
</title></titleblk></doctitle></ibmbibentry></bibentrydefs>
...
<p>See <cit bibid="fred"> for most excelent tips on
catching walleyes.</p>
```

Automatically generating a bibliography

In most cases, bibliographic references are listed in individual BibEntry elements that are contained in the BibEntryDefs element in the Prolog element. Each of these bibliographic references usually has an ID attribute. This ID allows the BibEntry to be referred to in Cit and LibEntry elements. The LibEntry element contains the IDs of the BibEntry elements that make up that library.

When bibliographic elements are arranged as described in the preceding paragraph, a Bibliography will be generated when the SPEC attribute value is AUTO.

```
<BIBLIOG>
<P>A list of the documents referred to in this book....
follows.
<BIBLIST SPEC="AUTO" FORM="full"><GENDTITLE>
```

Defining library entries

LibEntry and IBMLibEntry elements are used to structure and organize information about libraries and collections of documents. You can use IBMLibEntry elements within IBMBibEntryDefs (or BibEntryDefs) , BibList, and Cit elements. The LibEntry element performs the same function as an IBMLibEntry element, but applies only to non-IBM documents.

IBMLibEntry contains the Title of the library. IBMLibEntry can also contain Publisher, PrtLoc, IBMBofNum, IBMPartNum, Prod, ISBN, PubID, ContainedDocs, and Desc elements. IBMLibEntry indicates which books are in the library it describes by referencing the IBMBibEntry elements that describe them. It can contain a list of individual IBMBibEntry elements, or it can contain elements and links that refer to IBMBibEntry elements contained in BibEntryDefs. These entries are referenced using the CONTAINEDDOCS attribute.

The IBMLibEntry in the example that follows shows the ContainedDocs element that references two books:

```
<bibentrydefs>
<ibmlibentry>
<library><titleblk><title>BS/300</title></titleblk>
</library>
<ibmbofnum>SB0F-1234-0</ibmbofnum>
<containeddocs bibids="booka bookb"></ibmlibentry>
<ibmbibentry id="booka"><doctitle><titleblk><title>
BS/300 Guide</title></titleblk></doctitle></ibmbibentry>
<ibmbibentry id="bookb"><doctitle><titleblk><title>
BS/300 Reference</title></titleblk></doctitle></ibmbibentry>
</libentry>
```

```

| <library><titleblk><title>Back'n'Recovery</title>
| </titleblk></library>
| </libentry>
| </bibentrydefs>

```

Linking BibEntry elements and other documents

A BibEntry and all references to it are links to the document the BibEntry describes. Using the DOCNAME attribute on the BibEntry element allows you to refer to an SGML entity that represents the document being described in the BibEntry. When this attribute is used, any element that refers to the BibEntry will also become a link to the document represented by the SGML entity referred to by this DOCNAME attribute.

An example of using BibEntry and BibEntryDefs

The example that follows illustrates a common usage of the BibEntryDefs and BibEntry elements.

```

:
:
<PROLOG>
:
:
<LDESCS>
  <NAMELOC ID="UGNAME" OBJTYPE="BOOK">
    <NMLIST DOCNAME="UGX">USERGIDE</NMLIST>
  </NAMELOC>
</LDESCS>
<BIBENTRYDEFS>
  <IBMBIBENTRY ID="BOOK1">
    <DOCTITLE>
      <TITLEBLK><TITLE>IBMIDDOC MIGRATION GUIDE</TITLE>
    </TITLEBLK>
    </DOCTITLE>
  </IBMBIBENTRY>
  <IBMBIBENTRY ID="BOOK2">
    <DOCTITLE><TITLEBLK><TITLE>IBMIDDOC REFERENCE</TITLE></TITLEBLK>
    </DOCTITLE>
  </IBMBIBENTRY>
  <IBMBIBENTRY DOCNAME="UGX" ID="BOOK3">
    <DOCTITLE>
      <LIBRARY><TITLEBLK><TITLE>IBMIDDOC</TITLE></TITLEBLK></LIBRARY>
      <TITLEBLK><TITLE>IBMIDDOC USER'S GUIDE</TITLE></TITLEBLK>
    </DOCTITLE>
    <AUTHORS><AUTHOR><PERSON>
      <NAME>RICK DENNIS</NAME>
      <ADDRESS>
        <INTERNET>rickd@namdo.net</INTERNET>
        <INTERNET>rickd@rtppnotes.raleigh.ibm.com</INTERNET>
        <VNET>RICKD AT RALVM13</VNET>
        <PHONE>919-254-4062</PHONE>
      </ADDRESS>
    </PERSON></AUTHOR>
    </AUTHORS>
    <PUBLISHER>
      <CORPNAME>IBM CORPORATION</CORPNAME>
    </PUBLISHER>
    <IBMDOCNUM>SH21-0783-01</IBMDOCNUM>
  </IBMBIBENTRY>
  <IBMLIBENTRY>
    <LIBRARY ID="IDDOCLIB">
      <TITLEBLK><TITLE>IBMIDDOC LIBRARY</TITLE></TITLEBLK>
    </LIBRARY>
    <CONTAINEDDOCS BIBIDS="BOOK1 BOOK2 BOOK3"></IBMLIBENTRY>
  </IBMLIBENTRY>
</BIBENTRYDEFS>

```

```

</PROLOG>
<BODY>
<D>
:
:
<P>FOR MORE INFORMATION, SEE <XREF REFID="UGNAME" OBJTYPE="BOOK"></P>
</DBODY>
</D>
</body>
:
:

```


Chapter 14. Program Syntax Definitions

IBMIDDoc contains a number of elements that are used to define program syntax diagrams. These elements include:

- Syntax; contains the diagram and the markup.
- Delim; delimiters, such as commas and parentheses.
- Fragment; a portion of the diagram.
- FragRef; a reference to a fragment.
- Group; gathers parts of the diagram together.
- Kwd; a keyword, such as something that must be entered or chosen.
- Oper; an operator, such as a plus sign.
- RepSep; a way of repeating and specifying a separator for the repeat.
- Sep; a separator.
- SynBlk; combines groups together.
- SynNote; a diagram footnote.
- SynPh; a syntax phrase.
- Var; a variable, such as a file name.

This chapter contains some general information about creating syntax definitions, and examples of the IBMIDDoc markup are used to obtain the formatted output.

Migration Note

If you are familiar with Bookmaster syntax definitions, you will notice several differences when using IBMIDDoc syntax definitions:

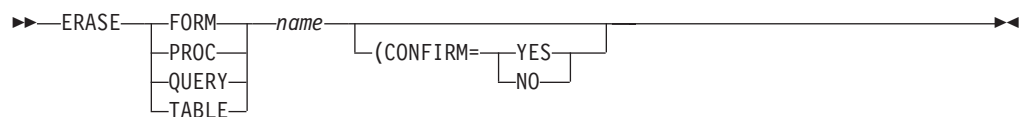
- RepSep definitions
- Descriptions (now within the group or fragment)
- Most BookMaster style attributes are gone; except "composite"
- Group and Syntax elements can have titles

For conversion purposes, the outermost group element, which contains all of the other groups in a typical syntax definition, may need to be broken up into several groups, in order to accommodate BookMaster conversion limitations.

Defining the syntax diagram

IBMIDDoc provides elements and attributes that let you create program syntax diagrams. A sample syntax diagram is shown below:

SAA CPI Database Reference



The sample diagram includes the following syntax diagram elements:

Syntax

The diagram itself. In the sample diagram, the diagram is set off from the text by a labeled box and contains the diagram title, “SAA CPI Database Reference.” IBMIDDoc provides the Syntax element and its end element to define a syntax diagram. The Syntax element has attributes that let you specify the characteristics of the diagram.

Groups

A collection of items or of other groups. One group in the sample diagram comprises the keywords FORM, PROC, QUERY, and TABLE. Another group in the sample comprises the two keywords YES and NO.

Items Individual elements inside the diagram. In the sample diagram, the items are keywords (the words shown in uppercase letters), a variable (the word *name*), a delimiter (the left parenthesis), and an operator (the = character). Items can also include fragment references and separators. These items need to be in groups.

IBMIDDoc provides elements and attributes to mark up the syntax diagram elements. Here is the markup we used for the sample diagram:

```
<syntax><title>SAA CPI Database Reference</title>
<group>
<kwd>ERASE</kwd>
</group>
<group choiceseq="CHOICE">
<kwd>FORM</kwd>
<kwd>PROC</kwd>
<kwd>QUERY</kwd>
<kwd>TABLE</kwd>
</group>
<group>
<var>name</var>
</group>
<group optreq="OPT" style="BKM:(COMPOSITE)">
<delim>(</delim>
<kwd>CONFIRM</kwd>
<oper>=</oper>
<group choiceseq="CHOICE">
<kwd>YES</kwd>
<kwd>NO</kwd>
</group>
</group>
</syntax>
```

IBMIDDoc also provides elements and attributes for elements not illustrated in the sample diagram.

Fragments

A part of a syntax diagram, separated from the diagram to show greater detail. Like a syntax diagram, a fragment can contain items and groups. We do not mean to imply that the main syntax diagram is always complete. Often a main syntax diagram shows only a part of the syntax of the whole program. The word “fragment,” as used here, means a part of your main diagram or of another fragment.

Syntax notes (SynNote)

Notes often placed at the bottom of the diagram. Syntax notes are similar to footnotes placed in text.

RepSep

Defines a repeat separator in a syntax diagram.

SynBlk

Organizes syntax definitions into subdivisions and keeps them together on a line.

SynPh

Contains syntax elements, and is usually used to show a portion of a syntax definition.

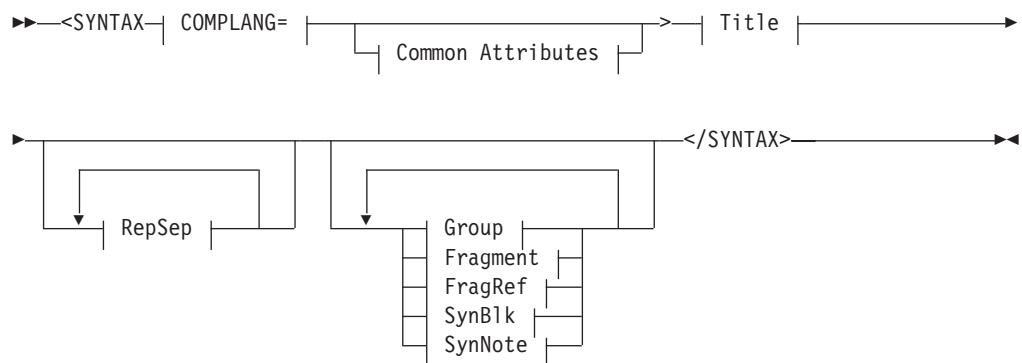
The Syntax element

The Syntax element contains the syntax diagram markup. The attributes of the Syntax element define the characteristics of the diagram.

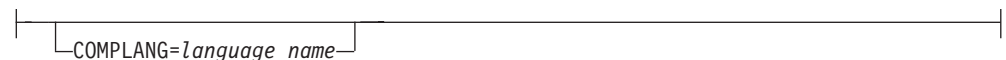
The text of the syntax diagram title can be contained in a Title element within the Syntax element. For example, we included a title, "SAA CPI Database Reference," within the Title element on our previous sample syntax diagram.

The following diagram shows the format for the Syntax element itself:

SYNTAX



COMPLANG=



Special style settings to control how the diagram formats.

- To have large diagrams flow from page to page in BookMaster, you need to use this override on the style attribute. XyVision uses this type of style by default.

```
<syntax style="bkm:(style=space split=yes)">
```

- For wide diagrams being output for BookManager BOOKs, specify a DWIDTH (the default value is 74):

```
<syntax style="bkm:(style=space split=yes dwidth=100)">
```

- If you have a lot of syntax diagrams and are formatting using BookMaster, you may want the "style=space split=yes" setting. Coding the PROPDEF causes all your syntax diagrams to have this style.

```
<propdef eletypes="syntax" style="bkm:(style=space split=yes)">
</propdef>
```

You can have a box around your diagram, rules above and below it, or a labeled box around your diagram. Use the SynStyle attribute to add these style effects. The default is a space (SynStyle=Space).

SynStyle=LblBox

Causes a box to be placed around the diagram. The top line of the box has text label that is taken from the diagram's Title tag.

SynStyle=Box

Causes a box to be placed around the diagram.

SynStyle=Rule

Causes a line to be placed above and below the diagram; to visually separate it from the surrounding text.

The Group element

The Group element defines the syntax group and lets you give the group a name in a Title element. The Title element enables the Group to be automatically fragmented if it is too large to fit the current area. All items in a sequential group are kept on the same line. If you have several items that are too wide for one line, you'll have to split them into separate groups.

Each of the following examples shows a group with two keywords.

- In the first example the group is required and sequential:

▶▶—FORM—PROC————▶▶

Here's its markup:

```
<syntax>
<group>
<kwd>FORM</kwd>
<kwd>PROC</kwd>
</group>
</syntax>
```

- In the second example the group is sequential and optional (optreq attribute):

▶▶└─FORM—PROC─┐————▶▶

Here's its markup:

```
<syntax>
<group optreq="opt">
<kwd>FORM</kwd>
<kwd>PROC</kwd>
</group>
</syntax>
```

- In the next example, the group is a choice of the two keywords (choiceseq attribute); one is required:

▶▶└─FORM
└─PROC─┐————▶▶

Here's its markup:

$$\begin{matrix} + \\ + \\ + \\ + \\ + \\ + \end{matrix}$$

- +
- +



+

+

$$\begin{matrix} + \\ + \\ + \\ + \\ + \end{matrix}$$

- +
- +



+

+

+
+
+
+
+
+

- +
- +



+

+

+

+
+
+
+
+
+
+
+
+
+
+

```

+      </group>
+      <group>
+      <kwd>SOME LARGE KEYWORD TO GET THE DIAGRAM TO BREAK AND FLOW</kwd>
+      </group>
+      </syntax>

```

The KWD (keyword) element

The KWD element describes a keyword, which is a command name or any other literal information.

Examples:

- In this example, a group element contains two keywords, each contained in KWD elements:

```

+      ►►—FORM—PROC—►►

```

Here's its markup:

```

+      <syntax>
+      <group>
+      <kwd>FORM</kwd>
+      <kwd>PROC</kwd>
+      </group>
+      </syntax>

```

- In this example, the PROC keyword is optional:

```

+      ►►—FORM—┐
+                  └─PROC─┘

```

Here's its markup:

```

+      <syntax>
+      <group>
+      <kwd>FORM</kwd>
+      <kwd optreq="opt">PROC</kwd>
+      </group>
+      </syntax>

```

- In this example, the PROC keyword is a default:

```

+      ►►—FORM—┐
+                  └─PROC─┘

```

Here's its markup:

```

+      <syntax>
+      <group>
+      <kwd>FORM</kwd>
+      <kwd optreq="def">PROC</kwd>
+      </group>
+      </syntax>

```

The VAR (variable) element

The VAR element describes any variable information.

In this example, the VAR element contains the text language_name.

+ ▶▶—LANGUAGE—==—*language_name*————▶▶

+
+ Here's its markup:
+ <syntax>
+ <group>
+ <kwd>LANGUAGE</kwd>
+ <oper>==</oper>
+ <var>language_name</var>
+ </group>
+ </syntax>

+ The OPER (operator) element

+ The OPER element describes an operator. Operators include add (+), subtract (-),
+ multiply (*), divide (/), equal (=), and other mathematical operators. The operator
+ can consist of more than one character.

+ In this example, the OPER element contains an equals (=) sign.

+ ▶▶—LANGUAGE—==—*language_name*————▶▶

+
+ Here's its markup:
+ <syntax>
+ <group>
+ <kwd>LANGUAGE</kwd>
+ <oper>==</oper>
+ <var>language_name</var>
+ </group>
+ </syntax>

+ The SEP (separator) element

+ The SEP element describes a separator that is to separate keywords, variables,
+ operators, or groups. The separator can be more than one character.

+ ▶▶—FRED—, —BARNEY————▶▶

+
+ Here's its markup:
+ <syntax>
+ <group>
+ <kwd>FRED</kwd>
+ <sep>,</sep>
+ <kwd>BARNEY</kwd>
+ </group>
+ </syntax>

+ The Delim (delimiter) element

+ The Delim element specifies a delimiter that is to indicate the start or end of
+ keywords, variables, operators, or groups. The delimiter can be one or more
+ characters.

+ Examples:

- + • In this example, the delimiter is a plus (+) sign:

+ >>—FRED—+—WILMA—<<

+
+ Here's its markup:

+ <syntax>
+ <group>
+ <kwd>FRED</kwd>
+ <delim>+</delim>
+ <kwd>WILMA</kwd>
+ </group>
+ </syntax>

- + • You can use the STARTEND attribute to ensure that delimiters are specified in
+ matched sets. If the syntax diagram requires a single delimiter, do not use
+ STARTEND.

+ >>—ID(identifier)—<<

+
+ Here's its markup:

+ <syntax>
+ <group style="bkm:(composite)">
+ <kwd>ID</kwd>
+ <delim startend="start"></delim>
+ <kwd>identifier</kwd>
+ <delim startend="end"></delim>
+ </group>
+ </syntax>

+ The RepSep (repeat separator) element

+ The RepSep element specifies whether the group of items or groups can repeat,
+ and also the repeat separator character, if one is to be used. If the repeat separator
+ character is specified, it separates the repeated group of items or groups in the
+ syntax diagram.

+ The RepSep element must have an ID value. This ID is used when referencing the
+ RepSep element from within the syntax markup. Use the REPID attribute on the
+ repeating group; it references the ID on a RepSep element.

+ >>—<repsep-id=identifier—
+ 

+ >| common attributes |<<

+
+ Examples:

- + • The following example shows a group containing a variable you can repeat;
+ there is no repeat separator character.

+
+ >>——<<

+
+ Here's its markup:

```

<syntax>
<repsep id="rsep0003a"></repsep>
<group repid="rsep0003a">
<var>variable</var>
</group>
</syntax>

```

- The following example shows a group containing a variable and a repeat separator character. In this example, the repeat separator character is required:



Here's its markup:

```

<syntax>
<repsep id="rsep0003">,</repsep>
<group repid="rsep0003">
<var>variable</var>
</group>
</syntax>

```

- The following example shows a group containing a variable and a repeat separator character. Here the repeat separator character is optional.



Here's its markup:

```

<syntax>
<repsep optreq="OPT" id="rsep0004">,</repsep>
<group repid="rsep0004">
<var>variable</var>
</group>
</syntax>

```

The FRAGMENT and FRAGREF (fragment reference) element

A syntax diagram can contain a section that has too many items or groups to fit in the diagram, or it can contain a section that is used more than once. You can present such a section as a separate fragment. You give the fragment a name that corresponds to the name of the section in the main diagram represented by the fragment.

The Fragment element specifies a fragment of your main syntax diagram or another fragment. The Fragment element is similar to the Syntax element. You can use Kwd, Var, Oper, Delim, Sep, FragRef, Group, and SynNote. These elements let you specify a diagram fragment in the same way that you specify a main diagram.

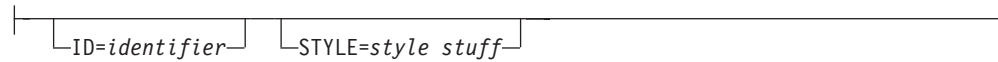
You can specify as many fragments as you want for a main diagram. The Fragment elements cannot be placed inside a Group element. Fragment is valid only within Syntax and SynBlk elements.

The FRAGREF element describes a reference to a syntax diagram fragment. The text of the FRAGREF element is placed in the syntax diagram and must match the name of the fragment reference that it refers to.

Example:



Common attributes:



Here's its markup:

```
<syntax>
<fragref><title>Common attributes</title></fragref>
<fragment><title>Common attributes</title>
<group optreq="opt" style="bkm:(composite)">
<kwd>ID</kwd>
<oper>=</oper>
<var>identifier</var>
</group>
<group optreq="opt" style="bkm:(composite)">
<kwd>STYLE</kwd>
<oper>=</oper>
<var>style stuff</var>
</group>
</fragment>
</syntax>
```

Syntax Notes

IBMIDDoc provides the SynNote element for placing notes in your syntax diagrams. Syntax notes are similar to footnotes in regular text. At processing time, a number or other callout is placed next to an item, group, or fragment in the diagram, indicating that a note is associated with that part of the diagram, and the note appears at the bottom of the diagram, after any fragments.

Examples:

- This shows a simple note:



Notes:

1 This is a rather common name.

Here's its markup:

```
<syntax>
<group>
<kwd>FRED</kwd>
<synnote>This is a rather common name.</synnote>
</group>
</syntax>
```

- You can use the CALLOUT attribute in hardcopy to have a specific character displayed for the note item. This diagram is brought to you by the letter "N":

+ (N)
+ >>—FRED—<<

+

+ **Notes:**

+ **N** This is a rather common name.

+ Here's its markup:

+ <syntax>
+ <group>
+ <kwd>FRED</kwd>
+ <synnote callout="N">This is a rather common name.
+ </synnote>
+ </group>
+ </syntax>

+ • You can also specify a note once, then refer to it more than once:

+ (1) (1)
+ >>—FRED—BARNEY—<<

+

+ **Notes:**

+ **1** This is a rather common name.


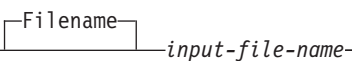
+ Here's its markup:

+ <syntax>
+ <synnote id="comname">This is a rather common name.
+ </synnote>
+ <group>
+ <kwd>FRED</kwd>
+ <synnote refid="comname">
+ </group>
+ <group>
+ <kwd>BARNEY</kwd>
+ <synnote refid="comname">
+ </group>
+ </syntax>

+

+ Syntax Phrases

+ Syntax phrases allow you to use a portion of a syntax statement; such as a term in
+ a parameter list. For example:

+ 
+ >>——<<

+

+ **Filename**

+ Sample description for this syntax item.

+ This is the markup:

+ <syntax>
+ <group>
+ <kwd optreq="def">Filename</kwd>
+ <var>input-file-name</var>
+ </group>
+ </syntax><parml>

```
<parm><term><synph><kwd optreq="def">Filename</kwd></synph></term>
<defn>Sample description for this syntax item.</defn>
</parm>
</parm1>
```

+ Examples of Syntax Definitions and Markup

The examples in the sections that follow represent typical syntax definitions.

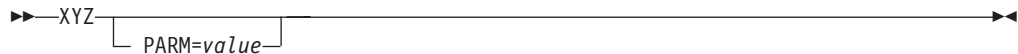
Example 1: A simple syntax definition

This example illustrates one of the simplest styles of syntax definition, with only one optional parameter value.

```
<SYNTAX>
<TITLE>XYZ Command</TITLE>
<GROUP>
  <TITLE>CMD</TITLE>
  <KWD>XYZ</KWD>
  <GROUP OPTREQ="OPT">
    <TITLE>OPTION 1</TITLE>
    <SEP>&ssbl</SEP>
    <KWD>PARAM</KWD>
    <OPER>&equals</OPER>
    <VAR>value</VAR>
  </GROUP>
</GROUP>
</SYNTAX>
```

This SGML input will produce the following output.

XYZ Command



Example 2: A simple syntax definition that repeats

This example illustrates a syntax definition for a command with a parameter that can be repeated.

Syntax Diagram With Repetition



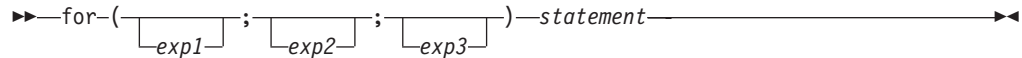
Here's its markup:

```
<syntax><title>Syntax Diagram With Repetition</title>
<repsep id="REP1">,</repsep>
<group>
<kwd>command</kwd>
</group>
<group repid="REP1" optreq="OPT" style="bkm:(composite)">
<kwd>parm</kwd>
<oper>=</oper>
<var>value</var>
</group>
</syntax>
```

Example 3: A more complex syntax definition

The following syntax definition contains a single group comprising three variable expressions, two separators, and two delimiters. Each variable expression is optional. The definition also includes a required keyword and a required variable statement.

SAA CPI C Reference



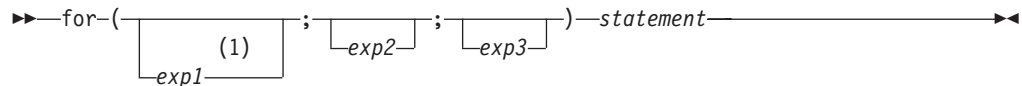
Here's its markup:

```
<syntax><title>SAA CPI C Reference</title>
<group style="BKM:(COMPOSITE)">
  <kwd>for</kwd>
  <group style="BKM:(COMPOSITE)">
    <delim optreq="req" startend="START">(</delim>
    <var optreq="OPT">exp1</var>
    <sep optreq="req">&semi;</sep>
    <var optreq="OPT">exp2</var>
    <sep optreq="req">&semi;</sep>
    <var optreq="OPT">exp3</var>
    <delim optreq="req" startend="END">)</delim>
  </group>
  <var>statement</var>
</group>
</syntax>
```

Example 4: A variation on Example 3

This is the same diagram as in "Example 3: A more complex syntax definition", with a syntax note added.

SAA CPI C Reference



Notes:

- 1 This indicates the beginning condition.

Here's its markup:

```
<syntax><title>SAA CPI C Reference</title>
<group style="BKM:(COMPOSITE)">
  <kwd>for</kwd>
  <group style="BKM:(COMPOSITE)">
    <delim optreq="req" startend="START">(</delim>
    <var optreq="OPT">exp1</var>
    <synnote>This indicates the beginning condition.</synnote>
    <sep optreq="req">&semi;</sep>
    <var optreq="OPT">exp2</var>
    <sep optreq="req">&semi;</sep>
    <var optreq="OPT">exp3</var>
    <delim optreq="req" startend="END">)</delim>
  </group>
  <var>statement</var>
</group>
</syntax>
```

```

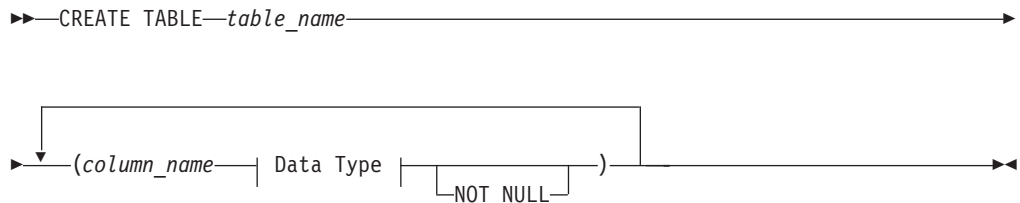
+      </group>
+      <var>statement</var>
+      </group>
+      </syntax>

```

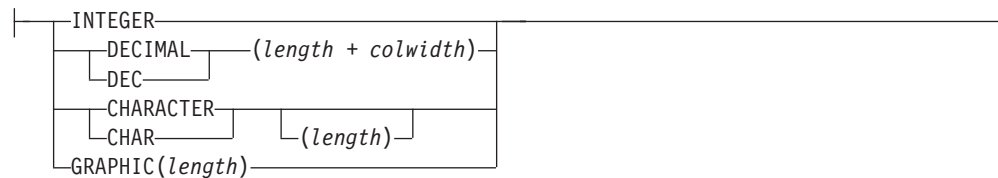
Example 5: A syntax definition showing a fragment and significant blanks

The following syntax definition includes a fragment called “Data Type.” The fragment is placed below the main syntax definition. This example also shows the use of the syntax significant blank symbol (&ssbl.); use this to ensure a blank is left in the diagram where the user should code a space.

Database Reference



Data Type:



Here's its markup:

```

+      <syntax><title>Database Reference</title>
+      <repsep id="rsep0006"></repsep>
+      <group>
+      <kwd>CREATE TABLE</kwd>
+      </group>
+      <group>
+      <var>table_name</var>
+      </group>
+      <group repid="rsep0006">
+      <group style="bkm:(composite)">
+      <delim startend="START"></delim>
+      <var>column_name</var>
+      </group>
+      <fragref><title>Data Type</title></fragref>
+      <kwd optreq="OPT">NOT NULL</kwd>
+      <delim optreq="req" startend="END"></delim>
+      </group>
+      <fragment><title>Data Type</title>
+      <group choiceseq="CHOICE">
+      <kwd>INTEGER</kwd>
+      <group>
+      <group choiceseq="CHOICE">
+      <kwd>DECIMAL</kwd>
+      <kwd>DEC</kwd>
+      </group>
+      </group>
+      <group style="BKM:(COMPOSITE)">

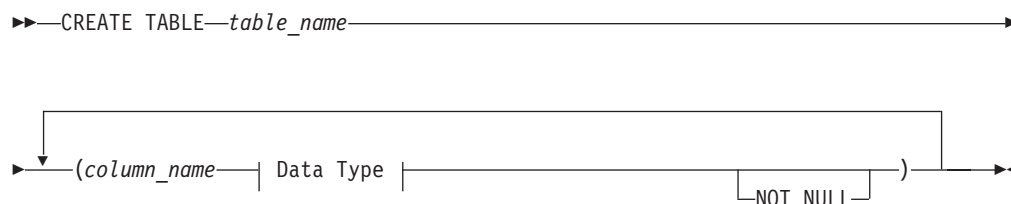
```

```
+      <delim startend="START"></delim>  
+      <var>length</var>  
+      <sep>&ssbl;+&ssbl;</sep>  
+      <var>colwidth</var>  
+      <delim startend="END"></delim>  
+    </group>  
+  </group>  
+  <group>  
+    <group choiceseq="CHOICE">  
+      <kwd>CHARACTER</kwd>  
+      <kwd>CHAR</kwd>  
+    </group>  
+    <group optreq="OPT" style="BKM:(COMPOSITE)">  
+      <delim startend="START"></delim>  
+      <var>length</var>  
+      <delim startend="END"></delim>  
+    </group>  
+  </group>  
+  <group style="BKM:(COMPOSITE)">  
+    <kwd>GRAPHIC</kwd>  
+    <delim startend="START"></delim>  
+    <var>length</var>  
+    <delim startend="END"></delim>  
+  </group>  
+ </fragment>  
+ </syntax>
```

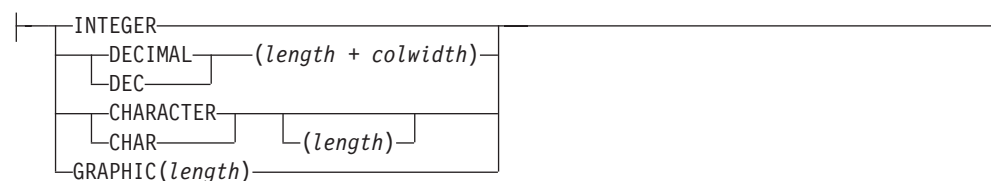
Example 6: A syntax definition with automatic fragmenting

The following example is identical to “Example 5: A syntax definition showing a fragment and significant blanks” on page 138 except that the fragment is marked up as a group with a title. Because the group is very wide, it automatically fragments.

Database Reference



Data Type



Here's its markup:

```
<syntax><title>Database Reference</title>
<repsep id="rsep0006"></repsep>
<group>
<kwd>CREATE TABLE</kwd>
```

```

+      </group>
+      <group>
+      <var>table_name</var>
+      </group>
+      <group repid="rsep0006">
+      <group style="bkm:(composite)">
+      <delim startend="START">(</delim>
+      <var>column_name</var>
+      </group>
+      <group choiceseq="CHOICE"><title>Data Type</title>
+      <kwd>INTEGER</kwd>
+      <group>
+      <group choiceseq="CHOICE">
+      <kwd>DECIMAL</kwd>
+      <kwd>DEC</kwd>
+      </group>
+      <group style="BKM:(COMPOSITE)">
+      <delim startend="START">(</delim>
+      <var>length</var>
+      <sep>&ssbl;+&ssbl;</sep>
+      <var>colwidth</var>
+      <delim startend="END">)</delim>
+      </group>
+      </group>
+      <group>
+      <group choiceseq="CHOICE">
+      <kwd>CHARACTER</kwd>
+      <kwd>CHAR</kwd>
+      </group>
+      <group optreq="OPT" style="BKM:(COMPOSITE)">
+      <delim startend="START">(</delim>
+      <var>length</var>
+      <delim startend="END">)</delim>
+      </group>
+      </group>
+      <group style="BKM:(COMPOSITE)">
+      <kwd>GRAPHIC</kwd>
+      <delim startend="START">(</delim>
+      <var>length</var>
+      <delim startend="END">)</delim>
+      </group>
+      </group>
+      <group>
+      <kwd optreq="OPT">NOT NULL</kwd>
+      <delim optreq="req" startend="END">)</delim>
+      </group>
+      </group>
+      </syntax>

```

Chapter 15. Developing Programming Language Reference Materials

We are often called upon to produce reference information for the various elements of programming languages, either as the major portion of a language reference manual, or as part of a combined language guide and reference. In this context we use the term “programming language” very broadly, including the higher-level languages (such as Java), command and control languages (such as JCL, TSO, and CMS commands), macro languages, such as Access Method Services macros, and markup languages (such as our very own IBMIDDoc).

These reference materials typically have a couple of things in common:

- The same set of subtopics (format, parameters, usage, and so on) is repeated for each language element (statement, macro, and so on). While each subtopic may have its own heading, you don’t want these headings to appear in your table of contents.
- Consistency of presentation and retrievability are critical, as readers want to find the information as quickly as possible and not have to re-interpret the presentation each time.

Another good reason to use IBMIDDoc elements for language element reference materials is that once you’ve determined how you’re going to use the elements for your particular language reference material, you’ll find they are a help in developing consistent, well-structured materials that are easier to maintain.

The Structure of a Language Element Reference Section

Use LERS to contain reference information for computer languages and command information. LERS contains one or more Language Elements (LEs) that contain the description of a computer language element such as commands, and description items, such as format, purpose, and examples. There are several elements used to complete the LERS section.

A typical LERS section looks like:

```
language element reference section <LERS>
  language element <LE>
    language element name <LEN>
    language element description <LEDesc>
    language element description item <LEDI>
  :
  language element <LE>
    language element name <LEN>
    language element description <LEDesc>
    language element description item <LEDI>
  :
end language element reference section </LERS>
```

The language element reference section (LERS) contains many language elements (described with the LE, LEN, and LEDESC tags), and for each of the language elements you can have lots of description items (LEDI tags). By description items,

we mean such things as format (sometimes called “syntax”), purpose, examples — those categories of information we typically provide when describing a language element.

Describing Your Reference Section

Your first task in creating a language element reference section (after you figure how you’re going to present the material, of course) is to describe it, which you do with attributes on either the LERS (language element reference section) element or the LERSDEF (LERS definition) element.

While a book typically might have only a few, or perhaps just one language element reference section, these sections can be enormously long. It is impractical and inefficient to handle these long sections in a single source file; you will want to break up the material into multiple files, each with its own LERS element, so that each file can be processed independently. This is why we have the LERSDEF element. It allows you to specify all of the LERS attributes — you put the LERSDEF in your Prolog, and it is referenced in your document.

What can you describe on the LERS or LERSDEF element about your reference section?

- **How to get at the description**

The LERSDEF element has a CLASSNAME attribute so you can refer to it using the CLASS attribute on the LERS element.

- **What text you want generated for the description items**

Each language element description item element (LEDI) has an attribute that defines the category of that description item. IBMIDDoc will generate a subheading for each description item, as determined by the attribute.

For each category, IBMIDDoc has a corresponding attribute on the LERS and LERSDEF elements that allows you to specify the subheading text you want in place of the text generated by default. You can specify any text you want, or you can specify that *no heading be generated at all*.

Table 15 shows the attribute name for each of the language element description item categories, the text that will be generated by default in styles that use IBMIDDoc’s initial setting, and what the category means. Of course, you will only use the categories that are appropriate for your material.

Table 15. Categories of language element description items

Attribute name	Generated heading	Description
AUTH	Authorization	the authorization level necessary to use this language element
COMMENTS	Comments	just about anything you consider comments
CONTEXT	Context	the context in which this language element is valid
DEFAULTS	Defaults	the defaults
ERRCOND	Error Conditions	error conditions that can arise from misuse
EXAMPLES	Examples	examples of input and output
FLAGS	Flags	the flags that could be set by the language element

Table 15. Categories of language element description items (continued)

Attribute name	Generated heading	Description
FORMAT	Format	the general format (or syntax)
INTREP	Internal Representation	the internal representation (for example, binary) of the language element (sometimes called “encoding”)
MESSAGES	Messages	messages that can be generated as a result of use of this element
OTHER		a build-your-own category
PARMS	Parameters	the parameters of the language element
PROCESS	Processing	the processing that will be done for the language element (that is, the logic)
PURPOSE	Purpose	the purpose of the language element
RESTRICT	Restrictions	restrictions on use of the language element
RESULTS	Return Codes	explanations of the return codes possible with the language element
SYSENV	System Environment	the system environment in which the language element is valid
USAGE	Usage	how the language element is used
VERSION	Version	the version of the program in which the language element is valid

The categories have been selected based on a review of what has been used in the past. The OTHER category exists to allow you to create a category that has not been anticipated in the list above; before you decide to use OTHER to create a new category, review the ones available carefully to make sure that your category doesn’t already exist.

Be assured that IBMIDDoc is not demanding that you structure your reference information to match these categories. You might very well want to deal with the defaults as part of the discussion of the parameters, rather than as a separate “Defaults” category; similarly, you might want to deal with “Context” in the discussion of usage. These decisions depend on the nature of the language you are describing and the approach you take to its presentation.

So you might, for example, decide the following:

- Your reference section should have the categories PURPOSE, FORMAT, PARMS, USAGE, and EXAMPLES
- You do want the FORMAT category headed “Syntax”
- You do want the PARMS category headed “Attributes and Contained Elements”

Your LERSDEF might look like this:

```
<LERSDEF
  FORMAT="Syntax"
  PARMS="Attributes and Contained Elements"
  CLASSNAME="UGREFLERS">
  <DESC>Contains the LEDI IBMIDDoc User's Guide
    and Reference LERS name specifications.</DESC>
</LERSDEF>
```

You didn't have to say anything about PURPOSE or EXAMPLES because you want the default headings generated.

Describing the language element

Following the language element name, you have a series of language element description items, marked up with the LEDI element. Each LEDI element CLASS attribute needs to have an attribute that describes the category of the item:

AUTH	EXAMPLES	OTHER	RESULTS
COMMENTS	FLAGS	PARMS	RET_CODES
CONTEXT	FORMAT	PROCESS	SYSENV
DEFAULTS	INTREP	PURPOSE	USAGE
ERRCOND	MESSAGES	RESTRICT	VERSION

These are the same description item category attributes that occur on the LERS and LERSDEF element (see Table 15 on page 142).

Normally, each language element name begins on a new page. To suppress these new pages, use the following processing instruction in the prolog of your document:

```
<?xpp:lrs nopage>
```

In Adept, insert a processing instruction, then modify the PI and insert this:
xpp:lrs nopage

Example of a Simple Language Element Reference Section

Time for an example. In this example, we use these categories:

PURPOSE

Without a heading. Because it follows the LEN immediately, it doesn't need its own heading.

FORMAT

Uses the special heading text "Syntax".

PARMS

Without a heading. It appears to the reader as simply part of the syntax discussion.

USAGE

Uses the default heading text.

RESTRICT

Uses the special heading text "Do's and Don'ts".

EXAMPLES

Uses the default heading text.

MESSAGES

Uses the default heading text. We only have one command that yields messages, so this is used only once.

Our language in the following example is a command language for a culinary robot. Here's its coding:

```
<lrsdef format="Syntax" parms="" purpose="" restrict="Do's and Don'ts"
classname="culinary"></lrsdef>
...
<lrs class="culinary"><le>
<len>DISHDEF
```

```

defining a dish</len>
<ledi class="purpose">
  <p>The DISHDEF command defines a dish – the ingredients
  that it contains and the processing steps to prepare
  it.</p>
</ledi>
<ledi class="format">
  <syntax>
    <repsep id="cul"></repsep>
    <group>
      <kwd>DISHDEF</kwd>
    </group>
    <group style="bkm:(composite)">
      <kwd>NAME</kwd>
      <delim>=</delim>
      <var>name-of-dish</var>
    </group>
    <group repid="cul" style="bkm:(composite)">
      <kwd>INGREDIENT</kwd>
      <delim>=</delim>
      <var>ingredient-name</var>
      <delim>/</delim>
      <var>quantity</var>
    </group>
    <group repid="cul" style="bkm:(composite)">
      <kwd>STEP</kwd>
      <delim>=</delim>
      <var>process-name</var>
      <delim> (</delim>
      <var>ingredient-list</var>
      <delim>)</delim>
    </group>
    <group choiceseq="choice">
      <group>
        <kwd>UNTIL</kwd>
        <var>condition-name</var>
      </group>
      <group>
        <kwd>FOR</kwd>
        <var>time</var>
      </group>
    </group>
  </syntax>
</ledi>
<ledi class="parms">
  <parml>
    <parm><term><synph><kwd>NAME</kwd><delim>=</delim><var>
name-of-dish</var></synph></term>
<defn>identifies the dish name.</defn>
</parm>
    <parm><term><synph><kwd>INGREDIENT</kwd><delim>=</delim><var>
ingredient-name</var><delim>/</delim><var>quantity
</var></synph></term>
<defn>identifies an ingredient and the quantity per
serving. The ingredient must be expressed in international
culinary ingredient units (ICIUs). The quantity must
be expressed in international culinary quantity units
(ICQUs). This parameter is repeated as often as necessary
to define each of the ingredients in the dish.</defn>
</parm>
    <parm><term><synph><kwd>STEP</kwd><delim>=</delim><var>
process-name</var><delim>(</delim><var>ingredient-list
</var><delim>)</delim></synph></term>
<defn>identifies a preparation step and the ingredients
to use. Process names must be expressed in international
culinary step units (ICSUs). This clause is repeated
as often as necessary to define each of the preparation

```

```

steps for the dish.</defn>
</parm>
<parm><term><synph><kwd>UNTIL</kwd><var>condition-name
</var></synph></term><term><synph><kwd>FOR</kwd><var>
time</var></synph></term>
<defn>identifies a condition under which the step
is to conclude or an amount of time for processing.
Condition names must be expressed in international
culinary condition units (ICCU). </defn>
</parm>
</parml>
</ledi>
<ledi class="usage">
<p>Use the DISHDEF command to specify to the culinary
robot how to prepare a dish. </p>
</ledi>
<ledi class="restrict">
<p>Make sure that the ingredients list is in the order
in which the ingredients are to be used. For example,
for a SAUTE step, make sure that BUTTER is specified
before VEAL or the robot will put the veal in the
pan before the butter.</p>
</ledi>
<ledi class="examples">
<xmp>dishdef name=vealalfred
ingredient=butter/1tsp
ingredient=vealscallop/6oz ingredient=salt/pinch
ingredient=tarragon/1tsp ingredient=sourcream/halfcup
step=saute (butter vealscallop) until golden brown
step=add (salt tarragon) for 1 min
step=deglaze (sourcream) for 6 min</xmp>
</ledi>
</le>
<len>EVALUATE
evaluate nutrition, cost, or preparation time</len>
<ledi class="purpose">
<p>Once you have a menu defined, use the EVALUATE
command to determine its nutritional characteristics
and the preparation time. If you have access to the
Daily Market Cost data base, you can also evaluate
the cost of a shopping list containing one or more
menus.</p>
</ledi>
<ledi class="format">
<syntax>
<group>
<kwd>EVALUATE</kwd>
</group>
<group choiceseq="choice">
<group>
<kwd>NUTRITION</kwd>
<group style="bkm:(composite)">
<kwd>MENU</kwd>
<delim>=</delim>
<var>menu-name</var>
</group>
</group>
<group>
<kwd>COST</kwd>
<group style="bkm:(composite)">
<kwd>SHOPLIST</kwd>
<delim>=</delim>
<var>shopping-list-name</var>
</group>
</group>
</group>

```

```

<kwd>PREPTIME</kwd>
<group style="bkm:(composite)">
<kwd>MENU</kwd>
<delim>=</delim>
<var>menu-name</var>
</group>
<group style="bkm:(composite)">
<kwd>SERVING</kwd>
<delim>=</delim>
<var>number</var>
</group>
</group>
</syntax>
</ledi>
<ledi class="parms">
<parml>
<parm><term><synph><kwd>MENU</kwd><delim>=</delim><var>
menu-name</var></synph></term>
<defn>requests a nutritional evaluation of menu name.
</defn>
</parm>
<parm><term><synph><kwd>COST</kwd> <kwd>SHOPLIST</kwd><delim>
=</delim><var>shopping-list-name</var></synph></term>
<defn>as determined by your marketing profile and
the Daily Market Cost data base, this command generates
a cost for the named shopping list for each of the
markets in the profile, including the cost of the
gasoline for driving to those markets designated in
your profile as not providing delivery service.</defn>
</parm>
<parm><term><synph><kwd>PREPTIME</kwd> <kwd>MENU</kwd><delim>
=</delim><var>menu-name</var> <kwd>SERVING</kwd><delim>
=</delim><var>number</var></synph></term>
<defn>requests an evaluation of the preparation time
for the designated menu serving the designated number
of people.</defn>
</parm>
</parml>
</ledi>
<ledi class="usage">
<p>Use the EVALUATE command as required to maximize
the nutrition and minimize the cost of meals. Knowing
the preparation time is critical in requesting that
menus be prepared.</p>
</ledi>
<ledi class="examples">
<xmp>//evaluate nutrition menu=companydinner
//evaluate cost sholist=monday
//evaluate preptime menu=companydinner serving=8</xmp>
</ledi>
</le></lers>

```

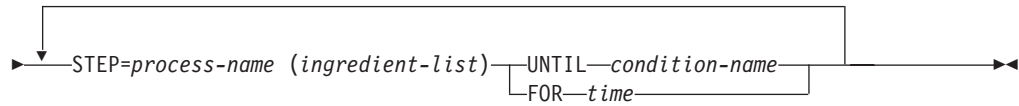
DISHDEF

defining a dish

The DISHDEF command defines a dish — the ingredients that it contains and the processing steps to prepare it.

Syntax





NAME=*name-of-dish*
identifies the dish name.

INGREDIENT=*ingredient-name/quantity*
identifies an ingredient and the quantity per serving. The ingredient must be expressed in international culinary ingredient units (ICIUs). The quantity must be expressed in international culinary quantity units (ICQUs). This parameter is repeated as often as necessary to define each of the ingredients in the dish.

STEP=*process-name(ingredient-list)*
identifies a preparation step and the ingredients to use. Process names must be expressed in international culinary step units (ICSUs). This clause is repeated as often as necessary to define each of the preparation steps for the dish.

UNTIL*condition-name*

FOR*time*
identifies a condition under which the step is to conclude or an amount of time for processing. Condition names must be expressed in international culinary condition units (ICCUUs).

Usage

Use the DISHDEF command to specify to the culinary robot how to prepare a dish.

Do's and Don'ts

Make sure that the ingredients list is in the order in which the ingredients are to be used. For example, for a SAUTE step, make sure that BUTTER is specified before VEAL or the robot will put the veal in the pan before the butter.

Examples

```

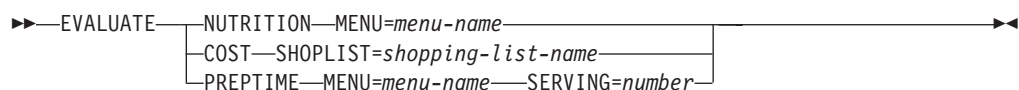
dishdef name=vealalfred
ingredient=butter/1tsp
ingredient=vealscallop/6oz ingredient=salt/pinch
ingredient=tarragon/1tsp ingredient=sourcream/halfcup
step=saute (butter vealscallop) until golden brown
step=add (salt tarragon) for 1 min
step=deglaze (sourcream) for 6 min
  
```

EVALUATE

evaluate nutrition, cost, or preparation time

Once you have a menu defined, use the EVALUATE command to determine its nutritional characteristics and the preparation time. If you have access to the Daily Market Cost data base, you can also evaluate the cost of a shopping list containing one or more menus.

Syntax



| **MENU**=*menu-name*

| requests a nutritional evaluation of menu name.

| **COST SHOPLIST**=*shopping-list-name*

| as determined by your marketing profile and the Daily Market Cost data base,
| this command generates a cost for the named shopping list for each of the
| markets in the profile, including the cost of the gasoline for driving to those
| markets designated in your profile as not providing delivery service.

| **PREPTIME MENU**=*menu-name* **SERVING**=*number*

| requests an evaluation of the preparation time for the designated menu serving
| the designated number of people.

| **Usage**

| Use the EVALUATE command as required to maximize the nutrition and minimize
| the cost of meals. Knowing the preparation time is critical in requesting that menus
| be prepared.

| **Examples**

| //evaluate nutrition menu=companydinner

| //evaluate cost sholist=monday

| //evaluate preptime menu=companydinner serving=8

Chapter 16. Defining Modular Information

Use modular information to describe information that is often repetitive in structure and content, and requires precise markup. To do this, you can define many different modular information classes for each type of information you need to describe. You can use modular information to create reference information for most anything. This chapter introduces ways to create modular information using IBMIDDoc.

You can create modular information classes that describe programming data structures, commands and syntax, or command definitions, for example. This information is often displayed in a tabular format. You can define this format using the modular information elements described in this section.

Tables are a presentation-oriented structure. Modular information attempts to capture the relationships that are often expressed in tables, using classes of information, in a more meaningful way. Many tables have aspects with specific meanings. There are times when, for presentation-specific purposes, a table must be altered or resized—to fit on a page, for example. These changes can cause confusion about the relationships the table is intended to illustrate.

Modular information elements capture these meanings using the class mechanism that is part of modular item specifications. IBMIDDoc's modular information elements are a way to express many of the structures that are currently expressed as tables. IBMIDDoc modular information can be expressed in many ways, without obscuring the meaning of the relationships expressed in the information's classes.

You can use several elements in combination to define modular information classes, descriptions, and properties. These elements include:

- ModInfo
- ModInfoDef
- Mod
- ModItem
- ModItemDef

The following example illustrates how to use these elements to define modular information. See the reference section entries for the elements used in the example that follows for more information about these elements.

```

:
:
<PROLOG>
:
:
<PROPDEFS>
:
:
<MODINFODEF CLASSNAME="payobj">
  <DESC>This class of modular information should be used to
  describe payroll objects on a data entry screen.
  </DESC>
  <MODITEMDEF CLASSNAME="exempt">Exemptions
    <DESC>Contains the number of exemptions the employee claims.
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="rate">Hourly Rate
    <DESC>Contains the amount, in dollars and cents, the employee is
    paid per hour.
  </MODITEMDEF>
:
:
```

```

      .
    </PROPDEFS>
      .
    </PROLOG>
      .

```

After defining the payobject, exemptions, and rate classes, they can be used as shown in the following example.

```

<MODINFO CLASS="payobj">
<MOD ID="empinf">
<MODNAME>Employee Pay Information</MODNAME>
<MODITEM CLASS="exempt">
<P>Enter the number of exemptions that the employee claims.</P></MODITEM>
<MODITEM CLASS="rate">
<P>Enter the employee's hourly rate of pay.</P></MODITEM>
</MOD>
      .
    </MODINFO>

```

Examples of Using Modular Information

Defining modular information allows the information to be displayed in a variety of ways, including a table presentation style. The example that follows contains modular information elements and a formatted example of one way to express the meaning of the modular information in a table presentation.

```

<PROPDEFS>
<MODINFODEF CLASSNAME="CUST">
<DESC>CUSTOMER INFORMATION</DESC>
  <MODITEMDEF CLASSNAME="NAME">
    <TITLE>NAME</TITLE>
    <DESC>THIS IS THE FIRST AND LAST NAME OF THE CUSTOMER</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="CUSTID">
    <TITLE>ID</TITLE>
    <DESC>THIS IS THEIR CUSTOMER ID NUMBER</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="INC">
    <TITLE>INCOME</TITLE>
    <DESC>THIS IS THEIR ANNUAL ESTIMATED INCOME</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="LPD">
    <TITLE>LAST PURCHASE DATE</TITLE>
    <DESC>THIS IS THEIR LAST PURCHASE DATE</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="NOTES">
    <TITLE>NOTES</TITLE>
    <DESC>PERSONAL NOTES</DESC>
  </MODITEMDEF>
</MODINFODEF>
</PROPDEFS>
      .
<MODINFO CLASS="CUST" STYLE="TABLE">
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">FRED SMITH</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1000</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><NUM BASE="10">40000</NUM></MODITEM>
    <MODITEM CLASS="LPDATE"><P><DATE>12/25/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>BIG SPENDER</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">SUZANNE STANLEY</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1001</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">50000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>11/22/92</DATE></P></MODITEM>
  </MOD>

```

```

    <MODITEM CLASS="NOTES"><P>LIKES GAME SOFTWARE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">JEFF GEORGE</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1002</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">60000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/02/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">MIKE GIDENTO</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1003</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">35000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/12/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
</MODINFO>

```

The resulting modular information is processed and the result will look similar to the example that follows.

Customer Name	ID	Income	Last Purchase Date	Notes
Fred Smith	1000	40000	12/25/93	Big spender
Suzanne Stanley	1001	50000	11/22/92	Likes game software
Jeff George	1002	60000	12/02/93	None
Mike GIdento	1003	35000	12/12/92	None

Chapter 17. File, text, and character entities and reusing information

Where multiple output documents contain common information, or a single document repeats the same information, the only way to ensure that the common information is the same is to take the information in each case from the same source file. At its simplest level, where the common information is in a few, largish blocks, the way to do this is to put each block of the common information in a file of its own and imbed it in the multiple output documents that use it.

IBMIDDoc allows you to reuse elements and information defined in entities; see "File and text entities". You can also reuse information from within the document, see "Reusing elements from an object library" on page 166.

File and text entities

Entities can be used to retrieve document fragments. An entity is any information that is referred to as a unit from a document. All entities are declared at the beginning of a document. Entities cannot be redefined within a document. Entity names are also case sensitive. Thus, product, PRODUCT, and Product refer to different entities.

There are two different types of entities for holding reusable information: file entities and text entities.

File entities

These are sometimes referred to as external entities or imbeds. These are files of markup and text to include in the document. They can be as large as a chapter (or even larger), or as small as a word (though this will drive translation centers nuts). The declaration points at the file. We recommended that these entities contain complete elements.

Text entities

These are sometimes referred to as symbols or internal entities. These entity declarations include the replacement text. A text entity can specify up to 2400 characters of information and markup.

To include a text or file entity, use an entity reference. An entity reference requests the entity data to be processed at the place where the reference occurs. Any entity can be referenced in this way, but they must be valid in the context in which they are referenced.

The following example shows how text and file entities are defined and used. The entities product, PRODUCT, and Prod are all text entities; the preface is a file entity.

```
<!ENTITY product "ABC Pgm.">
<!ENTITY PRODUCT "PQR Component.">
<!ENTITY Prod "XYZ Pgm">
<!ENTITY preface SYSTEM "xyz10pre.ide">
...
This book teaches you how to use &product;.
...
&preface;
```

```

...
<D>Using the &PRODUCT; of the &product;
...
<D>Using &product; with &Prod;

```

Special characters

Sometimes you need to specify characters that can be printed on the printer but cannot be typed at your keyboard. An example of that would be the bullet, which looks like this: •

No matter how hard you look, you can't find a key on your keyboard with one of those on it (unless you have a special keyboard). All symbols are entered the same way: an ampersand (&), followed by the symbol name, followed by a semicolon. So our symbol for the bullet, which is named "bul", would look like this:

```
&bul;
```

Table 16 shows the character entities defined in IBMIDDoc.

Table 16. IBMIDDoc Character Entities

Symbol	Appearance	Description
aa	á	a acute
Aa	Á	A acute
ac	â	a circumflex
Ac	Â	A circumflex
acute	'	accent acute
ae	ä	a umlaut
Ae	Ä	A umlaut
aelig	æ	ae ligature
AElig	Æ	AE ligature
ag	à	a grave
Ag	À	A grave
aleph	ℵ	aleph
all	∀	all
alpha	α	alpha
Alpha	Α	Alpha
amp	&	ampersand
and	^	and symbol
angle	∠	angle
angstrom	Å	angstrom
ao	â	a overcircle
Ao	Â	A overcircle
apos	'	apostrophe
app	≈	approximately
approx	≈	approximately
approxid	≈	approximately identical
arc	↷	arc
asterisk	*	asterisk
at	ã	a tilde
At	Ã	A tilde
atsign	@	at sign
aus	^a	underscored a
ballot	□	ballot box
because	∴	because
beta	β	beta
Beta	Β	Beta

Table 16. IBMIDDoc Character Entities (continued)










Symbol	Appearance	Description
bin	B'	binary
blank	b	blank (b with slash)
box		ballot box
BOX		solid box
BOXBOT		solid box bottom half
BOXLEFT		solid box left half
BOXRIGHT		solid box right half
BOXTOP		solid box top half
box12		shaded box 1/2 dots
box14		shaded box 1/4 dots
box34		shaded box 3/4 dots
bs		backspace
bsl	\	back slash
bslash	\	back slash
bul	•	bullet
bullet	•	bullet
bxas	⊥	box ascender
bxbj	⊥	box ascender
bxcj	⊕	box cross
bxcr	⊕	box cross
bxde	⊥	box descender
bxh	—	box horizontal
bxle	└	box left junction
bxlj	└	box left junction
bxll	└	box lower-left
bxlr	└	box lower-right
bxri	└	box right junction
bxrj	└	box right junction
bxtj	⊥	box descender
bxul	└	box upper-left
bxur	└	box upper-right
bxv		box vertical
bx0012	⌘	ASCII code 184
bx0021	⌘	ASCII code 183
bx0022	⌘	ASCII code 187
bx0120	⌘	ASCII code 214
bx0121	⌘	ASCII code 210
bx0202	=	ASCII code 205
bx0210	F	ASCII code 213
bx0212	⌘	ASCII code 209
bx0220	⌘	ASCII code 201
bx0222	⌘	ASCII code 203
bx1002	⌘	ASCII code 190
bx1012	⌘	ASCII code 181
bx1200	⌘	ASCII code 212
bx1202	⌘	ASCII code 207
bx1210	⌘	ASCII code 198
bx1212	⌘	ASCII code 216
bx2001	⌘	ASCII code 189
bx2002	⌘	ASCII code 188
bx2020	⌘	ASCII code 186
bx2021	⌘	ASCII code 182
bx2022	⌘	ASCII code 185

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
bx2100	␣	ASCII code 211
bx2101	␣	ASCII code 208
bx2120	␣	ASCII code 199
bx2121	␣	ASCII code 215
bx2200	␣	ASCII code 200
bx2202	␣	ASCII code 202
bx2220	␣	ASCII code 204
bx2222	␣	ASCII code 206
caret	^	caret
cc	ç	c cedilla
Cc	Ç	C cedilla
cdot	⊙	circled dot
cdq	”	close double quote
cdqf	»	French close double quote
cdqg	”	German close double quote
cedilla	ç	cedilla
cent	¢	cent
cequal	⊖	circled equals
char	'	character
check	✓	checkmark
chi	χ	chi
Chi	X	Chi
circ	○	circle
circle	○	circle
CLUB	♣	club solid
cminus	⊖	circled minus
colon	:	colon
comma	,	comma
concat		concatenate
congruent	≈	congruent
cont		continuation character
contains	⊃	contains as a subset
copr	©	copyright
copyr	©	copyright
cplus	⊕	circled plus
csq	'	close single quote
csqg	'	German close single quote
ctimes	⊗	circled times
currency	¤	currency international
cursor	■	fat cursor
dagger	†	dagger
dahead	▼	down arrowhead
darrow	↓	down arrow
date	October 30, 2000	date
dbldag	‡	double dagger
dbls	§	double S
dblus	=	underscore double
dblxclam	!!	double exclamation point
dblxclm	!!	double exclamation point
decrease	↘	decrease
def	::=	definition/defined as
deg	°	degree
degree	°	degree

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
del	∇	del
delta	δ	delta
Delta	Δ	Delta
determines	↗	determines
diam	◇	diamond wide
diamond	◇	diamond
DIAMOND	◆	diamond solid
div	÷	divide
divide	÷	divide
divslash	/	division slash
dollar	\$	dollar
dot	·	dot
dotdot	..	double dot
doubleC	©	double C
doubleN	ℕ	double N
doubleP	ℙ	double P
doubleQ	ℚ	double Q
doubleR	ℝ	double R
doubleZ	ℤ	double Z
Dstroke	Ɔ	Eth or D stroke
ea	é	e acute
Ea	É	E acute
ebin	'	binary end
ec	ê	e circumflex
Ec	Ê	E circumflex
echar	'	character end
ee	ë	e umlaut
Ee	Ë	E umlaut
eg	è	egrave
Eg	È	E grave
egml	.	gml end tag delimiter
ehex	'	end quoted hex string
ellip	...	ellipsis
ellipsis	...	ellipsis
emdash	—	em dash
endash	–	en dash, dash
epsilon	ε	epsilon
Epsilon	E	Epsilon
eq	=	equals
eqsym	=	equals
equals	=	equals
eqv	~	equivalent
eserver_logo	@server	e(logo)server
eserver_logo_TM	@server	e(logo)server, trademarked
eta	η	eta
Eta	H	Eta
eth	ð	eth, Icelandic small
Eth	Ð	eth, Icelandic capital
euler	ℰ	Eulers
euro	€	Either Euro glyph or EUR
eurochar	€	Either Euro glyph or E
euromtext	EUR	Always EUR
exists	∃	exists

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
FACE	●	face solid
face	⊙	face
factorial	!	factorial
female	♀	female symbol
ff	ff	ff ligature
ffi	ffi	ffi ligature
ffl	ffl	ffl ligature
fi	fi	fi ligature
finespace		finespace
fl	fl	fl ligature
florin	ƒ	florin
fnof	ƒ	function of
frac12	$\frac{1}{2}$	one half
frac14	$\frac{1}{4}$	one quarter
frac18	$\frac{1}{8}$	one eighth
frac34	$\frac{3}{4}$	three fourths
frac38	$\frac{3}{8}$	three eighths
frac58	$\frac{5}{8}$	five eighths
frac78	$\frac{7}{8}$	seven eighths
gamma	γ	gamma
Gamma	Γ	Gamma
ge	≥	greater than or equal to
gerank	⋇	greater than or equal rank
gesym	≥	greater than or equal to
gml	:	gml delimiter
grave	'	accent grave
gt	>	greater than
gtequiv	≧	greater than or equivalent
gtgt	⋇	much greater than
gtlt	≧	greater than or less than
gtrank	>	greater than rank
gtsym	>	greater than
hamilton	\mathcal{H}	hamiltonian
hat	^	hat
hbar	ℏ	h bar
HEART	♥	heart solid
hex	X'	hex
house	△	house
hyphen	-	hyphen
ia	í	i acute
Ia	Í	I acute
ic	î	i circumflex
Ic	Î	I circumflex
Icap	I	I capital character
identical	≡	identical
idotless	ı	i dotless
ie	ï	i umlaut
Ie	Ï	I umlaut
iff	↔	if and only if
ig	ì	i grave
Ig	Ì	I grave
ij	ij	ij ligature
increase	↗	increase

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
infinity	∞	infinity
intbot	\int	integral bottom half
integral	\int	integral
intersect	\cap	intersection of sets
inttop	\int	integral top half
inve	!	inverted exclamation
invellip	⋮	indented vertical ellipsis
invq	¿	inverted question mark
iota	ι	iota
Iota	Ι	Iota
isubset	\subsetneq	improper subset
isuperset	\supsetneq	improper superset
join	∪	join
kappa	κ	kappa
Kappa	Κ	Kappa
lahead	◄	left arrowhead
lambda	λ	lambda
Lambda	Λ	Lambda
larrow	←	left arrow
lbarb	↵	left barb
lbrace	{	left brace
lbracket	[left bracket
lbrc	{	left brace
lbrk	[left bracket
lbullet	●	large bullet
ldarrow	⇐	left double arrow
le	≤	less than or equal to
lerank	≲	less than or equal rank
less	≤	less than or equivalent
lesym	≤	less than or equal to
liter	ℓ	liter
lmultidot	•	multiply dot large
lnot	¬	logical not
lnotrev	⌞	backward logical not
lnotusd		upside down not
lor		logical or
loz	◻	lozenge
lozenge	◻	lozenge
lpar	(left parenthesis
lparen	(left parenthesis
lrrarrow	↔	left-right arrow
Lsterling	£	pound sterling
lt	<	less than
ltequiv	≲	ltequiv
ltlt	≪	much less than
ltrank	<	less than rank
ltsym	<	less than
male	♂	male symbol
mathast	*	mathematics asterisk
mdash	—	em dash
meet	∩	meet
memberof	∈	member of

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
minus	–	minus operation
minusop	–	minus operation
mp	∓	minus-plus
mu	μ	mu
Mu	Μ	Mu
mult	×	multiply
ndash	–	en dash, dash
ne	≠	not equal to
nearly	≈	nearly equal
nesym	≠	not equal to
nexists	∄	not existant
nidentical	≠	not identical
nisubset	⊄	not improper subset
nisuperset	⊈	not improper superset
nlerank	≠	not less or equal rank
nltrank	≠	not less than rank
nmemberof	∉	not a member of
nnearly	≠	not nearly equal
note1616	♪	pair of 16th notes
note18	♪	eighth note
notsym	≠	not symbol
nsubset	⊆	not a subset
nsuperset	⊇	not a superset
nt	ñ	n tilde
Nt	Ñ	N tilde
nu	ν	nu
Nu	Ν	Nu
numsign	#	number sign
oa	ó	o acute
Oa	Ó	O acute
oc	ô	o circumflex
Oc	Ô	O circumflex
odq	“	open double quote
odqf	«	French open double quote
odqg	„	German open double quote
oe	ö	o umlaut
Oe	Ö	O umlaut
oelig	œ	oe ligature
OElig	Œ	OE ligature
og	ò	o grave
Og	Ò	O grave
omega	ω	omega
Omega	Ω	Omega
omicron	ο	omicron
Omicron	Ο	Omicron
or	∨	or symbol
os	ø	o slash
Os	Ø	O slash
osq	‘	open single quote
osqg	‚	German open single quote
ot	õ	o tilde
Ot	Õ	O tilde
ous	◌ _o	underscored o

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
overline	$\overline{}$	overline
par	¶	paragraph
parallel		parallel
partial	∂	partial
per	.	period (starter set)
percent	%	percent
period	.	period
perpend	\perp	perpendicular
peseta	₧	peseta
phi	ϕ	phi
Phi	Φ	Phi
pi	π	pi
Pi	Π	Pi
planck	\hbar	h bar
plus	+	plus
plusend	+	plus at end of line
plusmin	\pm	plus-minus
plusop	+	plus operation
pm	\pm	plus-minus
prime	'	prime
product	\prod	product
proportion	\propto	proportion
psi	ψ	psi
Psi	Ψ	Psi
quest	?	question mark
rahead	►	right arrowhead
rarrow	→	right arrow
ratio	:	ratio
rbarb	→	right barb
rbl		required blank
rbrace	}	right brace
rbracket]	right bracket
rbrc	}	right brace
rbrk]	right bracket
rdarrow	⇒	right double arrow
regtm	®	registered trademark
revbul	◐	reverse bullet
revcir	◑	reverse circle
rho	ρ	rho
Rho	ρ	Rho
riemann	\mathcal{R}	riemann integral
rpar)	right parenthesis
rparen)	right parenthesis
rprime	'	right prime
Rx	℞	physician Rx
scriptI	\mathcal{I}	script I
scriptl	ℓ	liter
sdq	"	straight double quote
sect	§	double S
section	§	double S
semi	;	semicolon
shiftin	␣	double byte shift in
shiftout	␣	double byte shift out

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
sigma	σ	sigma
Sigma	Σ	Sigma
similar	\sim	similar
slash	/	slash right
slr	/	slash right
smuldot	\cdot	multiply dot small
SPADE	♠	spade solid
splitvbar		split veritcal bar
sqbul	▪	square bullet
sqbullet	▪	square bullet
sqrt	$\sqrt{}$	square root
ss	ß	German es-zet
ssbl		syntax significant blank
ssq	'	straight single quote
STAR	★	star solid
sublpar	(subscript left parenthesis
subminus	-	subscript minus
subplus	+	subscript plus
subrpar)	subscript right parenthesis
subset	\subset	subset of, included in
sub0	0	subscript 0
sub1	1	subscript 1
sub2	2	subscript 2
sub3	3	subscript 3
sub4	4	subscript 4
sub5	5	subscript 5
sub6	6	subscript 6
sub7	7	subscript 7
sub8	8	subscript 8
sub9	9	subscript 9
suchthat	\ni	such that
sum	Σ	sum
sun	☼	sun
superset	\supset	superset
suplpar	(superscript left parenthesis
supminus	-	superscript minus
supn	n	superscript n
supplus	+	superscript plus
suprpar)	superscript right parenthesis
sup0	0	superscript 0
sup1	1	superscript 1
sup2	2	superscript 2
sup3	3	superscript 3
sup4	4	superscript 4
sup5	5	superscript 5
sup6	6	superscript 6
sup7	7	superscript 7
sup8	8	superscript 8
sup9	9	superscript 9
tab		tab
tau	τ	tau
Tau	T	Tau
telephone	☎	telephone

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
TELEPHONE	☎	telephone solid
therefore	∴	therefore
theta	θ	theta
Theta	Θ	Theta
thorn	þ	thorn, Icelandic small
Thorn	Þ	Thorn, Icelandic capital
tilde	~	tilde
time	8:48 a.m.	time
times	×	multiply
tm	™	trademark
TRIANGLE	▲	triangle solid
triangle	△	triangle
ua	ú	u acute
Ua	Ú	U acute
uahead	▲	up arrowhead
uarrow	↑	up arrow
uc	û	u circumflex
Uc	Û	U circumflex
udarrow	↕	up-down arrow
udarrowus	↕	up/down arrow/underscore
ue	ü	u umlaut
Ue	Ü	U umlaut
ug	ù	u grave
Ug	Û	U grave
ulbarb	┑	up left barb
umlaut	¨	umlaut
union	∪	union of 2 sets
upsilon	υ	upsilon
Upsilon	Υ	Upsilon
urbarb	┑	up right barb
us	—	underscore
usec	μ	micro second
vardelta	∂	delta (variation)
varphi	φ	phi (variation)
varsigma	ς	sigma (variation)
vartheta	θ	theta (variation)
vbar		vertical bar
vector	→	vector
vellip	⋮	vertical ellipsis
weierstr	℘	weierstrass elliptic
won	₩	won (Korean currency)
xclam	!	exclamation point
xclm	!	exclamation point
xi	ξ	xi
Xi	Ξ	Xi
ya	ý	y acute
Ya	Ý	Y acute
ye	ÿ	y umlaut
Ye	Ÿ	Y umlaut
yen	¥	yen
zero	0	zero slashed
zeta	ζ	zeta

Table 16. IBMIDDoc Character Entities (continued)

Symbol	Appearance	Description
Zeta	Z	Zeta
The following entities define the character graphic symbols:		
Ad	↓	arrow down
Al	◀	arrow left
Ar	▶	arrow right
Au	↑	arrow up
Eb		end of line, bottom
El	—	end of line, left
Er	—	end of line, right
Et		end of line, top
Ju	+	line junction
Lh	—	line horizontal
Ll	└	lower left corner
Lr	└	lower right corner
Lv		line vertical
Td	⊥	T char, bar down
Tl	└	T char, bar left
Tr	└	T char, bar right
Tu	⊥	T char, bar up
Ul	┐	upper left corner
Ur	┐	upper right corner
The following entities make description of the SGML syntax easier to show:		
stago	<	start tag open
etago	</	end tag open
tagc	>	tag close
mdo	<!	markup declaration open
mdc	>	markup declaration close
pio	<?	processing instruc. open
pic	>	processing instruc; close
pero	%	parm entity ref; open
ero	&	entity reference open
erc	;	entity reference close
dso	[declaration subset open
dsc]	declaration subset close
msc]]>	marked section close
lit	"	literal delimiter
lita	'	alternate literal delimiter
The following entities define typographic quote symbols which are used because the Q element has implied citation semantics:		
ctq	"	close typographic quote
otq	"	open typographic quote
The following entities can be used by translation centers for hyphenation:		
shy		soft hyphen

Reusing elements from an object library

Elements that are to be reused many times throughout a document can be defined in an object library. Object libraries are an alternative to using file or text entities. You create object libraries by using the OBJLIB element in the prolog. You place the elements and the content you want to reuse in that object library. The elements in the object library must have an ID.

To use an element from the object library, use the CONLOC attribute to refer to that element. The element in the object library must be the same as the element with the CONLOC attribute. That is, P tags refer to P tags; LI tags refer to LI tags; a PBLK tag cannot refer to a P tag.

The following example shows a small object library and two references to elements in the library. This object library contains an introductory paragraph for service needs. It also contains an ordered list of things that must be done if service is required.

```
<prolog>
:
<objlib>
<objlibbody>
<p id="para1">If your system stops
working, follow these instructions:</p>
<ol id="list2">
<li>Note the system code displayed on the front of
the unit.</li>
<li>Unplug the unit.</li>
<li>Contact your service representative.</li>
</ol>
</objlibbody>
</objlib>
:
</prolog>
:
<d>
<dprolog><titleblk>
<title>If you need service</title>
</titleblk></dprolog>
<dbody>
<p conloc="para1">
<ol conloc="list2">
</dbody></d>
```

When the P element with the CONLOC attribute of “para1” is processed, the content of the P element in the OBJLIB with the “para1” attribute is used. This markup portion:

```
<p conloc="para1">
```

Is the same as specifying this markup:

```
<p>If your system stops
working, follow these instructions:</p>
```

The content of an element defined in an object library is used only if you refer to that element in the document content.

An element defined in an object library can be referred to only from within the document containing the object library. If you have information that will be re-used by other documents, the object library can be declared as a file entity and imbedded in each document. This level of reuse allows much more flexibility and function for reuse across documents.

Migration Note

ObjLib can also be used as, in the Bookmaster paradigm, a DVCF side file that uses the include macro.

The element with the ID must be a direct child of the ObjLibBody tag. You cannot use a CONLOC to refer to an element nested inside something else in the ObjLibBody tag. For example, referencing the LI element “unplug” in the next example is **not correct**:

```
<objlib>
<objlibbody>
<ol id="list2">
<li>Note the system code displayed on the front of
the unit.</li>
<li id="unplug">Unplug the unit.</li>
<li>Contact your service representative.</li>
</ol>
</objlibbody>
</objlib>
```

To correct the example, you need move the LI outside the list, and include an LI with a CONLOC. The items reused within an object library must be defined before they are referenced, so the LI is before the list.

```
<objlib>
<objlibbody>
<li id="unplug">Unplug the unit.</li>
<ol id="list2">
<li>Note the system code displayed on the front of
the unit.</li>
<li conloc="unplug">
<li>Contact your service representative.</li>
</ol>
</objlibbody>
</objlib>
```

If you want to reuse content of a part of a list, containing the content you wish to reuse within an LIBlk element makes it easy to reference the LIBlk using the CONLOC attribute.

You can have several divisions that get reused by using the DBLK tag to contain those divisions.

Reusing attributes in the CONLOC reference

Starting with IDWB release 3.4, patch IDWXF036, the attributes for an item in an object library are now passed through to the reference. For example, you have a list item with a revision ID:

```
<objlib>
<objlibbody>
<li id="renew" rev="rel34a">Renew your subscription</li>
</objlibbody>
</objlib>
```

You can refer to the list item, and the REV attribute is carried along. For example:

```
<li conloc="renew">
```

is now the same as this:

```
<li rev="rel34a">Renew your subscription</li>
```

Before patch IDWXF036, you would have only gotten the text, the REV attribute would be ignored:

```
<li>Renew your subscription</li>
```

If the item in the object library and its reference have the same attributes, the value on the CONLOC reference wins.

Cross-referencing items that use CONLOC

Now, for every solution, there is a problem.⁶If you want to cross-reference an item with a CONLOC, you need to add the ID to the tag with the CONLOC. For example, you want to reuse a division, plus cross-reference to it. You cannot cross-reference to “service”. You need to add unique IDs to each division.:

```
<objlib>
<objlibbody>
<d id="service">
<dprolog><titleblk>
<title>If you need service</title>
</titleblk></dprolog>
<dbody>
<p>If you need service...
</dbody></d></objlibbody>
</objlib>
...
<d conloc="service" id="abc">
...
<d conloc="service" id="def">
...
<xref refid="abc">
...
<xref refid="def">
```

6. Yes, I wrote that correctly.

Chapter 18. Conditionally including information

Where multiple output documents contain common information, or a single document repeats the same information, the only way to ensure that the common information is the same is to take the information in each case from the same source file. At its simplest level, where the common information is in a few, largish blocks, the way to do this is to put each block of the common information in a file of its own and imbed it in the multiple output documents that use it.

However, this process becomes cumbersome and inefficient where the common information is strewn throughout the document, or, alternatively, where the differences for the multiple output documents are scattered through the common information. This approach also has the drawback that reviewers of the documents must do redundant reviewing of the common information. For these reasons, we have property-based retrieval. See “Property-Based Retrieval”.

There are also times when you need alternative text under different conditions. You can specify the modification text (the insertion or replacement) either in line in the source file or out of line in an object library. See “Retrieval alternatives” on page 174.

Property-Based Retrieval

Property-based retrieval allows you to structure documents such that different versions can be produced based on a property value file (that is, a set of one or more conditions). With property-based retrieval, you can:

- Insert, delete, or replace text based on conditions you specify.
- Specify conditions that are simple or complex.
- Pass the conditions at run time or inside the document itself.

In IBMIDDoc, conditional processing is done by evaluating certain attribute values on elements to determine whether or not those elements should be processed. The attributes are called *property attributes* because they define properties of elements.

The Props Attribute

The Props attribute on a tag, when true, causes that content to be processed and appear. The attribute can contain a simple condition that is either true or false, or a complex condition of Boolean operators. The attribute’s value, the specification, can be any valid SGML name, but should be a word or phrase that is clear and meaningful to those who are writing or editing the information. The specification may be a version, a software code name, or a hardware platform; for example, v4r5, win32, or PC, respectively.

It is up to those working on a product or group of products to choose consistent terminology for assigning occurrences of the Props attribute. The names and meaning should be documented and available to all writers and editors involved in the development effort to ensure consistency.

For example, each these items will format when the conditions v4r5, win32, and PC are all true:

```

<ul>
<li props="v4r5">The version 4, release 5 level has
special stuff.</li>
<li props="win32">This software release has more special
stuff.</li>
<li props="PC">Is the PC going to be replaced?</li>
</ul>

```

Conditional text can be a powerful feature, but you need to use care because of translation considerations. If you have a condition within a sentence, ensure it is a noun string. Do not combine a noun and a verb in the same conditional phrase. Not every language has sentence constructs like English (or whatever language you are writing in).

If you have a condition within a sentence, ensure your sentence makes sense with all possible logic conditions. Here are some examples:

- Here's a good example:

The <ph props="os2">OS/2</ph><ph props="#NOT os2">Windows</ph> operating system runs on PCs.

This way you always get a complete sentence, as in this when "os2" is true:

The OS/2 operating system runs on PCs.

And this when "os2" is false:

The Windows operating system runs on PCs.

- In this next example — you get an incomplete sentence when both conditions are false:

When <ph props="equine">horses gallop</ph><ph props="canine">dogs run</ph> down the track, you will know the race has begun.

When both "equine" and "canine" are false, the sentence becomes:

When down the track, you will know the race has begun.

Setting the properties to true or false

The properties are initially assumed to be false. When you process your document, messages indicate that when a condition is found, false is assumed.

To set a property value, you can do the following:

- Use a VAL file to set the properties to true or false. This is described in the *ID Workbench Getting Started and User's Guide* and in the online help for ID Workbench. See the **Transform** tab in the processing option displays.
- Use a PropDesc tag in the document's prolog.

Here is an example showing how to set the properties "v4r5" and "PC" to true and false, respectively.

```

<prolog>
...
<propdefs>
...
<propdesc propname="v4r5" default="true">
<desc>Version 4, release 5</desc>
</propdesc>
<propdesc propname="pc" default="false">
<desc>PC platform information.</desc>

```

```

</propdesc>
...
</propdefs>
...
</prolog>

```

Specifying boolean properties

Sometimes you need to specify fancier conditions, called complex conditions. Suppose we have two conditions, A and B. To have an action (insert or delete) occur:

- When (and only when) both conditions are true, enter:
a #AND b
- When either one or both of the conditions is true, enter:
a #OR b
- When a condition is not true, enter:
#NOT a

You might have a situation where you want one sentence for one condition, and different sentence for the opposite condition. Instead of having two properties and setting one to true and the other to false; you can just have one property and use the #NOT operator. For example:

```

<ul>
<li>Always use this item</li>
<li props="PC">This is a PC-only item</li>
<li props="#NOT PC">This item is for everything EXCEPT PCs</li>
</ul>

```

The order of precedence in evaluation is:

1. specifications inside parentheses are evaluated first
2. #NOT specifications are evaluated next
3. Finally, #AND and #OR operators are evaluated from left to right

These three functions, #AND, #OR, and #NOT, can be strung out to the point where only a computer could figure out what to do. See Table 17 for a set of sample conditions and the results with different logic groupings.

Table 17. Property Truth Table. T means true; blank indicates false.

These conditions			Yield these complex conditions:						
A	B	C	A #and B	A #or B	A #and #not B	A #and B #or C	A #or B #and C	A #and #not B #or C	A #or #not B
T	T	T	T	T		T	T	T	T
T	T	F	T	T		T	T		T
T	F	T		T	T	T	T	T	T
T	F	F		T	T		T	T	T
F	T	T		T		T	T	T	
F	T	F		T					
F	F	T				T		T	T
F	F	F							T

This example shows that the second paragraph applies to MVS in both versions of the product, and to VM in only the first version of the product.

```
<p PROPS="(VM #or MVS) #and (V1 #or V2)">
This paragraph applies to both versions and operating systems.
</p>
<p PROPS="MVS #or (VM #and V1)">
This paragraph applies to MVS or version 1 on VM.
</p>
<p PROPS="VM #and V2">
This paragraph applies only to version 2 on on VM.
</p>
```

Retrieval alternatives

Sometimes you need alternative text for a condition. The RETALTS attribute can point to one or more retrieval alternatives for the text in your document.

You need to create your alternative text in an object library in the prolog of your document. The main element will need an ID. See “Reusing elements from an object library” on page 166 for information about creating object libraries.

To have the alternative text be used:

- The main element must have a false Props attribute, and a Retalts attribute with one or more IDs.
- The Retalts attribute points at the ID of a replacement element found in the object library. The elements must be the same type. The first true element is used for the alternative text.

The following example illustrates this type of conditional processing.

```
<prolog>
...
<objlib>
<objlibbody>
<p id="v1intro" props="v1">This is an introduction for version 1...</p>
<p id="v2intro" props="v2">This is an introduction for version 2...</p>
<p id="v3intro">This is an introduction for version 3...</p>
</objlibbody>
</objlib>
...
</prolog>
...
<p props="x" retalts="v1intro v2intro v3intro">This
is an introduction...</p>
```

The paragraphs will print under these conditions:

- When “x” is true, this prints:
This is an introduction...
- When “x” is false and “v1” is true, this prints:
This is an introduction for version 1...
- When “x” and “v1” are false and “v2” is true, this prints:
This is an introduction for version 2...
- When “x”, “v1”, and “v2” are false, this prints:
This is an introduction for version 3...

Using Marked Sections

Marked sections provide two key functions:

1. they allow conditional inclusion or exclusion of material and
2. they control SGML delimiter recognition for documenting SGML and markup as well as documenting other subjects that use SGML markup characters for other purposes.

Marked sections should not be used to provide conditional processing capability. Use the property-based retrieval function instead.

Marked sections have the following format:

```
<![ keyword status area [marked section data]]>
```

keyword status area

This is specifications that control the function of the marked section. See below for possible values.

marked section data

data to be treated based on the content of the keyword status area.

Marked sections support two keywords for conditional inclusion and exclusion: IGNORE and INCLUDE. One or both are specified in the keyword status area described above. If both are specified, IGNORE has higher precedence. Here is an example:

```
<![ IGNORE [  
This information will be ignored.  
]]>
```

```
<![ INCLUDE [  
This information will be included.  
]]>
```

Why would anyone would specify both INCLUDE and IGNORE? The keyword status area may include parameter entity references that allow the author to parameterize these inclusions from the document prolog without changing the document (and marked section keyword status area) content. Examples of this follow the parser recognition control description.

Parameter entities may be used to parameterize the keywords found in a marked section keyword status area. This is particularly useful in conditional processing cases. For example, assuming you have material that is intended for two uses, say reference cards and full language reference, you can encode both in the same document and then change the parameters to include just the material for the output currently desired. Here is an example:

```
<!DOCTYPE IBMIDDOC PUBLIC "-//..." [  
<!ENTITY % langrefonly "include" >  
<!ENTITY % refcardonly "ignore" >  
:  
<P>Material that belongs in both output docs doesn't have any marked  
section markup  
<![ %langrefonly; [  
This is material that goes only in the language reference.  
]]>  
<![ %refcardonly; [  
This is material that goes only in the reference card.  
]]>  
More material that goes in both.
```

Controlling SGML Delimiter Recognition

There are two keywords to control SGML delimiter recognition:

CDATA

inhibits the recognition of all markup except the marked section close delimiters ']]>'

RCDATA

supports recognition of entity references and the marked section close delimiter.

You use this for including SGML markup examples and for including other material that uses SGML markup delimiter characters in other ways:

```
<![ CDATA [  
<P>This is an example paragraph with an example entity reference:  
&entref;.  
]]>
```

```
<![ RCDATA [  
<p>This is an example paragraph of IBMIDoc coding with an example symbol reference: &bkmsym; ]]></p>
```

In the second case, the & SGML entity resolves to an & that gives the correct result in documenting IBMIDoc encoding.

Chapter 19. Property and Class Definition

This chapter describes how to define element properties, element classes, and properties for element classes.

If the referenced element also has a PROPSRC specification, this reference is followed until the end of the chain of property specification is reached. Property specifications on the referencing element override any properties from referenced or inherited property specifications. The hierarchy of property use is:

1. properties specified on the element,
2. properties specified on an element which is referenced using the CONLOC attribute or properties specified on an element which is referenced using the RETALTS attribute and whose properties were satisfied for this processing run (the CONLOC and RETALTS referenced elements are treated as independent elements and their properties do not interact),
3. properties from a PROPSRC referenced element, however long the reference chain may be,
4. properties from ClassDef elements referenced by class.
5. properties specified on a PropDef element without an ID (with or without an ELETYPES attribute)
6. properties from ancestors in the document tree

Defining Element Properties

In IBMIDDoc, you can define properties for an element, such as language, status, and classification. You can define properties directly, by linking to another element, through inheritance, or, for security classification, by implying it from an element's children. You can also define reusable sets of properties.

Defining Element Properties Directly

Properties can be defined for an element by using the property attributes.

The Propdef tag sets default properties for tags that point at them with propsrc attributes. This allows you to set defaults once, and reference them. When something changes, you only have to change the propdef). The only attribute currently able to be passed from PROPDEF to an element is STYLE. For example; this propdef sets a style of bold for the propname fred:

```
<propdef propname="fred" style="bold">
```

Now, when a tag that supports a bold style uses the fred property, the content will be bold:

```
<ph propsrc="fred">hi there</ph>
```

Values of the same attributes on the tags override the values on the propdef tag. The following tag, because the style attribute is used on the tag itself, overrides the style attribute on the propdef. It will be italic:

```
<ph propsrc="fred" style="italic">hi there</ph>
```

The following shows a propdef that sets a style of bold and a conditional property of aix:

```
<propdef propname="fredaix" style="bold" props="aix">
```

Consider this tag:

```
<ph propsrc="fredaix" props="win">hi there</ph>
```

Because the attributes on the tags override the same attributes on the propdef, the ph tag effectively becomes the following:

```
<ph style="bold" props="win">hi there</ph>
```

This is because the props attribute on the tag overrides the props attribute on the propdef. The style attribute on the propdef is carried through to the tag.

You can leave off the propname and propsrc to get a property default for all the tags. For example, this propdef sets a style of bold for any tag that supports a style of bold:

```
<propdef style="bold">
```

These will print in bold:

```
<ph>hi there</ph>  
<term>hello again</term>
```

Additionally, because you can define style overrides on both propdef and classdef, then use them together, the ID Workbench uses a rule determines which will win. The special override rule for style and class when used together is: 1) Style on the tag, then 2) Class on the tag, then 3) Style on Propdef.

The following example shows a paragraph with defined properties of OS/2 V2.1 only.

```
<P props="os2 #AND v21">This paragraph  
is used for OS/2 V2.1 only.
```

Properties defined directly take precedence over properties defined by linking or through inheritance. The only exception is the security classification property, which can be implied from an element's children.

Defining Element Properties by Linking

An element can take on the properties of another element by linking to the other element as a property source. The following example shows a division that takes on the properties defined in a PropDef element:

```
<PROLOG>  
  <PROPDEFS>  
    <PROPDEF PROPNAME="ONENOT2" PROPS="(version1 #AND #NOT version2)">  
  </PROPDEFS>  
</PROLOG>  
<BODY>  
  <D PROPSRC="ONENOT2">For Version 1 and Not Version 2  
    <P>  
  
    :
```

Properties defined by linking take precedence over the properties an element inherits from its parent.

Defining Element Properties Using Inheritance

All elements inherit their parent's properties. For example, a paragraph within a division that has a language property value of Spanish will inherit that property and will thus be identified as Spanish as well.

Defining the Security Classification from an Element's Children

The security classification property can be implied from an element's children. That is, an element's security classification is the highest classification of that of any of its children, even if a lower security classification is directly defined for the parent.

Defining Reusable Sets of Element Properties

To define a single set of properties for several elements, you can use and link to the PropDef element. Suppose you have a complex specification that applies to more than one element. Rather than defining the specification for each element that needs it, you can define it once with a PropDef element and use the PROPSRC attribute to link to the PropDef element, as shown in this example:

```
<PROLOG>
  <PROPDEFS>
    <PROPDEF PROPNAME="ONENOT2"
      PROPS="(version1 #AND #NOT version2) #AND MVS #AND 370 #AND A11">
    </PROPDEFS>
  </PROLOG>
  <BODY>
    <D PROPSRC="ONENOTE2">For Version 1 and Not Version 2
    <P>

    :
    <D PROPSRC="ONENOT2">XYZ Function
    <P>

    :
```

In the example, the PropDef element has an ID value of *v1not2*, and a specification indicating a number of properties associated with a certain version. Because the D element uses the properties defined in the PropDef element, the division is used only when those properties are active.

Note: The property values that are active when you process a document depend on your processing system. In many systems, you can define the active property values using an input panel or an options file.

Defining Element Classes

You can use the ClassDef element to define classes of IBMIDDoc elements. Using classes of elements enables processing functions such as detailed glossaries or bibliographies, precise associative links, presentation effects unique to your information, or automatic indexing. You can also use classes to control the inheritance of element properties.

Typically, element classes are used to define specific phrase classes that reflect the product being described. You can enable the various processing functions by using the Phrase elements with the classes you define.

Usually, element classes are defined for an entire collection of documents by someone responsible for designing the information in the collection, such as an

information designer or planner. If you are working on information for which element classes have been defined, you do not need to understand how classes are defined. However, you do need to know the class names and the affected elements.

In defining element classes, first determine what classes are needed and decide the class names. Analyze your product to identify what kinds of things you need to write about. Classes should be meaningful and should describe real things or aspects of real things. Classes should not relate purely to processing or presentation effects. Thus, a class of *bold* is probably not meaningful. Usually, class names should be nouns.

Next, define exactly what the classes are so that you understand when and why to use them.

Then, define the element classes using ClassDef elements. ClassDef elements are valid within PropDefs, which is in either the document prolog or a division prolog. If a class applies to an entire document, put the ClassDef element in the document prolog. Use the Sem element within each ClassDef element to describe the class.

Finally, after you define the classes, use the class names with the elements to which they apply to assign element classes in your document.

Use ClassDef to define element classes that are specific to your information. The most common use of ClassDef is to define new phrase classes. For example, in the documentation of a graphical user interface, you may want to define phrase classes for all the different types of user interface elements in order to make your markup more precise and to enable the automatic generation of indexes for print presentation.

You can associate presentation styles and other processing with specific element classes. Do not define classes that are purely presentational, such as Bold or Italic.

Suppose, for example, that you are documenting software that uses two important types of objects, Whatsits and Thingies, that are not accounted for in the base IBMIDDoc language. Whatsits are hardware components, and Thingies are software components. The class names for Whatsit and Thingy objects are "Whatsit" and "Thingy", respectively. Because the Whatsit and Thingy classes apply to an entire document, they are defined in the prolog.

The following example shows the ClassDef elements that define the Whatsit and Thingy classes and the use of the classes with the Ph element:

```
<PROLOG>
  ⋮
  <PROPDEFS>
    <CLASSDEF CLASSNAME="whatsit">
      <SEM>Identifies whatsit objects.  Whatsits are hardware
        components.
    </CLASSDEF>
    <CLASSDEF CLASSNAME="thingy">
      <SEM>Identifies thingy objects.  Thingies are software
        components.
    </CLASSDEF>
  </PROPDEFS>
  ⋮
</PROLOG>
  ⋮
  <D>Hardware Problems
    <P>Hardware problems are usually caused when
```

```

the <PH CLASS="whatsit">famtoozler</PH>
gets out of adjustment.
Readjust it using the <PH CLASS="thingy">famtoozlometer</PH>
component of the &prodname; analyzer.

```

Using classes with phrase elements can improve the retrievability of your information by enabling precise searching and associative linking, as well as automatic indexing.

Precise searching is enabled for online presentation systems that preserve the structure of the original document markup. In those systems, a reader can specify a search so that only elements of a given class satisfy the search. For example, a reader could specify a search to find all elements of class Whatsit in your document.

Associative linking is enabled because an online presentation system can, for example, associate phrases of a certain class with other elements that both have the same class and contain the text of the phrase. Suppose you define all the different Whatsits in a definition list, as shown in this example:

```

<DL CLASS="whatsit"><TITLE>Definition of Whatsits</TITLE>
<DENTRY>
  <TERM>famtoozler
  <DEFN>A thing that famtoozles
<DENTRY>
  <TERM>whantingler
  <DEFN>A thing that whantingles
  :
</DL>

```

If you assign the class Whatsit to the definition list, an online presentation system can automatically associate any phrase of class Whatsit with the definition list.

Property-Reference Links

Elements can refer to other elements in order to inherit their property values using the linkends attribute. The PROPSRC attribute is used to refer to a PropDef element. By referring to a PropDef element, its property values can be used by the element using the PROPSRC attribute.

In the example that follows, the phrase in the Defn element concerning the napping habits of Maine Coon cats will appear as **ITALIC** when the document is formatted. Its style is determined by naming its PROPSRC as ID=PHLOOK.

```

<PROPDEFS>
  <PROPDEF PROPNAME="phlook" ELETYPES="ph" STYLE="bold italic">
    <DESC>This PropDef defines the presentation
      style for special phrases.</DESC>
  </PROPDEF>
  :
<GLDEFS>
  <GENTRY>
    <TERM ID="mainec">Maine Coon</TERM>
    <DEFN>A friendly and gentle breed of cat.
    <ANNOT>
      <ANNOTBODY>
        <P>A Maine Coon's motto is <PH PROPSRC="phlook">Nap early,
          nap often.</PH>
        </P>
      </ANNOTBODY>
    </ANNOT>
  </GENTRY>

```

```
</ANNOT>  
</DEFN>  
</GENTRY>  
  
:  
</GLDEFS>  
:  
:
```

Chapter 20. Creating maintenance analysis procedures

IBMIDDoc provides a handy format for writing step-by-step procedures to help isolate the cause of a symptom. These procedures are called Maintenance Analysis Procedures (MAPs). If you are familiar with flowcharts, you know how they lead you through a sometimes complex series of steps by having you make one simple yes-or-no decision at a time. However, flowcharts can be difficult to work with because the flowcharting symbols contain so little space for writing questions, directions, or other text. We solve this space problem while keeping the technique of using simple yes-and-no answers to lead people through their procedures.

Procedures consist of several parts:

- procedure entry (see “Using ProcEntry for Entry Requirements” on page 185)
- procedure steps and commands (see “Using ProcStep and ProcCmnd to Describe Each Step” on page 185)
- decision points (see “Using DecisionPnt for Outcome-Dependent Action Descriptions” on page 186)
- reference keys (see “Using RefKeys to Refer to Labels in a Graphic” on page 186)
- procedure exit (see “Using ProcExit to Complete a Procedure or Sub-Procedure” on page 187)

The following shows a sample map some father made for caring for his little one.

MAP 0010: Baby Johnny is crying

Six-month-old Baby Johnny was sleeping peacefully; suddenly he began to cry.

001

- Check Johnny’s diaper.

Is the diaper wet?

Yes No

Continue at Step 003.

002

- Change the diaper.

Johnny was uncomfortable.

003

(From step 001)

Is Johnny hungry?

Yes No

004

- Rock Johnny to sleep.

Johnny was sleepy.

005

Does Johnny have teeth?

Yes No

006

- Warm a bottle.
- Feed Johnny.

Johnny needed a bottle.

007

Johnny can eat solid food.

Continue at “MAP 0020: The Steak is Frozen” on page 185.

MAP 0020: The Steak is Frozen

001

Do you have a microwave oven?

Yes No

002

– Johnny can't wait for it to thaw.

Continue at Step 006 on page 184.

003

– Thaw the steak.

Using ProcEntry for Entry Requirements

The ProcEntry element contains a description of the entry point to the procedure. It contains the description, and references to any prerequisite or related procedures. Related and prerequisite procedures are referenced by ID using the RELPROCS and PREREQPROCS attributes.

```
<PROCENTRY PREREQPROCS="proca" RELPROCS="proc1 proc2 proc3">
SIX-MONTH-OLD BABY JOHNNY WAS SLEEPING PEACEFULLY;
SUDDENLY HE BEGAN TO CRY.
</PROCENTRY>
```

Using ProcStep and ProcCmnd to Describe Each Step

The ProcStep element contains the actions to take and the expected results of taking the actions. The title for each ProcStep is contained in the required TitleBlk elements. The Desc element contains the description of the action that must be performed.

The ProcCmnd element contains specific instructions that the user must follow in order to complete the step.

```
<proc id="babymap" style="BKM:(STYLE=BASE SEP=INLINE COMPACT)">
<titleblk><title>Baby Johnny is Crying</title></titleblk>
<procentry>Six-month old baby Johnny was sleeping
peacefully. Suddenly he began to cry.</procentry>
<procstep>
<proccmnd>
<desc>Check Johnny's diaper.</desc>
</proccmnd>
<decisionpnt>
<cond>Is the diaper wet?</cond>
<then><procstep><proccmnd>
<desc>Change the diaper.</desc>
</proccmnd><procexit>Johnny was uncomfortable.</procexit>
</procstep>
</then>
<else>
<desc>Continue at <xref refid="hungry">.</desc>
</else>
</decisionpnt>
</procstep><procstep id="hungry">
<decisionpnt>
<cond>Is Johnny hungry?</cond>
```

```

<then><procstep><decisionpnt>
<cond>Does Johnny have teeth?</cond>
<then><procstep><stepnotes><li>Johnny can eat solid
food.</li>
<li>Continue at <xref refid="frozstk"></li>
</stepnotes></procstep>
</then>
<else><procstep id="bottle"><proccmnd>
<desc>Warm a bottle.</desc>
</proccmnd><proccmnd>
<desc>Feed Johnny.</desc>
</proccmnd><procexit>Johnny needed a bottle.</procexit>
</procstep>
</else>
</decisionpnt></procstep>
</then>
<else><procstep><proccmnd>
<desc>Rock Johnny to sleep.</desc>
</proccmnd><procexit>Johnny was sleepy.</procexit>
</procstep>
</else>
</decisionpnt>
</procstep><procstep id="frozstk">
<proccmnd>
<desc>Thaw and broil a steak for Johnny. Include a
baked potato with butter and sour cream.</desc>
</proccmnd>
<procexit>Johnny was really hungry.</procexit>
</procstep></proc>

```

Using DecisionPnt for Outcome-Dependent Action Descriptions

The DecisionPnt element defines one or more condition/action (Then/Else) pairs that describe actions that must be completed under certain conditions.

```

<DECISIONPNT>
<COND>IS THE DIAPER WET?</COND>
<THEN>
<PROCSTEP>
<PROCCMND>
<DESC>CHANGE THE DIAPER.</DESC>
</PROCCMND>
<PROCEXIT>JOHNNY WAS UNCOMFORTABLE.</PROCEXIT>
</PROCSTEP>
</THEN>
<ELSE>
<DESC>CONTINUE AT <XREF REFID="HUNGRY">.</DESC>
</ELSE>
</DECISIONPNT>

```

Using RefKeys to Refer to Labels in a Graphic

The RefKey element contains a reference to a label in a graphic. When processed, this label provides a visual link to a spot in, for example, a graphic containing a chart or table.

```

:
<LI><P>The current 1995 Sales chart column
<REFKEY>4</REFKEY>
shows that sales
are up 10%, but operating expenses grew by 13.2%.</P>
</LI>

```

Using ProcExit to Complete a Procedure or Sub-Procedure

The ProcExit element contains the expected result of performing the task, and describes what to do after completing the procedure tasks.

```
<PROCSTEP>
  <PROCCMND>
    <DESC>ROCK JOHNNY TO SLEEP.</DESC>
  </PROCCMND>
  <PROCEXIT>JOHNNY WAS SLEEPY.</PROCEXIT>
</PROCSTEP>
```

Procedure Markup Examples

The examples that follow illustrates the use of procedure elements in IBMIDDoc.

Starting the Procedure

All Proc elements must contain a TitleBlk element, and a Desc element that contains a description of the procedure's purpose. The STYLE attribute on the Proc element in the example that follows specifies a value of STEPLIST

```
<PROC STYLE="steplist" id="proc1">
  <TITLEBLK><TITLE>Installing the ISDN Whantoozler</TITLE></TITLEBLK>
  <DESC>This procedure describes the steps one must perform or follow
    in order to successfully install the ISDN Whantoozler.
  </DESC>
  ⋮
  ⋮
  ⋮
</PROC>
```

Describing the Entry Point for the Procedure

The ProcEntry element contains the description of the entry point for the procedure. In the next portion of the example, the ProcEntry element contains RelProcs and PreReqProcs attributes, which reference related and prerequisite procedures. It also includes a prose description of the entry point for the procedure.

Note that the format in which related and prerequisite procedures are presented is style and processing dependent.

```
⋮
⋮
<PROCENTRY PREREQPROCS="proc1a" RELPROCS="proc3 proc2">
  This procedure assumes that you already have your
  ISDN line installed, and that there is no
  thunderstorm activity in the area.
</PROCENTRY>
⋮
⋮
⋮
```

Entering the Procedure Steps

Each procedure step is contained in a ProcStep element, which contains the title of the step and the step instructions. ProcStep may also contain:

- StepNotes, which allow you to make general notes about the step
- DecisionPnt which contain decision-making information for the step.

```
<PROCSTEP>
  <TITLEBLK><TITLE>Pre-Configuring ISDN Whantoozler</TITLE></TITLEBLK>
  <PROCCMND><DESC>Set the 4 DIP switches on the ISDN Whantoozler
    to correspond to the hemispheric location of your ISDN server.</DESC>
  </PROCCMND>
  <STEPNOTES>
    <LI>NA-ISDN Server: SW1234 ON ON ON ON</LI>
    <LI>SA-ISDN Server: SW1234 ON ON ON OFF</LI>
  </STEPNOTES>
</PROCSTEP>
```

```

<LI>EU-ISDN Server: SW1234 ON ON OFF OFF</LI>
<LI>AU-ISDN Server: SW1234 ON ON OFF ON</LI>
</STEPNOTES>
<DECISIONPNT>
<COND>Do you know your hemispheric location?</COND>
<THEN>
<PROCSTEP>
<PROCCMND><DESC>Continue to step <XREF REFID="nextstep">.</DESC>
</THEN>
<ELSE>
<PROCSTEP>
<PROCCMND><DESC>Find out the information and retry this step.</DESC>
</PROCSTEP>
</ELSE>
</DECISIONPNT>
</PROCSTEP>

```

Exiting the Procedure

The end of the procedure is described in the ProcExit element. You must include a description. You can also include the RECOVERYPROC attribute.

```

<PROCEXIT ID="pxita" RECOVERYPROC="rc1">
<P>The ISDN Whantoozler should have installed without problems.
The machine should have powered up
successfully. If so, you may continue to <XREF REFID="ok1">.
<P>If the machine smoked when you applied power,
see <XREF REFID="rc1"> for troubleshooting information.
</PROCEXIT>

```

Controlling Procedure Output Styles

Future Enhancement

Control of procedure output styles may be implemented in future release. These are presented for proposals only; they are not presently working. They are not slated for inclusion in any future release. If you need these sorts of output, please submit a requirement.

The default style for procedures is a MAP style. A typical MAP style output is illustrated in the formatted example that follows.

There is limited HTML and IPF support for Maintenance Analysis Procedures. When outputting to RTF, IPF, or Windows Help, MAPS/PROC become nested divisions. For hardcopy, the output is placed in a “flowchart” type mode.

The three proposed styles are :

Plaintext

results in a procedure with headings as the step numbers, and the step descriptions contained in paragraphs

Steplist

looks like an ordered list with Step 1, Step 2, Step 3, and so forth, as the numbering scheme.

1. This is the first step description.
2. This is the second step description.
3. This is the third step description.

Procedure exit description.

Table presents the procedure information in a table format, as shown in the following example.

Table 18. Test of Prereq and Coreq

Step	Description	Reference Keys
1	<p>This is the first step description. It contains several paragraphs of information.</p> <p>This is the procedure entry.</p> <p>This is some more information about the procedure.</p> <p>And here is even more information.</p>	
2	<p>This is the second step description. This step contains both a decision point and a step notes section.</p> <p>To continue:</p> <p>IF: The step worked.</p> <p>THEN: Continue to step 3.</p> <p>IF: The step did not work.</p> <p>THEN: Ensure that all cables are connected, and repeat the step again.</p> <p>Notes:</p> <ol style="list-style-type: none">1. This is the first note2. The second note.	
3	<p>This is the third step description. This step includes explicit reference key elements.</p>	A b c

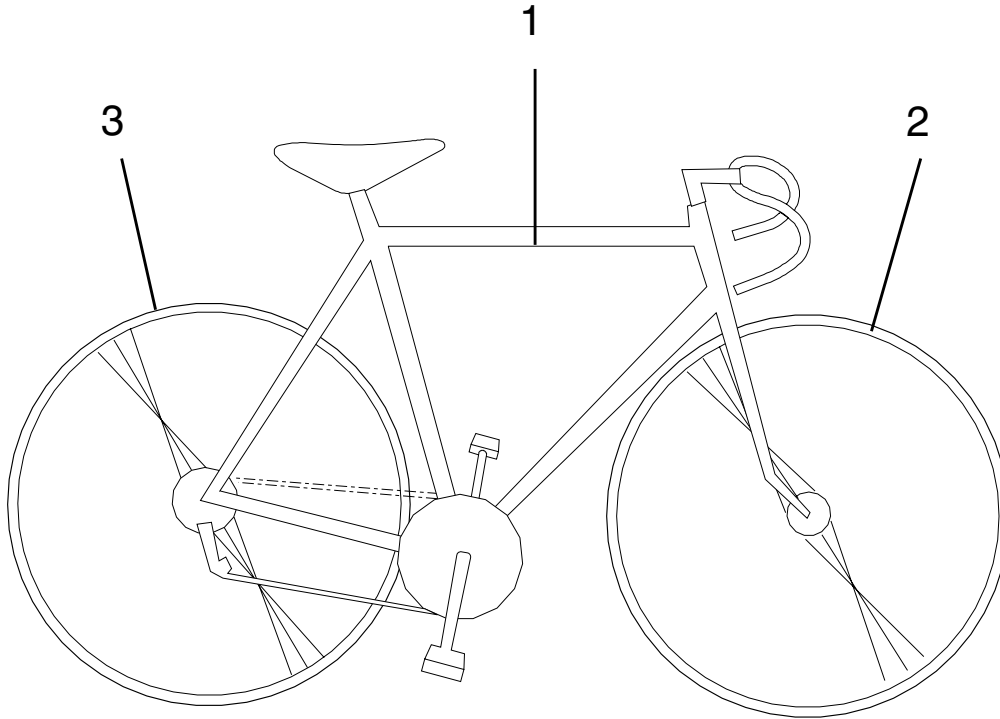
Proc Exit:

Procedure exit description.

Chapter 21. Creating parts catalog lists

A list of parts in a catalog is presented in conjunction with an illustration that shows where the parts fit in an **assembly** (a collection of parts that make up a unit of a machine or other product). The parts (also called **component items**) in the component list are keyed to numbered callouts on the artwork. IBMIDDoc provides an effective way to present both the component list in a parts catalog and the artwork associated with it. Component lists are usually presented in one of two ways: with the artwork on the top of a page followed by the list with the artwork showing the assembly on the left, (even-numbered page), and the list on the right, (odd-numbered page).

Assembly 1: Bicycle



Asm-Index	Part Number	Units	Description
1-1	4563423	1	Bike
-1	1230987	1	• Frame
-2	1238475	1	• Wheel assembly, front For detailed breakdown, see "Assembly 2: Wheel, front" on page 195.
-3	1234939	1	• Wheel assembly, rear

Markup source

```
<partasm id="bike" style="bkm:(layout=same)"><title>
Bicycle</title><mmobj><objref obj="bike">
<textalt>Bicycle</textalt>
</mmobj><compl>
<ci idxnum="1" partnum="4563423" upa="1">Bike</ci>
<compl>
<ci idxnum="1" partnum="1230987" upa="1">Frame</ci>
<ci idxnum="2" partnum="1238475" upa="1">Wheel assembly, front</ci>
<compcmt>For detailed breakdown, see <xref refid="wheel.xml">.</compcmt>
<ci idxnum="3" partnum="1234939" upa="1">Wheel assembly, rear</ci>
</compl>
</compl>
</partasm>
```

Creating the heading for a component list

Use the PartAsm (part assembly) tag to begin a component list. You need to enter a title for the assembly. You can get the heading of the bicycle example by entering these lines of markup:

```
<partasm id="bike" style="bkm:(layout=same)">
<title>Bicycle</title>
```

The formatter provides the following parts of the heading: the prefix, “Assembly” the number of the assembly, beginning with 1 and continuing with increments of 1 in succeeding assembly numbers the colon following the assembly number. “Bicycle” is the name we chose for our assembly. There are other things you can do with the PartAsm tag to make your component list easier to use; we used the BookMaster LAYOUT of SAME to tell BookMaster to place the artwork and as much of the component list as will fit on the same page.

A word about the artwork: Use the MMObj tag to include the drawing of your assembly. We use it after the Title tag. Here’s what it looked like in our bicycle assembly markup:

```
<mmobj><objref obj="bike">
<textalt>Bicycle</textalt>
</mmobj>
```

For more information about artwork, see “Including artwork in documents” on page 45.

Developing the component list

Now that we’ve discussed the beginning of a parts list, let’s take a look at the markup we used to create the component list for Assembly 1.

```
<compl>
<ci idxnum="1" partnum="4563423" upa="1">Bike</ci>
<compl>
<ci idxnum="1" partnum="1230987" upa="1">Frame</ci>
<ci idxnum="2" partnum="1238475" upa="1">Wheel assembly, front</ci>
<ci idxnum="3" partnum="1234939" upa="1">Wheel assembly, rear</ci>
</compl>
</compl>
```

Use the Compl (component list) tag to begin each component list. It has no attributes and no text, so it looks quite simple. When the formatter encounters the first Compl tag after a PartAsm tag, it supplies the column headings and the lines that make up the framework that encloses the catalog list and separates the columns.

Often an item in a component list is made up of other items. In order to show this hierarchy, you must nest them in your markup. That is, you must put component lists within other component lists. Component lists may be nested up to three deep. Remember, each component list must begin with its own Compl tag.

Use one CI (component item) tag for each item you want in your list. Whenever we use parts catalogs, we expect to find certain standard information, like a catalog number for ordering parts, a callout number so we can locate the part on the assembly drawing, and a number telling us how many of these items are in the assembly. We use attributes on the CI tag to add that information to our component lists, so the CI tag with all its attributes looks a lot more complicated than it really is. Here’s what one of the CI tags from our example looks like:

```
<ci idxnum="1" partnum="4563423" upa="1">Bike</ci>
```

Now let’s look at each of the attributes and how to use them:

ID Use the ID attribute when you need to identify a component item so that cross references can be made to it.

IDXNUM

Use the IDXNUM attribute to assign to the item a number that matches an

artwork index (callout) number. The number you assign with the IDXNUM attribute shows up in the “Asm-Index” column and is prefixed with a dash character. The number to the left of the dash is the same assembly number that the formatter used in the heading prefix.

PARTNUM

Use the PARTNUM attribute to assign the item’s part number. The number you assign with the PARTNUM attribute shows up in the “Part Number” column. Part numbers are limited to seven alphanumeric characters (A–Z, a–z, 0–9), with no intervening blanks.

UPA

Use the UPA (units per assembly) attribute to tell how many of this particular item there are in the assembly. The number you assign with the UPA attribute shows up in the “Units” column.

Including comments in the component list

You can use the CompCmt (component comment tag) to include helpful information that is not part of a component description. Just enter the text of your comment inside CompCmt tag. The comment text you enter appears indented in the “Description” column. Here’s what the CMT tag line from our front wheel example on page 438 looks like. This line also contains the CIREF tag that we’re going to discuss next. :cmt.For next higher assembly see :ceref refid=fwheel..

Cross-referencing part assemblies and component lists

Often we need to tell the readers of our component list where to find other related component lists or component items. Usually we want to point them to another assembly that shows a more detailed breakdown of a particular item. Sometimes we want to point them to another component that shows where a particular item fits in a larger assembly. In both cases, we must first use the ID attribute to identify the target (what we’re referring to), then use either the CIREF or the XREF tag to point to the target. Because we need another component list to show you how to refer from one component list to another, here’s the markup for a second assembly and component list.

```
<partasm id="wheelxmp" style="bkm:(layout=same)">
<title>Wheel, front</title><compl>
<ci partnum="56-2345">Wheel assembly, front</ci>
<compcmt>For next higher assembly see <xref refid="bike">.</compcmt>
<ci idxnum="1" partnum="33-5234" upa="1">Tire, clincher 27 x 1.125</ci>
<ci idxnum="2" partnum="56-4352" upa="1">Tube, 27 x 1.125</ci>
<ci idxnum="3" partnum="56-3489" upa="1">Rim liner</ci>
<ci idxnum="4" partnum="56-6534" upa="1" id="wheel2a">Wheel assembly</ci>
<compl>
<ci idxnum="5" partnum="56-3476" upa="1">Rim, aluminum alloy 27 x 1.125</ci>
<ci idxnum="6" partnum="56-8393" upa="36">Spoke, 298mm</ci>
<ci idxnum="7" partnum="56-9845" upa="36">Spoke bolt</ci>
</compl>
<ci idxnum="8" partnum="56-9874" upa="1">Hub assembly, front</ci>
</compl></partasm>
```

Assembly 2: Wheel, front

Asm-Index	Part Number	Units	Description
2-	56-2345		Wheel assembly, front For next higher assembly see "Assembly 1: Bicycle" on page 192.
-1	33-5234	1	Tire, clincher 27 x 1.125
-2	56-4352	1	Tube, 27 x 1.125
-3	56-3489	1	Rim liner
-4	56-6534	1	Wheel assembly
-5	56-3476	1	• Rim, aluminum alloy 27 x 1.125
-6	56-8393	36	• Spoke, 298mm
-7	56-9845	36	• Spoke bolt
-8	56-9874	1	Hub assembly, front

Keeping track of assemblies and parts

The AsmList (assembly list) tag and the PNIndex (part number index) tag help you find each assembly and part.

Getting an assembly list

The AsmList tag works much like a partial table of contents; it gives you an alphabetical listing of the headings from your PartAsm tags, along with the page numbers on which their headings appear.

If you want to put your assembly list in the front matter of your document, here is how you might enter your markup:

```
<frontm style="display='tipage cover'">
<toc><gendtitle></toc>
<d>
<dprolog><titleblk>
<title>List of assemblies</title>
</titleblk></dprolog>
<dbody>
<asmlist>
</dbody></d>
</frontm>
```

You can put your assembly list at the beginning of a chapter instead of in the front matter, like so:

```
<d>
<dprolog><titleblk>
<title>Parts catalog</title>
</titleblk></dprolog>
<dbody>
<p>This portion contains the parts and assembly instructions
for your Mark-21 Super Bi-Pedal Tricycle.</p>
<d>
<dprolog><titleblk>
<title>List of assemblies</title>
</titleblk></dprolog>
<dbody>
<asmlist>
</dbody></d>
<partasm>
...
</partasm></dbody></d>
```

But wherever you put it, remember, only one AsmList tag is allowed per document.

Note: ASMLIST is not supported in the HTML output transform.

Getting a part number index

An index of part numbers can help in retrieving individual parts in your document. If you enter the PNIndex tag in the back matter of a document, the formatter sorts the part numbers you entered with the CI tag's PartNum attribute and prints them and their page numbers.

Part numbers from sample parts assemblies are excluded from the part number index. The part number sort sequence is different than that of the regular index; all one-digit part numbers are listed, followed by all two-digit part numbers, and so on. Here's the markup used to get the part number index in this book.

```
<pnindex id="partnumindex">  
<gendtitle>  
</pnindex>
```

Part 3. IBMIDDoc Markup Reference

Chapter 22. Reference Explanation	201	+ DBody (division body)	246
Element and Attribute Descriptions	201	+ Dec (decimal number)	247
How to Read the Syntax Diagrams	201	DecisionPnt (decision point)	247
Common Element Attributes (large set).	204	+ Defn (definition of a term)	249
Common Element Attributes (small set)	205	+ DefnHd (definition description heading)	250
 Chapter 23. IBMIDDoc Elements	207	+ Delim (syntax delimiter).	250
+ Abbrev (abbreviations)	207	+ Desc (element description)	251
+ Abstract (abstract).	207	+ DIntro (division introduction)	252
+ Address (address)	208	+ DL (definition list).	253
+ Annot (annotation)	209	+ DLBlk (definition list block)	254
+ AnnotBody (annotation body).	210	+ DLEntry (definition list entry)	255
+ APL (APL data)	211	+ DocTitle (document title)	256
+ Appendix	212	+ DProlog (division prolog)	256
+ Approvers (document approvers).	212	+ DSum (division summary)	257
+ AreaDef (defines graphic hot spot area)	213	+ DVCFObj (DVCF Migration Element)	258
+ AsmList (list of parts assemblies).	214	+ EdNotices (edition notices)	259
+ Attention (safety notice).	214	Else (other procedure path to follow)	259
+ Author	215	+ Entry (table entry).	260
+ Authors	216	+ ExternalFileName	262
+ BackCover (back cover)	217	+ Fig (figure)	262
+ BackM (back matter)	217	+ FigList (list of figures)	263
+ BibEntry (bibliographic entry)	218	+ FigSeg (figure segment)	264
+ BibEntryDefs (contains bibliographic entries)	219	+ FileNum (file number)	265
+ Bibliog (bibliography)	219	+ Fn (footnote)	265
+ BibList (bibliography entry list)	220	+ FNList (footnote list)	266
+ Bin (binary data)	221	+ Formula (math formula).	267
+ Body (document body)	222	+ Fragment (syntax fragment)	268
+ BOFNum (bill of forms number)	222	+ FragRef (syntax fragment reference)	269
+ Bridge (bridge between concepts).	223	FrontCover	270
+ Cap (caption)	225	+ FrontM (front matter).	270
+ Caution (caution notice).	225	+ GendTitle (default title specification)	272
+ CGraphic (character graphic)	226	+ GL (glossary list)	272
+ Char (character data)	227	+ GLBlk (glossary list block)	274
+ CI (component item)	228	+ GLDefs (glossary definitions)	275
+ Cit (document citation)	229	+ GLEntry (glossary list entry)	275
+ ClassDef (element class definition)	230	+ Glossary	276
CLE (content list entry)	231	+ Group	277
+ Code (message code number)	232	+ Hex (hexadecimal).	277
+ ColSpec (column specification)	232	+ IBMBibEntry (IBM bibliographic entry).	279
+ CompCmt (component comment)	234	+ IBMBOFNum (bill of forms number)	280
+ CompL (component list).	234	+ IBMDocNum (IBM document number).	280
Cond (procedure result)	235	+ IBMFeatNum (IBM feature number).	281
+ ContainedDocs (documents in IBMLibEntry and		IBMIDDoc (IBM-specific product documentation)	281
LibEntry).	236	IBMLibEntry (IBM document library definition)	287
CopyR (copyrights)	237	IBMMail (IBMMail e-mail address)	289
CopyRDefs (copyright definitions)	237	+ IBMPartNum (IBM part number).	290
+ Corp (enterprise name and address).	238	+ IBMPgmNum (IBM program number)	290
+ CorpName (corporation name)	239	+ IBMProdInfo (IBM product information)	291
+ CoverDef (cover definition).	239	+ IBMSafety (IBM safety notices)	291
+ CritDate (critical date for a document)	240	+ IdxDefs (central index entries).	292
+ CritDates (set of critical dates).	240	+ IdxTerm (index term).	293
+ D (hierarchical division).	242	+ Index	293
+ Danger (danger notice)	243	+ Internet (internet e-mail address).	294
+ Date	244	+ IRef (index entry reference).	294
+ DBlk (Division block).	245	+ ISBN (document ISBN number)	295
		+ I1 (primary index entry).	296

+ I2 (secondary index entry)	297	ObjLibBody (object library body)	367
+ I3 (tertiary index entry)	298	+ ObjRef (object reference).	368
Kwd (syntax keyword)	299	+ Oct (octal number)	369
+ L (explicit link)	300	+ OL (ordered list)	370
LDescs (link descriptions)	302	Oper (syntax operator)	371
+ LE (language element)	302	OrderNum (order number)	372
LeDesc (language element description)	304	+ OrigIBMDocNum (original IBM document number)	372
+ LEDI (language element description item)	304	+ Owners	373
+ Legend	306	+ P (paragraph)	374
+ LEN (language element name).	307	+ Parm (parameter list entry).	375
+ LERS (language element reference section)	308	+ ParmBlk (parameter list block)	376
+ LERSDef (LERS property definition).	310	+ ParmL (parameter list)	376
+ LI (list item)	311	+ Part (major document part).	378
+ LibEntry (document library definition)	312	+ PartAsm (part assembly)	379
+ LIBlk (list item block)	314	+ PartAsmSeg (part assembly segment)	380
+ Library	314	+ PBlk (paragraph block)	380
+ Lines (text with line boundaries)	315	+ Person (person's name and address).	381
+ Litdata (literal data)	316	+ Ph (Phrase)	382
+ LQ (stand-alone quotation)	318	+ Phone (telephone number)	384
+ Maintainer (reader comment)	320	+ PK (programming keyword)	385
+ Mark (marked note definition).	320	+ PNIndex (part number index)	386
+ MarkList (marked note list).	321	+ PostalCode (postal or zip code)	387
+ MasterIndex (master index).	323	+ Preface	387
+ MasterIndexInfo (master index information)	324	+ Proc (procedure)	388
+ MasterIndexObj (master index object)	324	+ ProcCmnd (procedure command).	389
+ MasterIndexPrefix (master index prefix)	325	+ ProcEntry (procedure entry point)	390
+ MD (marked deletion)	326	+ ProcExit (procedure exit point)	391
+ MkAction (marked note action definition)	327	+ ProcIntro (procedure introduction)	391
+ MkClass (marked note class definition).	328	+ ProcStep (procedure step)	392
MkDesc (mark description).	329	+ ProcSumm (procedure summary).	393
MkNote (marked note)	331	+ ProcSummItem (procedure summary item)	393
+ MMObj (multi-media object; artwork)	333	+ ProdInfo (product information)	393
+ MMObjLink (multi-media object link)	335	+ ProdName (product name)	394
Mod (information module)	336	+ Prolog (document metainformation)	395
ModInfo (modular information)	338	+ PropDef (property set definition)	395
ModInfoDef (modular information property definition)	340	+ PropDefs (property definitions)	396
ModItemDef (item class definitions)	341	+ PropDesc (property description)	397
ModDesc (modular content description)	343	+ PropGroup (property group)	397
ModItem (module description item)	344	+ PrtLoc (country where printed)	398
+ ModLvl (modification level)	346	+ PublicID (public identifier)	399
ModName (modular information element name)	346	+ Publisher (document publisher)	400
+ Msg (message or code description)	348	+ PV (parameter variable)	400
+ MsgItem (message description item).	348	+ Q (quotation phrase)	402
+ MsgItemDef (definition of message description items)	350	+ Qualif (qualification)	403
+ MsgList (list of message or code descriptions)	352	+ QualifDefs (qualification definitions)	404
+ MsgNum (message identifier)	353	+ RCF (reader comment form)	404
MsgText (message text)	354	+ RefKey (reference key)	405
+ MV (message variable)	355	+ Release (product release identifier)	406
+ Name (person's name)	357	+ RepSep (syntax repeat separator)	406
+ NameLoc (named location).	357	+ RetKey (retrieval key)	407
+ NItem (notice item)	359	+ Rev (revision)	408
NMList (named list of IDs or entities)	360	+ RevDefs (revision tracking information)	409
+ Note	362	+ Row (table row)	409
+ NoteBody (note body)	362	+ Safety (safety notices).	411
+ NoteList (ordered note list).	363	+ Screen (display screen)	411
+ Notices (contains notices)	364	+ Sem (semantic meaning).	412
Notloc (notation-specific location)	365	+ Sep (syntactic separator).	412
+ Num (number)	366	+ SOA (summary of amendments)	413
+ ObjLib (object library)	367	+ SpanSpec (span specification)	414
		+ SpecDProlog (special section division prolog).	415
		+ StepNotes (step notes)	416

+ StepRef (procedure step reference)	416
+ STitle (shortened title)	416
+ SubTitle (descriptive subtitle)	417
+ SynBlk (syntax block).	417
+ SynNote (syntax note)	418
+ SynPh (syntax phrase)	419
+ Syntax (syntax diagram).	420
+ Table	423
+ TBody (table body)	424
+ Term	425
+ TermHd (term heading)	426
+ TextAlt (text alternative).	427
+ TFoot (table footer)	427
+ TGroup (table group).	428
+ THead (table heading)	429
+ Then (procedure action to take)	430
+ Title	431
TitleBlk (title information)	432
+ TList (list of tables)	432
+ TM (Trademark)	433
+ TOC (table of contents)	435
+ UL (unordered list)	436
+ Var (syntax variable)	437
+ Version (product version number)	437
+ VNet (IBM VNet mail address)	438
+ VolId (volume identifier)	438
+ Warning (warning notice)	439
+ WebPage	440
+ Xmp (example).	440
+ XPh (example phrase)	441
+ XRef (cross reference).	442

Chapter 22. Reference Explanation

This chapter lists the type of information that is provided for each element or attribute in “Chapter 23. IBMIDDoc Elements” on page 207 and describes how to read the syntax diagrams.

Element and Attribute Descriptions

The elements and attributes are listed in alphabetical order. For each element or attribute, the following information is provided:

Name The name and a short description of the element or attribute.

Purpose

The purpose of the element or attribute.

Examples

One or more examples showing how the element or attribute is used.

Attributes and contained elements

Descriptions of attributes, contained elements, and attribute values.

Usage Description of how to use the element or attribute.

Contexts

A list of the elements that can directly contain the element or have the attribute, or a description of where the element or attribute can be used.

How to Read the Syntax Diagrams

This section describes how to read and use the syntax diagrams, which define the rules for typing element markup in a text-editing environment such as XEDIT or EPM. For more information about markup, see “Markup Rules” on page 11.

Note: The syntax diagrams do not show contained elements that have omissible start and end tags. However, when you use an SGML editor, you usually see all the elements, including any omissible elements.

- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads (>>) and ends on the right with two arrowheads facing each other (<<).

▶▶ | Syntax Diagram | ◀◀

- If a diagram is longer than one line, each line to be continued ends with a single arrowhead (>) and the next line begins with a single arrowhead.

▶▶ | First Line | ▶

▶▶ | Middle Line | ▶

►► | Last Line | ◄◄

- A word in all uppercase is an operand or value you must spell exactly as shown. However, you can enter it using any case.

►►—OPERAND—◄◄

If an operand or value can be abbreviated, the abbreviation is discussed in the text associated with the syntax diagram.

- A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

►►—*variable*—◄◄

- Single-word attribute values are not shown with quotation marks (but any attribute value can be entered with quotation marks around it). Multiple-word attribute values and any attribute value that contains special characters must be enclosed in quotation marks. Quotation marks are always shown as double quotation marks ("), but single quotation marks (') can be used unless the value contains single quotation marks or an apostrophe. For more information about markup with quotation marks, see "Markup Rules" on page 11.
- Required operands and values appear on the main path line. You must code required operands and values.

►►—REQUIRED_OPERAND—◄◄

If several mutually exclusive required operands or values exist, they are stacked vertically in alphanumeric order.

►►—
REQUIRED_OPERAND_OR_VALUE_1
REQUIRED_OPERAND_OR_VALUE_2—◄◄

- Optional operands and values appear below the main path line. You can choose not to code optional operands and values.

►►—
OPERAND—◄◄

If several mutually exclusive optional operands or values exist, they are stacked vertically in alphanumeric order below the main path line.

►►—
OPERAND_OR_VALUE_1
OPERAND_OR_VALUE_2—◄◄

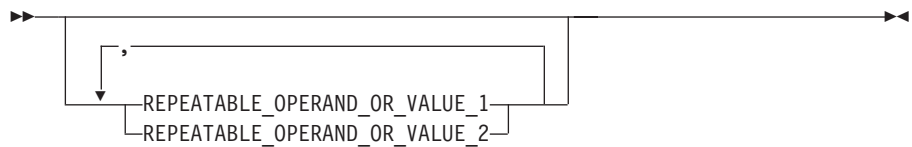
- Default operands and values appear above the main path line. If you omit the operand entirely, the default is used.



- An arrow returning to the left above an operand or value on the main path line means that the operand or value can be repeated. The comma means that each operand or value must be separated from the next by a comma. If a space is shown, each operand or value must be separated from the next by a space.



- An arrow returning to the left above a group of operands or values means that more than one can be selected or that a single one can be repeated.



- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.



Notes:

1 An example of a syntax note.

- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram. The fragment is placed either below the main diagram or in a separate description.



Syntax Fragment:



Common Element Attributes (large set)

Several elements are defined to use this set of attributes:

Class

The Class attribute associates an element class with an element. This attribute must contain an SGML name that has been defined as a class name in a ClassDef element. This attribute only applies to elements specified on the ClassDef's ELETYPE attribute. Element classes are defined with ClassDef elements within PropDefs. See "Chapter 19. Property and Class Definition" on page 177 for more information.

Conloc

The CONLOC attribute specifies that the content of another element of the same type is to be used as the content of the referencing element. This enables reuse of information. When the CONLOC attribute is specified, you cannot specify the element's end tag. The result of using CONLOC is exactly the same as if the element being referred to had occurred at that point in the document. See "Reusing elements from an object library" on page 166.

+ Attributes on the element are now passed to the element with the CONLOC;
+ this is a feature that began with IDWB release 3.4, patch IDWXF036.

HyTime

ignored by processes

| **ID** The ID attribute identifies an element within an SGML document. IDs must be
| unique within a single document. Any element that has an ID can be
| cross=referenced or linked to. IDs can be up to 64 characters long. IDs must
| start with an alphabetic character and can contain letters, numbers, dashes (-),
| or periods (.).

InfoMast

A fixed attribute used to classify the element.

Props

The Props attribute is used to specify the condition under which the information contained within the element appears. See "Property-Based Retrieval" on page 171.

PropSrc

Points to an element whose properties are to be used as the properties of the referencing element. See "Chapter 19. Property and Class Definition" on page 177.

Qualif

The QUALIF attribute refers to the ID of a qualification element. See "Qualifying information" on page 41.

Reftype

ignored by processes

RetAlts

The RETALTS attribute points to one or more elements whose content may be used in place of, or in addition to, the referencing element's content. This attribute must reference one or more elements of the same element type. This attribute usually references elements in an object library. See "Retrieval alternatives" on page 174.

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See “Using Revisions” on page 91.

Status

ignored by processes

Style

The Style attribute contains either a reference to a separate style specification for an element or a set of element-specific presentation style definitions when the processing system does not support separate styles for elements. Assigning a value to the STYLE attribute modifies how an element is presented when it is output.

XrefText

The XrefText attribute defines the text to be used when an element is the target of a link that generates a reference. See “Chapter 6. Cross-referencing” on page 51.

Common Element Attributes (small set)

Several elements are defined to use this set of attributes:

Class

The Class attribute associates an element class with an element. This attribute must contain an SGML name that has been defined as a class name in a ClassDef element. This attribute only applies to elements specified on the ClassDef’s ELETYPE attribute. Element classes are defined with ClassDef elements within PropDefs. See “Chapter 19. Property and Class Definition” on page 177 for more information.

ID The ID attribute identifies an element within an SGML document. IDs must be unique within a single document. Any element that has an ID can be cross-referenced or linked to. IDs can be up to 64 characters long. IDs must start with an alphabetic character and can contain letters, numbers, dashes (-), or periods (.).

Props

The Props attribute is used to specify the condition under which the information contained within the element appears. See “Property-Based Retrieval” on page 171.

PropSrc

Points to an element whose properties are to be used as the properties of the referencing element. See “Chapter 19. Property and Class Definition” on page 177.

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See “Using Revisions” on page 91.

Status

ignored by processes

Style

The Style attribute contains either a reference to a separate style specification for an element or a set of element-specific presentation style definitions when the processing system does not support separate styles for elements. Assigning a value to the STYLE attribute modifies how an element is presented when it is output.

HyTime

ignored by processes

InfoMast

A fixed attribute used to classify the element.

Reftype

ignored by processes

Chapter 23. IBMIDDoc Elements

This section describes the elements and attributes in the IBMIDDoc language.

Abbrev (abbreviations)

Purpose

The Abbrev element is a special division, and contains an explanation of abbreviations used in the document. The best way to create a list of abbreviations is to use the DL element.

Examples

```
<abbrev>
<specdprolog><gendtitle></specdprolog>
<dbody>
<dl>
<dlentry><term>IBMIDDoc</term>
<defn>IBMIDDoc is the name of IBM's implementation
of the SGML standard for software documentation.</defn>
</dlentry>
</dl>
</dbody></abbrev>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Special sections" on page 88.

Contexts

Children: "DBody (division body)" on page 246, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "SpecDProlog (special section division prolog)" on page 415.

Parents: "BackM (back matter)" on page 217, "FrontM (front matter)" on page 270.

Abstract (abstract)

Purpose

The Abstract special division element contains a short description of the content of the document. Use Abstract to contain a brief description of the document.

Examples

```
<abstract>
<specdprolog><gendtitle></specdprolog>
<dbody>
<p>This describes how to...</p>
</dbody></abstract>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Special sections” on page 88.

Contexts

Children: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “SpecDProlog (special section division prolog)” on page 415.

Parents: “D (hierarchical division)” on page 242, “FrontM (front matter)” on page 270, “Part (major document part)” on page 378.

Address (address)

Purpose

The Address element contains the address of a person or corporation. Address is normally used within the context of an Author element but may be used elsewhere. Enter the address text in the form you want it to be displayed. The text will not be reflowed when the markup is processed. Contained elements will be processed according to the styles used by the processing application.

Examples

```
<authors>
<author><person>
<name>Fred Mertz</name>
<address>
125 West Hollywood Blvd
Tinseltown, CA <postalcode>90210</postalcode></address>
</person></author>
</authors>
```

Attributes

OBJ=file-entity-name

The OBJ attribute names a file entity that contains the text for the address.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Author and Address” on page 77.

Contexts

Children: text (#pcdata), “IBMMail (IBMMail e-mail address)” on page 289, “Internet (internet e-mail address)” on page 294, “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Phone (telephone number)” on page 384, “PostalCode (postal or zip code)” on page 387, “Term” on page 425, “TM (Trademark)” on page 433, “VNet (IBM VNet mail address)” on page 438, “WebPage” on page 440.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Corp (enterprise name and address)” on page 238, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Person (person’s name and address)” on page 381, “Ph (Phrase)” on page 382, “Publisher (document publisher)” on page 400, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

Annot (annotation)

Purpose

Use Annot to annotate the content of its containing element, such as notes to reviewers or editors. Annotations can be presented or suppressed, depending on the options given to the processing system.

Migration Notes

- Annot cannot be used to comment out information.
- Unlike the BookMaster Annot element, Annot cannot span document structures. It is an element in the document hierarchy, like any other element.

Examples

```
<P>This text is a paragraph before the annotation.</p>
<ANNOT>
<ANNOTBODY>
<P>This is an annotation, the first paragraph.
<P>This is the second paragraph of the annotation
</ANNOTBODY>
</ANNOT>
<P>This is a paragraph after the annotation.</p>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Annotations” on page 41.

Contexts

Children: “AnnotBody (annotation body)”, “Title” on page 431.

Parents: “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “LEdi (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

AnnotBody (annotation body)

Purpose

Use AnnotBody to contain the body of the annotation; see “Annot (annotation)” on page 209.

Examples

```
<P>This text is a paragraph before the annotation.</p>
<ANNOT>
<ANNOTBODY>
<P>This is an annotation, the first paragraph.
<P>This is the second paragraph of the annotation
</ANNOTBODY>
</ANNOT>
<P>This is a paragraph after the annotation.</p>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Annotations” on page 41.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “APL (APL data)” on page 211, “Attention (safety notice)” on page 214, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Danger (danger notice)” on page 243, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD

(marked deletion)" on page 326, "MkNote (marked note)" on page 331, "MMObj (multi-media object; artwork)" on page 333, "ModInfo (modular information)" on page 338, "MV (message variable)" on page 355, "Note" on page 362, "NoteList (ordered note list)" on page 363, "Num (number)" on page 366, "Oct (octal number)" on page 369, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "Ph (Phrase)" on page 382, "PK (programming keyword)" on page 385, "PV (parameter variable)" on page 400, "Q (quotation phrase)" on page 402, "RefKey (reference key)" on page 405, "Screen (display screen)" on page 411, "SynPh (syntax phrase)" on page 419, "Syntax (syntax diagram)" on page 420, "Table" on page 423, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "Annot (annotation)" on page 209.

APL (APL data)

Purpose

Use the APL element to identify data that is part of an APL program. The content of the element may be APL data or other elements that make sense in the APL context. This data is encoded in the document source using the character encoding used for APL data and programs, not necessarily the character encoding used for the data everywhere else in the document. The content of this element is typically presented using the same font as is conventionally used for APL, which will also probably differ from that used to present the other data found in the document. An external entity containing the APL data may be referred to using the OBJ attribute, which must contain the name of a data entity. Character entities can also be used to represent APL characters.

Examples

```
<p>A matrix is defined with the string:  
<APL>2 3p1 2 3 4 5 6</APL></p>
```

Attributes

OBJ=*file-entity-name*

The name of the file entity that contains the APL data.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Highlighting" on page 35.

Contexts

Children: text (#pcdata).

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "CompCmt (component comment)" on page 234, "Danger (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "Entry (table entry)" on page 260, "Fn (footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on page 318

page 317, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439.

Appendix

Purpose

The Appendix element contains division-like elements that are to be considered appendixes. Appendixes are not considered part of the content of the main body of the SGML markup. They usually contain reference information. In the default presentation style, appendixes are numbered with letters rather than digits.

Examples

```
<backm>
<appendix>
<d>
<dprolog><titleblk>
<title>Special stuff</title>
</titleblk></dprolog>
<dbody></dbody></d>
</appendix></backm>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using appendix” on page 88.

Contexts

Children: “D (hierarchical division)” on page 242, “DBlk (Division block)” on page 245, “LERS (language element reference section)” on page 308, “ModInfo (modular information)” on page 338, “MsgList (list of message or code descriptions)” on page 352, “PartAsm (part assembly)” on page 379, “Proc (procedure)” on page 388, “RetKey (retrieval key)” on page 407.

Parents: “BackM (back matter)” on page 217.

Approvers (document approvers)

Purpose

Approvers contains the elements that identify the person or organization who must approve a document or division for publication.

Examples

```
<approvers>
<person><name>Ethel Mertzt</name></person>
</approvers>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “Corp (enterprise name and address)” on page 238, “Person (person’s name and address)” on page 381.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

AreaDef (defines graphic hot spot area)

Purpose

The AreaDef element contains the specifications of a graphic hot spot. The geometry of graphic hot spots is specified according to the shape of the hot spot. The numbers specified represent pels in the bitmap for bitmaps, and represent quanta in the underlying grid used in specifying points in a vector graphic. Multiple AreaDef elements can be used in a single MMObjLink element to indicate that more than one area in the graphic can be used to invoke the link.

Examples

```
<mmobj><objref obj="bear"><mmobjlink linkend="a">  
<areadef shape="circle" coords="10 15 20"></areadef>  
</mmobjlink>  
<textalt>One teddy bear.</textalt>  
</mmobj>
```

Attributes

SHAPE = rectangle | circle | polygon

describes the shape of the graphic hot spot.

COORDS = numbers

contains the coordinates for the graphic hot spot. Values are blank delimited.

rectangle

lower left *x*, lower left *y*, upper right *x*, upper left *y*.

circle

center *x*, center *y*, radius, with center specified relative to the origin of the graphic.

polygon

1st *x*, 1st *y*, 2nd *x*, 2nd *y*, Nth *x*, Nth *y* with automatic closure if the first and last point are not identical. It must be an error if a line drawn between any two adjacent points intersects with any other line drawn between any other two adjacent points in the specification.

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “TextAlt (text alternative)” on page 427.

Parents: “LDescs (link descriptions)” on page 302, “MMObjLink (multi-media object link)” on page 335.

AsmList (list of parts assemblies)

Purpose

The AsmList element is a specialized list element that contains a list of all parts assembly lists in the document.

Examples

<ASMLIST>

Attributes

SPEC=AUTO|MAN

This attribute has a fixed value of AUTO, generate the list from the assemblies in the document.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Getting an assembly list” on page 195.

Contexts

Children: empty.

Parents: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “LEDI (language element description item)” on page 304, “MsgItem (message description item)” on page 348, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391.

Attention (safety notice)

Purpose

Use an Attention notice to indicate the possibility of damage to a program, device, system, or data.

Examples

<attention>Here's a way to get someone's attention.
</attention>

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “The perils of processing: Attention, caution, and danger” on page 40.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition

list)" on page 253, "Fig (figure)" on page 262, "Formula (math formula)" on page 267, "GL (glossary list)" on page 272, "Hex (hexadecimal)" on page 277, "L (explicit link)" on page 300, "Lines (text with line boundaries)" on page 315, "Litdata (literal data)" on page 316, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "MMObj (multi-media object; artwork)" on page 333, "ModInfo (modular information)" on page 338, "MV (message variable)" on page 355, "Num (number)" on page 366, "Oct (octal number)" on page 369, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "Ph (Phrase)" on page 382, "PK (programming keyword)" on page 385, "PV (parameter variable)" on page 400, "Q (quotation phrase)" on page 402, "RefKey (reference key)" on page 405, "Screen (display screen)" on page 411, "SynPh (syntax phrase)" on page 419, "Syntax (syntax diagram)" on page 420, "Table" on page 423, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "AnnotBody (annotation body)" on page 210, "Bridge (bridge between concepts)" on page 223, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "Entry (table entry)" on page 260, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "PBlk (paragraph block)" on page 380, "ProcEntry (procedure entry point)" on page 390, "ProcIntro (procedure introduction)" on page 391, "Safety (safety notices)" on page 411.

Author

Purpose

Use Author to contain information about an author, such as name, title, and so forth.

Examples

```
<authors>
<author><person>
<name>Fred Mertz</name>
<address>125 West Hollywood Blvd
Tinseltown, CA <postalcode>90210</postalcode></address>
</person></author>
</authors>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See "Author and Address" on page 77.

Contexts

Children: "Corp (enterprise name and address)" on page 238, "Person (person's name and address)" on page 381, "Title" on page 431.

Parents: "Authors" on page 216, "Rev (revision)" on page 408.

Authors

Purpose

The Authors element contains information about one or more authors of the document.

Examples

```
<authors>
  <author><person>
    <name>Fred Mertz</name>
    <address>125 West Hollywood Blvd
    Tinseltown, CA <postalcode>90210</postalcode></address>
  </person></author>
</authors>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Author and Address” on page 77.

Contexts

Children: “Author” on page 215, “Desc (element description)” on page 251.

Parents: “BibEntry (bibliographic entry)” on page 218, “DProlog (division prolog)” on page 256, “IBMBibEntry (IBM bibliographic entry)” on page 279, “SpecDProlog (special section division prolog)” on page 415.

BackCover (back cover)

Purpose

The BackCover element contains a reference to the art used for the document's back cover.

Examples

```
<ibmbibentry><doctitle><titleblk>
<title>My Document</title>
</titleblk></doctitle>
<ibmdocnum></ibmdocnum>
<coverdef><frontcover><mmobj><objref obj="front1">
<textalt></textalt></mmobj></frontcover>
<backcover><mmobj><objref obj="back1">
<textalt></textalt></mmobj></backcover>
</coverdef></ibmbibentry>
```

Usage

See "Adding to the front or back cover (CoverDef)" on page 78.

Contexts

Children: "BibList (bibliography entry list)" on page 220, "CGraphic (character graphic)" on page 226, "DL (definition list)" on page 253, "Lines (text with line boundaries)" on page 315, "Litdata (literal data)" on page 316, "MMObj (multi-media object; artwork)" on page 333, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "Table" on page 423, "UL (unordered list)" on page 436, "Xmp (example)" on page 440.

Parents: "CoverDef (cover definition)" on page 239.

BackM (back matter)

Purpose

The BackM element contains the material that follows the body of a document. It may include appendixes, a glossary, and an index. The output transforms automatically provide a part separator for the back matter when the body of the document contained a Part tag. If you prefer want to suppress this part separator, use the following coding on the Backm tag:

```
<backm style="xpp:(nopart)">
```

Examples

```
<BACKM>
<APPENDIX>
<D>
<DPROLOG>
<TITLEBLK>
<TITLE>WHANTOOZLER TECHNICAL SPECIFICATIONS
</TITLE>
</TITLEBLK>
</DPROLOG>
<DBODY>
<P>THIS SECTION DESCRIBES THE ELECTRICAL CONFIGURATION
OF THE.....
</P>
```

```
</DBODY>
</D>
</APPENDIX>
</BACKM>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “About back matter (BackM)” on page 88.

Contexts

Children: “Abbrev (abbreviations)” on page 207, “Appendix” on page 212, “Bibliog (bibliography)” on page 219, “D (hierarchical division)” on page 242, “DBlk (Division block)” on page 245, “Glossary” on page 276, “Index” on page 293, “MasterIndex (master index)” on page 323, “PNIndex (part number index)” on page 386, “RCF (reader comment form)” on page 404, “SOA (summary of amendments)” on page 413.

Parents: “IBMIDDoc (IBM-specific product documentation)” on page 281.

BibEntry (bibliographic entry)

Purpose

The BibEntry element contains information about a document. The IBMBibEntry element is used to define bibliographic entries for IBM documents. You can use this to create bibliographic information for non-IBM bibliographies and title citations.

Examples

To create a bibliography definition:

```
<bibentrydefs><bibentry docname="dislike"><doctitle>
<titleblk><title>Things I Dislike</title></titleblk>
</doctitle></bibentry></bibentrydefs>
```

Attributes

DOCLINK=*ID*

The DocLink attribute specifies the ID of the URL defined on a Notloc element.

DOCNAME=*entity_name*

Contains a reference to the ID or name of an entity that is defined in the document that must also be referenced by a NameLoc element. This indicates a cross-document target with the specified ID value.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 13. Bibliographies and citations” on page 119.

Contexts

Children: “Authors” on page 216, “Desc (element description)” on page 251, “DocTitle (document title)” on page 256, “ExternalFileName” on page 262, “ISBN

(document ISBN number)" on page 295, "OrderNum (order number)" on page 372, "PrtLoc (country where printed)" on page 398, "PublicID (public identifier)" on page 399, "Publisher (document publisher)" on page 400.

Parents: "BibEntryDefs (contains bibliographic entries)", "BibList (bibliography entry list)" on page 220, "Cit (document citation)" on page 229.

BibEntryDefs (contains bibliographic entries)

Purpose

Use the BibEntryDefs element to contain BibEntry and LibEntry elements. When used in a DProlog element, BibEntryDefs contains elements used within that division. When used in the Prolog element, BibEntryDefs contains elements used in the document.

Examples

To create a bibliography definition:

```
<bibentrydefs><bibentry docname="dislike"><doctitle>
<titleblk><title>Things I Dislike</title></titleblk>
</doctitle></bibentry></bibentrydefs>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See "Chapter 13. Bibliographies and citations" on page 119.

Contexts

Children: "BibEntry (bibliographic entry)" on page 218, "IBMBibEntry (IBM bibliographic entry)" on page 279, "IBMLibEntry (IBM document library definition)" on page 287, "LibEntry (document library definition)" on page 312.

Parents: "DProlog (division prolog)" on page 256, "Prolog (document metainformation)" on page 395, "SpecDProlog (special section division prolog)" on page 415.

Bibliog (bibliography)

Purpose

The Bibliog special division contains lists of documents and other materials relevant or related to a document. You can use BibList elements within Bibliog to contain or generate bibliography lists.

Examples

```
<bibliog>
<specdprolog><gendtitle></specdprolog>
<dbody>
<biblist><bibentry><doctitle><titleblk><title>My Nice
Book</title></titleblk></doctitle></bibentry>
<bibentry><doctitle><titleblk><title>Your Nice Book
</title></titleblk></doctitle></bibentry>
</biblist>
</dbody></bibliog>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using bibliography (Bibliog)” on page 89.

Contexts

Children: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “SpecDProlog (special section division prolog)” on page 415.

Parents: “BackM (back matter)” on page 217, “FrontM (front matter)” on page 270.

BibList (bibliography entry list)

Purpose

The BibList element either generates or contains a list of bibliography entries.

You can create an explicit bibliography list by building it out of any combination of bibliography entries, library entries, or Cit elements.

Examples

```
<bibliog>
<specdprolog><gendtitle></specdprolog>
<dbody>
<biblist><bibentry><doctitle><titleblk><title>My Nice
Book</title></titleblk></doctitle></bibentry>
<bibentry><doctitle><titleblk><title>Your Nice Book
</title></titleblk></doctitle></bibentry>
</biblist>
</dbody></bibliog>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

SPEC= AUTO

Specifies that the content of the element is generated from bibliographic references from the body of the document.

ENTRYTYPE=DOC | LIB | DOCORLIB

Indicates whether or not the generated list is to contain BibEntry entries or LibEntry entries. If DOCORLIB is used, a list that includes both can be generated. DOC is the default value.

FORM= NORMAL | FULL | TITLE | DOCNUM

Defines the form of the generated bibliography entries. The specific meaning of **FULL** and **NORMAL** is defined by the active style. **NORMAL** is the default value.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 13. Bibliographies and citations” on page 119.

Contexts

Children: “BibEntry (bibliographic entry)” on page 218, “Cit (document citation)” on page 229, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

Parents: “BackCover (back cover)” on page 217, “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “FrontCover” on page 270, “LEDI (language element description item)” on page 304, “MsgItem (message description item)” on page 348, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391.

Bin (binary data)

Purpose

The Bin element contains text representing binary data.

Examples

```
<BIN>11000001</BIN>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Highlighting” on page 35.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc

(element description)" on page 251, "Entry (table entry)" on page 260, "Fn (footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgText (message text)" on page 354, "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)" on page 382, "Q (quotation phrase)" on page 402, "SynNote (syntax note)" on page 418, "Term" on page 425, "Warning (warning notice)" on page 439.

Body (document body)

Purpose

Use Body to contain the main body of your document.

Examples

```
<body>
<d>
<dprolog><titleblk>
<title>Introduction</title>
</titleblk></dprolog>
<dbody>
<p>Please type your text here. Thank-you.</p>
</dbody></d>
<d>
<dprolog><titleblk>
<title>Caring for your fruit bat</title>
</titleblk></dprolog>
<dbody>
<p>This contains all sorts of information</p>
</dbody></d>
</body>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See "Creating the body of your document" on page 17.

Contexts

Children: "D (hierarchical division)" on page 242, "DBlk (Division block)" on page 245, "LERS (language element reference section)" on page 308, "ModInfo (modular information)" on page 338, "MsgList (list of message or code descriptions)" on page 352, "Part (major document part)" on page 378, "PartAsm (part assembly)" on page 379, "Proc (procedure)" on page 388.

Parents: "IBMIDDoc (IBM-specific product documentation)" on page 281.

BOFNum (bill of forms number)

Purpose

The BOFNum element contains the bill of forms number assigned to the library described by the LibEntry element.

Examples

```
<libentry>
<library><titleblk><title>My Title</title></titleblk>
</library>
<bofnum>SB0F-1234</bofnum>
</libentry>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “LibEntry (document library definition)” on page 312.

Bridge (bridge between concepts)

Purpose

Use the Bridge element to link two locations in a document together and to explain the linkage. The simplest use of Bridge is to create a bridging title between items in a list. A good example of bridges are the titles used throughout magazine articles to bridge a reader from one topic to the next. Such titles do not define hierarchical divisions, but serve merely as a transition from one part to the next.

Examples

This example shows bridging of two sets of list items:

```
<ol>
<li>Saute the shallots and chopped mushrooms until
the shallots are tender and the liquid from the mushrooms
has cooked away.</li>
<li>Brown the sausage and add to the mushroom mixture.
</li>
<bridge><p>The above may be prepared several hours in
advance and refrigerated. Then, 30 minutes before
serving time, finish the dish</p></bridge>
<li>Mix one can of tomato sauce with the mushroom
and sausage mixture and bring to a slow simmer.
</li>
<li>Add the heavy cream and immediately pour into
a casserole.</li>
<li>Pop into 350-degree oven for 15 minutes.</li>
</ol>
```

Attributes

LINKENDS=*element_id1 element_id2*

The *element_ids* are optional identifiers of the element locations that are to be linked or bridged. If the IDs are not specified, the element to be linked is defaulted. For *element_id1*, the default is the element preceding the Bridge element. For *element_id2*, the default is the element following the Bridge element. If you specify only one of the link ends explicitly, you must specify the keyword **#IMPLIED** for the other element.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Separating or bridging list items” on page 33.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Attention (safety notice)” on page 214, “Bin (binary data)” on page 221, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Danger (danger notice)” on page 243, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “Title” on page 431, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DL (definition list)” on page 253, “DLBlk (definition list block)” on page 254, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “GL (glossary list)” on page 272, “GLBlk (glossary list block)” on page 274, “LELI (language element description item)” on page 304, “LI (list item)” on page 311, “LIBlk (list item block)” on page 314, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmBlk (parameter list block)” on page 376, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “StepNotes (step notes)” on page 416, “SynNote (syntax note)” on page 418, “UL (unordered list)” on page 436, “Warning (warning notice)” on page 439.

Cap (caption)

Purpose

The Cap element contains a caption for Figure or Table element. The caption text can appear in figure and table lists. It should be a relatively short description of the figure or table.

Examples

```
<fig id="pubhist">
  <cap>History of Publishing within IBM</cap>
  <mmobj><objref obj="pubhist">
    <textalt>First, there was the pencil, which begat
    the pen and the typewriter. Then came ATMS, Script/DCF,
    ISIL, BookMaster, and IBMIDDoc.</textalt>
  </mmobj></fig>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Figure captions and descriptions” on page 48 and “Table captions and descriptions” on page 59.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “Fig (figure)” on page 262, “Table” on page 423.

Caution (caution notice)

Purpose

Use Caution to create a caution notice, consisting of one or more paragraphs or other paragraph-level elements. Cautions are normally used to warn about actions that may cause damage to equipment.

Examples

```
<CAUTION>Running your engine without oil may cause irreparable damage.
</CAUTION>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “The perils of processing: Attention, caution, and danger” on page 40.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229

page 229, "Date" on page 244, "Dec (decimal number)" on page 247, "DL (definition list)" on page 253, "Fig (figure)" on page 262, "Formula (math formula)" on page 267, "GL (glossary list)" on page 272, "Hex (hexadecimal)" on page 277, "L (explicit link)" on page 300, "Lines (text with line boundaries)" on page 315, "Litdata (literal data)" on page 316, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "MMObj (multi-media object; artwork)" on page 333, "ModInfo (modular information)" on page 338, "MV (message variable)" on page 355, "Num (number)" on page 366, "Oct (octal number)" on page 369, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "Ph (Phrase)" on page 382, "PK (programming keyword)" on page 385, "PV (parameter variable)" on page 400, "Q (quotation phrase)" on page 402, "RefKey (reference key)" on page 405, "Screen (display screen)" on page 411, "SynPh (syntax phrase)" on page 419, "Syntax (syntax diagram)" on page 420, "Table" on page 423, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "AnnotBody (annotation body)" on page 210, "Bridge (bridge between concepts)" on page 223, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "Entry (table entry)" on page 260, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "PBlk (paragraph block)" on page 380, "ProcEntry (procedure entry point)" on page 390, "ProcIntro (procedure introduction)" on page 391, "Safety (safety notices)" on page 411.

CGraphic (character graphic)

Purpose

The CGraphic element contains a graphic created with box and line characters. See "Entities" on page 5 for information about declaring external data entities.

Migration Note

BookMaster CGraphics can contain characters that are not part of the IBMIDDoc document character set and thus **must** be made into external entities.

Examples

```
<!entity mygraphic SYSTEM "mygraph.cgr" ndata linespec>
...
<FIG>
<CAP>Simple box and line graphic</CAP>
<CGRAPHIC OBJ="mygraphic">
</FIG>
```

Attributes

OBJ=*data-entity-name*

Specifies the SGML file that contains the character graphic. When you specify OBJ, do not include the CGraphic end tag.

NData LINESPEC

You must use a notation of LINESPEC for a CGraphic.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Character graphics” on page 48.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Litdata (literal data)” on page 316, “Ph (Phrase)” on page 382, “RefKey (reference key)” on page 405, “Term” on page 425.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BackCover (back cover)” on page 217, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “FrontCover” on page 270, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

Char (character data)

Purpose

Use the Char element to identify literal character data.

Examples

<P>Enter this character string to indicate
cartoon cussing: <CHAR>%#\$@!@#</CHAR>.

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Highlighting” on page 35.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn

(footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgText (message text)" on page 354, "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)" on page 382, "Q (quotation phrase)" on page 402, "SynNote (syntax note)" on page 418, "Term" on page 425, "Warning (warning notice)" on page 439.

CI (component item)

Purpose

The CI element contains a component item in an assembly list.

Examples

```
<comp1>
<ci idxnum="1" partnum="4563423" upa="1">Bike</ci>
<comp1>
<ci idxnum="1" partnum="1230987" upa="1">Frame</ci>
<ci idxnum="2" partnum="1238475" upa="1">Wheel assembly, front</ci>
<ci idxnum="3" partnum="1234939" upa="1">Wheel assembly, rear</ci>
</comp1>
</comp1>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

IDXNUM

Use the IDXNUM attribute to assign to the item a number that matches an artwork index (callout) number. The number you assign with the IDXNUM attribute shows up in the Asm-Index column and is prefixed with a dash character. The number to the left of the dash is the same assembly number used in the heading prefix.

PARTNUM

Use the PARTNUM attribute to assign the item's part number. The number you assign with the PARTNUM attribute shows up in the Part Number column. Part numbers are limited to seven alphanumeric characters (A-Z, a-z, 0-9), with no intervening blanks.

UPA

Use the UPA (units per assembly) attribute to tell how many of this particular item there are in the assembly. The number you assign with the UPA attribute shows up in the Units column.

Usage

See "Chapter 21. Creating parts catalog lists" on page 191.

Contexts

Children: text (#pcdata), "L (explicit link)" on page 300, "Ph (Phrase)" on page 382, "Term" on page 425, "TM (Trademark)" on page 433.

Parents: "CompL (component list)" on page 234.

Cit (document citation)

Purpose

The Cit element contains a citation to another document.

Examples

Simple citation:

```
<cit><bibentry><doctitle><titleblk><title>Huckleberry  
Finn</title></titleblk></doctitle></bibentry></cit>,  
by Mark Twain, is a most excellent book.
```

Complex citation:

```
See this book <cit bibid="fruitybat"> and that book  
<cit bibid="vampbat" form="full"> for serious bedtime reading.
```

Attributes

BibId=*entry_id*

Specifies the ID of a bibliographic or library entry defined elsewhere. When BIBID is specified, it is an error to specify any content or the Cit end tag.

FORM=**NORMAL** | **FULL** | **TITLE** | **DOCNUM**

Specifies the form of the citation, as follows:

NORMAL

Specifies that the record is to be presented as defined by the active presentation style.

FULL

Specifies that the full range of information in the BibEntry is to be presented.

TITLE

Specifies that only the title is to be presented.

DOCNUM

Specifies that the document number is to be presented.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Simple title citations” on page 37 and “Using title citations” on page 120.

Contexts

Children: “BibEntry (bibliographic entry)” on page 218, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BibList (bibliography entry list)” on page 220, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326

page 325, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

ClassDef (element class definition)

Purpose

The ClassDef element defines an element class. ClassNames are defined for the document in which the ClassDef element occurs.

Examples

This example shows te common IBM class definitions

```
<classdef classname="ibmcommand" eletypes="ph" style="bold">
<sem>Command names. For example: COPY command</sem>
</classdef>
<classdef classname="ibmemphasis" eletypes="ph" style="italic">
<sem>Text the writer wants to emphasize.</sem>
</classdef>
<classdef classname="ibmfilepath" eletypes="ph">
<sem>File path names. For example: c:\config.sys</sem>
</classdef>
<classdef classname="ibmguicontrol" eletypes="ph" style="bold">
<sem>GUI control names: menu names, menu choices, entry fields,
icons, folders. list boxes, push buttons,
radio buttons, spin buttons, or check boxes; NOT:
windows or notebooks.</sem>
</classdef>
```

Here’s a sample use of IBMGuiControl:

Press <ph class="IBMGuiControl">OK</ph> to continue."

Attributes

See “Common Element Attributes (large set)” on page 204.

CLASSNAME=*classname*

The name of the class being defined.

ELETYPES=*element names*

Defines those element types (generic identifiers) to which this ClassDef applies. Use ELETYPES when a ClassDef is only meaningful for a specific set of element types.

STYLE=*styles*

Specifies the style of the element to assign to the class name.

SEM

Defines the semantic meaning of a given class. Sem is intended to document what a given element class means to the author that defined it.

Usage

See “Defining Element Classes” on page 179.

Contexts

Children: “IdxTerm (index term)” on page 293, “Sem (semantic meaning)” on page 412, “Title” on page 431.

Parents: "PropDefs (property definitions)" on page 396, "PropGroup (property group)" on page 397.

CLE (content list entry)

Purpose

Use the CLE element to contain an explicit table of contents, figure list, or table list entry to be used in a content list. The CLE element can also refer to an item to be used in a content list. CLE is used to create entries in content lists manually, rather than relying on a program's ability to create such a list.

The CLE may contain the entry information directly, in its content. This text may include the leader dots and page numbers. If the CLE is a table of contents entry, the LVL attribute may be used to set the level in the table of contents that the entry should take. If the page number is not included in the CLE content, it may be specified in the STYLE attribute using the Bookmaster page attribute.

```
<TOC SPEC="man">
  ⋮
  <CLE LVL="2" STYLE="BKM:(page=134)">entry text shown at level 2
  ⋮
</TOC>
```

You can also provide the text for the CLE, but refer to the material that it represents. This could be useful if you wanted to specify a different entry title in the table of contents than the one used on the object being referenced for TOC consistency, for example. In this case, page number is generated from the object reference.

```
<TOC SPEC="man">
  ⋮
  <CLE REFID="div3">Division Three
  ⋮
</TOC>
  ⋮
<D ID="div3">
  <DProlog>
    <TitleBlk>
      <Title>Division Title Number 3</Title>
    </TitleBlk>
  </DProlog>
```

The CLE element can also be used to manually generate a table of contents entry completely by reference. This could be done where all of the material is present, and the titles are correct but some entries generated by automatic processing are not desired. This method also allows specific entries to be created or included that are not created during automatic processing.

```
<TOC SPEC="man">
  ⋮
  <CLE REFID="div3">
  ⋮
</TOC>
  ⋮
<D ID="div3">Division Three Title
  <DProlog>
    <TitleBlk>
      <Title>Division Three Title</Title>
    </TitleBlk>
  </DProlog>
```

Each of the three methods illustrated here may be used in any combination when creating a manual table of contents.

Attributes

See “Common Element Attributes (large set)” on page 204.

REFID=*element_name*

Contains the ID of the element to which the CLE is referring.

%Title

Contains the entry text, or one of the %Title elements.

The LVL attribute is used only when the CLE element is used in a TOC, and when the REFID attribute is not used. The LVL attribute contains a number, and is used to specify the level of the TOC entries presented.

Usage

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “FigList (list of figures)” on page 263, “TList (list of tables)” on page 432, “TOC (table of contents)” on page 435.

Code (message code number)

Purpose

The Code element contains the number of the code being described by the Msg element.

Examples

```
<MSG><CODE>00C4</CODE>  
<MSGITEM CLASS="xp1">  
Occurs when the auto-framatizing circuit blows out.</MSGITEM>  
</MSG>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Message and code lists” on page 29.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “Msg (message or code description)” on page 348.

ColSpec (column specification)

Purpose

The ColSpec element contains the specification for a column.

Examples

```
<TABLE FRAME="ALL">
  <TGROUP COLS="4" COLSEP="1" ROWSEP="1" ORIENT="PORT" PGWIDE="0">
    <COLSPEC COLWIDTH="68*">
    <COLSPEC COLWIDTH="127*">
    <COLSPEC COLWIDTH="195*">
    <COLSPEC COLWIDTH="66*">

    :
```

Attributes

See “Common Element Attributes (large set)” on page 204.

COLNUM=*col_number*

This value indicates the number of the column.

COLNAME=*col_name*

Specifies the column name. This name can be referenced by other table elements.

ALIGN=LEFT | RIGHT | CENTER | JUSTIFY | CHAR

This attribute specifies the horizontal positioning of the text contained in the column:

LEFT

Specifies left alignment (the default).

RIGHT

Specifies right alignment.

CENTER

Specifies center alignment.

JUSTIFY

Specifies that the contained column text is justified,

CHAR

Specifies the character that is used for alignment.

CHAR

specifies the character that is used for alignment.

CHAROFF=*number*

Specifies the character offset for Entry elements in the column.

COLWIDTH=*measure*

Specifies a fixed, proportional, or mixed measure for the column width.

Migration Note

At this time, mixed measures are not supported. You should use proportional measures.

COLSEP=0 | 1

This attribute's value specifies that the internal column rules should be:

- drawn to the right of each cell's content (1)
- not displayed at all (0)

ROWSEP=0 | 1

This attribute's value specifies that the internal row rules should be:

- drawn below each Entry element that ends a row (1)

- not displayed at all (0)

Usage

“Chapter 7. Creating IBMIDDoc Tables” on page 57

Contexts

Children: empty.

Parents: “TGroup (table group)” on page 428.

CompCmt (component comment)

Purpose

The CompCmt element contains a comment about a component item in an assembly.

Examples

```
<ci idxnum="2" partnum="1238475" upa="1">Wheel assembly, front</ci>
<compcmt>For detailed breakdown, see <xref refid="wheel.xml">.</compcmt>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 21. Creating parts catalog lists” on page 191.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “Formula (math formula)” on page 267, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “MD (marked deletion)” on page 326, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term” on page 425, “TM (Trademark)” on page 433, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “CompL (component list)”.

CompL (component list)

Purpose

The CompL element contains a component list for an assembly.

Examples

```
<partasm id="bike" style="bkm:(layout=same)"><title>
Bicycle</title><mmobj><objref obj="bike">
<textalt>Bicycle</textalt>
</mmobj><compl>
```

```

|      <ci idxnum="1" partnum="4563423" upa="1">Bike</ci>
|      <compl>
|      <ci idxnum="1" partnum="1230987" upa="1">Frame</ci>
|      <ci idxnum="2" partnum="1238475" upa="1">Wheel assembly, front</ci>
|      <compemt>For detailed breakdown, see <xref refid="wheelxmp">.</compemt>
|      <ci idxnum="3" partnum="1234939" upa="1">Wheel assembly, rear</ci>
|      </compl>
|      </comp1>
|      </partasm>

```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 21. Creating parts catalog lists” on page 191.

Contexts

Children: “CI (component item)” on page 228, “CompCmt (component comment)” on page 234, “CompL (component list)” on page 234.

Parents: “CompL (component list)” on page 234, “PartAsm (part assembly)” on page 379, “PartAsmSeg (part assembly segment)” on page 380.

Cond (procedure result)

Purpose

The Cond element is half of the Condition/Action pair of elements that is used in the DecisionPnt element. The Cond element contains a description of a condition that requires that some action be taken by the person following the procedure.

Examples

```

<PROC ID="BABYMAP" STYLE="BKM:(STYLE=BASE SEP=INLINE COMPACT)">
<TITLEBLK><TITLE>BABY JOHNNY IS CRYING</TITLE></TITLEBLK>
<PROCENTRY>SIX-MONTH OLD BABY JOHNNY WAS SLEEPING
PEACEFULLY; SUDDENLY HE BEGAN TO CRY.</PROCENTRY>
<PROCSTEP>
<PROCCMND>
<DESC>CHECK JOHNNY'S DIAPER</DESC>
</PROCCMND>
<DECISIONPNT>
<COND>IS THE DIAPER WET?</COND>
<THEN><PROCSTEP><PROCCMND>
<DESC>CHANGE THE DIAPER.</DESC>
</PROCCMND><PROCEXIT>JOHNNY WAS UNCOMFORTABLE.</PROCEXIT>
</PROCSTEP>
</THEN>
<ELSE>
<DESC>CONTINUE AT <XREF REFID="HUNGRY">.</DESC>
</ELSE>
</DECISIONPNT>
</PROCSTEP><PROCSTEP ID="HUNGRY">
<DECISIONPNT>
<COND>IS JOHNNY HUNGRY?</COND>
<THEN><PROCSTEP><DECISIONPNT>
<COND>DOES JOHNNY HAVE TEETH?</COND>
<THEN><PROCSTEP><STEPNOTES><LI>JOHNNY CAN EAT SOLID
FOOD.</LI>
<LI>CONTINUE AT <XREF REFID="FROZSTK">.</LI>
</STEPNOTES></PROCSTEP>

```

```

</THEN>
<ELSE><PROCSTEP ID="BOTTLE"><PROCCMND>
<DESC>WARM A BOTTLE.</DESC>
</PROCCMND><PROCCMND>
<DESC>FEED JOHNNY.</DESC>
</PROCCMND><PROCEXIT>JOHNNY NEEDED A BOTTLE.</PROCEXIT>
</PROCSTEP>
</ELSE>
</DECISIONPNT></PROCSTEP>
</THEN>
<ELSE><PROCSTEP><PROCCMND>
<DESC>ROCK JOHNNY TO SLEEP.</DESC>
</PROCCMND><PROCEXIT>JOHNNY WAS SLEEPY.</PROCEXIT>
</PROCSTEP>
</ELSE>
</DECISIONPNT>
</PROCSTEP></PROC>

```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: text (#pcdata), “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436.

Parents: “DecisionPnt (decision point)” on page 247.

ContainedDocs (documents in IBMLibEntry and LibEntry)

Purpose

The ContainedDocs element is used within IBMLibEntry and LibEntry elements to contain IDs of IBMLibEntry and LibEntry elements.

Examples

```
<CONTAINEDDOCS BIBIDS="bk1 bk2 bk3 bk4 bk5">
```

Attributes

See “Common Element Attributes (large set)” on page 204.

ID=*element_id*

Contains the ID of a BibEntry or LibEntry element.

Usage

For more information about the ContainedDocs element, see “Defining library entries” on page 121.

Contexts

Children: empty.

Parents: “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

CopyR (copyrights)

Purpose

The CopyR element defines the copyright information that must be referenced. Use the COPYR attribute to reference the CopyR element.

Specify one CopyR for every copyright holder for the document or division. You must specify at least the copyright holder and the first date. The presentation of the copyright statement in the final document is a function of the output style.

Examples

```
<IBMIDDOC COPYR="ibmprimary">
  ⋮
<PROLOG>
  <COPYRDEFS>
    <COPYR ID="YR1994">&copyr; COPYRIGHT INTERNATIONAL BUSINESS
    MACHINES CORPORATION 1994. ALL RIGHTS RESERVED.
    <P>This text is added to the end of the generated notice.</P>
    <P>Note to U.S. Government Users -- Documentation related to
      restricted rights -- Use, duplication or disclosure is subject to
      restrictions set forth in GSA ADP Schedule Contract with
      IBM Corp.
    </P>
  </COPYR>
</COPYRDEFS>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

ID=*copyright_id*

The ID of the CopyR element. Contains the ID for CopyR element.

Usage

See “Using CopyRDefs” on page 78.

Contexts

Children: text (#pcdata), “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436.

Parents: “CopyRDefs (copyright definitions)”.

CopyRDefs (copyright definitions)

Purpose

The CopyRDefs element defines copyright attributions for a document or division. Copyright attributions define the intellectual property rights held in the information contained by the document.

Use CopyRDefs to contain the copyright ownership information for the document or division. Put copyrights at the highest level to which they apply. For example, the primary author of the document should always have a copyright attribution at the document level, but a portion of a document may be copyrighted by someone else.

In order for the copyright to apply, it must be referenced within the document.

Examples

```
<PROLOG>
<COPYRDEFS>
  <COPYR ID="ibmprimary">
    © COPYRIGHT INTERNATIONAL BUSINESS
    MACHINES CORPORATION 1994. ALL RIGHTS RESERVED.
    <P>This text is added to the end of the generated notice.</P>
    <P>Note to U.S. Government Users -- Documentation related to
    restricted rights -- Use, duplication or disclosure is subject to
    restrictions set forth in GSA ADP Schedule Contract with
    IBM Corp.</P>
  </COPYR>
</COPYR>
</COPYRDEFS>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

CopyR

Contains copyright attributions for the document or division.

Usage

See “Using CopyRDefs” on page 78.

Contexts

Children: “CopyR (copyrights)” on page 237.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

Corp (enterprise name and address)

Purpose

Corp contains CorpName and address pairs for use in contexts like Author, Approvers, and Owners where either a person or an enterprise could be meaningful.

Examples

```
<authors>
  <author><corp>
    <corpname>International Business Machines</corpname>
  </corp></author>
</authors>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using reader’s comment form (RCF)” on page 90.

Contexts

Children: “Address (address)” on page 208, “CorpName (corporation name)”.

Parents: “Approvers (document approvers)” on page 212, “Author” on page 215, “Maintainer (reader comment)” on page 320, “Owners” on page 373.

CorpName (corporation name)

Purpose

The CorpName simply contains the otherwise unstructured name of an enterprise, such as a company, government agency, or non-profit organization.

Examples

```
<authors>
<author><corp>
<corpname>International Business Machines</corpname>
</corp></author>
</authors>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using reader’s comment form (RCF)” on page 90.

Contexts

Children: text (#pcdata).

Parents: “Corp (enterprise name and address)” on page 238, “Publisher (document publisher)” on page 400.

CoverDef (cover definition)

Purpose

The CoverDef element contains elements that reference to the art used for the document’s covers.

Examples

This example shows how to define the artwork for the front and back covers.

```
<!entity front1 system "front1.eps" ndata graphics>
<!entity back1 system "back1.eps" ndata graphics>
...
<prolog><ibmbibentry><doctitle><titleblk>
<title>Sample Cover</title>
</titleblk></doctitle>
<coverdef>
<frontcover><mmobj><objref obj="front1">
<textalt>System/X cover artwork</textalt>
```

```
</mmobj></frontcover>
<backcover><mmobj><objref obj="back1">
<textalt>System/X back cover artwork</textalt>
</mmobj></backcover>
</coverdef>
</ibmbibentry></prolog>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Adding to the front or back cover (CoverDef)” on page 78.

Contexts

Children: “BackCover (back cover)” on page 217, “FrontCover” on page 270, “MMObj (multi-media object; artwork)” on page 333.

Parents: “IBMBibEntry (IBM bibliographic entry)” on page 279.

CritDate (critical date for a document)

Purpose

The CritDate element contains a critical date in the life of the document.

Examples

```
<CRITDATE>
<DATE>12 June 95</DATE>
<DESC>The date the document was approved for publication.</DESC>
</CRITDATE>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Date” on page 77.

Contexts

Children: “Date” on page 244, “Desc (element description)” on page 251.

Parents: “CritDates (set of critical dates)”.

CritDates (set of critical dates)

Purpose

The CritDates element contains a set of the critical dates in the life of the document.

Examples

```
<CRITDATES>  
<CRITDATE>  
<DATE>12 June 94  
<DESC>The date the document was approved for publication.</DESC>  
</CRITDATE>  
</CRITDATES>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Date” on page 77.

Contexts

Children: “CritDate (critical date for a document)” on page 240.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

D (hierarchical division)

Purpose

The D (division) element defines the hierarchical organization of the information.

Migration Note

D replaces all the Hx elements from BookMaster.

You must explicitly end the D element in order to start another D at the same hierarchical level. This is because the hierarchical level of each division is defined by the containment structure, not by explicit tag names.

Examples

A simple markup example of the D element is:

```
<d>
<dprolog><titleblk>
<title>About Hierarchical Divisions</title>
</titleblk></dprolog>
<dbody>
<p>Hierarchical divisions define the logical organization
of a document.</p>
</dbody></d>
```

Attributes

TOC=*toc* | *notoc*

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older *style=hidden* attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

LANGUAGE=*lang_name*

Specifies the language in which the division is written.

IBMSEC=*sec_level*

SEC=*sec_level*

Specifies the security classification of the D content.

COPYR=*copyrinf*

IBMCOPYR=*ibmcopyrinf*

Specifies the IBM or non-IBM copyright information.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating divisions (D element)” on page 18.

Contexts

Children: “Abstract (abstract)” on page 207, “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DProlog (division prolog)” on page 256, “DSum (division summary)” on page 257.

Parents: “Appendix” on page 212, “BackM (back matter)” on page 217, “Body (document body)” on page 222, “DBlk (Division block)” on page 245, “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “FrontM (front matter)” on page 270, “LEDI (language element description item)” on page 304, “MsgItem (message description item)” on page 348, “ProcIntro (procedure introduction)” on page 391.

Danger (danger notice)

Purpose

Use Danger to create a danger notice, consisting of one or more paragraphs or other paragraph or phrase-level elements. Danger elements are normally used to contain warnings about actions that may cause injury or death to a person.

Examples

```
<danger>
Working under an automobile supported only
by the jack may result in injury or death. Always
use jack stands or ramps in axle pairs to support
your vehicle.
</danger>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “The perils of processing: Attention, caution, and danger” on page 40.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL

(unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "AnnotBody (annotation body)" on page 210, "Bridge (bridge between concepts)" on page 223, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "Entry (table entry)" on page 260, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "PBlk (paragraph block)" on page 380, "ProcEntry (procedure entry point)" on page 390, "ProcIntro (procedure introduction)" on page 391, "Safety (safety notices)" on page 411.

Date

Purpose

Use the Date element to contain a date. IBMIDDoc does not define the format of the date element, but processing systems can define format constraints for Date element content.

Examples

```
<ANNOT>
  <P>This change was made on <DATE>August 14, 1994</DATE>.
</ANNOT>

<ANNOT>
  <P>This document was formatted on <DATE SPEC="AUTO">.
</ANNOT>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

SPEC=AUTO

Indicates that the presented date is to be defined by the presentation system and the defined document style. By default, this is the system date at the time the document is processed. When AUTO is specified, the element must be empty.

Usage

See "Date" on page 77.

Contexts

Children: text (#pcdata).

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "CompCmt (component comment)" on page 234, "CritDate (critical date for a document)" on page 240, "Danger (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "Entry (table entry)" on page 260, "Fn (footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344

page 342, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “Rev (revision)” on page 408, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

DBlk (Division block)

Purpose

The DBlk element is used to organize divisions. A common use is to include two or more divisions from an object library.

Examples

```
<objlib>
<objlibbody><dblk id="somechapters">
<d>
<dprolog><titleblk>
<title>A heading</title>
</titleblk></dprolog>
<dbody>
<p>some interesting information</p>
</dbody></d>
<d>
<dprolog><titleblk>
<title>Another heading</title>
</titleblk></dprolog>
<dbody>
<p>more interesting information.</p>
</dbody></d>
</dblk></objlibbody>
</objlib>
...
<dblk conloc="somechapters" props="novice">
```

Attributes

Conloc

The CONLOC attribute specifies that the content of another element of the same type is to be used as the content of the referencing element. This enables reuse of information. When the CONLOC attribute is specified, you cannot specify the element’s end tag. The result of using CONLOC is exactly the same as if the element being referred to had occurred at that point in the document. See “Reusing elements from an object library” on page 166.

Attributes on the element are now passed to the element with the CONLOC; this is a feature that began with IDWB release 3.4, patch IDWXF036.

ID The ID attribute identifies an element within an SGML document. IDs must be unique within a single document. Any element that has an ID can be cross=referenced or linked to. IDs can be up to 64 characters long. IDs must start with an alphabetic character and can contain letters, numbers, dashes (-), or periods (.).

Props

The Props attribute is used to specify the condition under which the information contained within the element appears. See “Property-Based Retrieval” on page 171.

PropSrc

Points to an element whose properties are to be used as the properties of the referencing element. See “Chapter 19. Property and Class Definition” on page 177.

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See “Using Revisions” on page 91.

Status

ignored by processes

Contexts

Children: “D (hierarchical division)” on page 242.

Parents: “Appendix” on page 212, “BackM (back matter)” on page 217, “Body (document body)” on page 222, “DBody (division body)”, “FrontM (front matter)” on page 270, “LEDI (language element description item)” on page 304, “MsgItem (message description item)” on page 348.

DBody (division body)

Purpose

DBody contains the content of a hierarchical division; that is, the main content of a chapter or topic.

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating divisions (D element)” on page 18.

Examples

```
<d>
<dprolog><titleblk>
<title>About Hierarchical Divisions</title>
</titleblk></dprolog>
<dbody>
<p>Hierarchical divisions define the logical organization
of a document.</p>
</dbody></d>
```

Contexts

Children: “Annot (annotation)” on page 209, “AsmList (list of parts assemblies)” on page 214, “Attention (safety notice)” on page 214, “BibList (bibliography entry list)” on page 220, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “D (hierarchical division)” on page 242, “Danger (danger notice)” on page 243, “DBlk (Division block)” on page 245, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “FNList (footnote list)” on page 266, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “LERS (language element reference section)” on page 308, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MarkList (marked note list)” on page 321, “MkNote (marked note)” on page 331, “MMObj (multi-media object);

artwork)" on page 333, "ModInfo (modular information)" on page 338, "MsgList (list of message or code descriptions)" on page 352, "Note" on page 362, "NoteList (ordered note list)" on page 363, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PartAsm (part assembly)" on page 379, "PBlk (paragraph block)" on page 380, "Proc (procedure)" on page 388, "Screen (display screen)" on page 411, "Syntax (syntax diagram)" on page 420, "Table" on page 423, "UL (unordered list)" on page 436, "Xmp (example)" on page 440.

Parents: "Abbrev (abbreviations)" on page 207, "Abstract (abstract)" on page 207, "Bibliog (bibliography)" on page 219, "D (hierarchical division)" on page 242, "Glossary" on page 276, "Legend" on page 306, "Part (major document part)" on page 378, "Preface" on page 387, "SOA (summary of amendments)" on page 413.

Dec (decimal number)

Purpose

Use the Dec element to identify decimal data, which is data that is encoded in a base-10 numbering system.

Examples

```
<BIN>11000001</BIN> = <DEC>193<DEC>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See Table 1 on page 36.

Contexts

Children: text (#pcdata).

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "CompCmt (component comment)" on page 234, "Danger (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "Entry (table entry)" on page 260, "Fn (footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgText (message text)" on page 354, "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)" on page 382, "Q (quotation phrase)" on page 402, "SynNote (syntax note)" on page 418, "Term" on page 425, "Warning (warning notice)" on page 439.

DecisionPnt (decision point)

Purpose

The DecisionPnt element defines one or more condition-action pairs that define the next step in a procedure.

Attributes

See “Common Element Attributes (large set)” on page 204.

Cond

Defines the condition which, if satisfied, indicates the action that should be taken.

Then

What action to take if the condition is satisfied.

Else

What action to take if the condition is not satisfied.

Usage

See .

Examples

```
<PROC ID="BABYMAP" STYLE="BKM:(STYLE=BASE SEP=INLINE COMPACT)">
<TITLEBLK><TITLE>BABY JOHNNY IS CRYING</TITLE></TITLEBLK>
<PROCENTRY>SIX-MONTH OLD BABY JOHNNY WAS SLEEPING
PEACEFULLY; SUDDENLY HE BEGAN TO CRY.</PROCENTRY>
<PROCSTEP>
<PROCCMND>
<DESC>CHECK JOHNNY'S DIAPER</DESC>
</PROCCMND>
<DECISIONPNT>
<COND>IS THE DIAPER WET?</COND>
<THEN><PROCSTEP><PROCCMND>
<DESC>CHANGE THE DIAPER.</DESC>
</PROCCMND><PROCEXIT>JOHNNY WAS UNCOMFORTABLE.</PROCEXIT>
</PROCSTEP>
</THEN>
<ELSE>
<DESC>CONTINUE AT <XREF REFID="HUNGRY">.</DESC>
</ELSE>
</DECISIONPNT>
</PROCSTEP><PROCSTEP ID="HUNGRY">
<DECISIONPNT>
<COND>IS JOHNNY HUNGRY?</COND>
<THEN><PROCSTEP><DECISIONPNT>
<COND>DOES JOHNNY HAVE TEETH?</COND>
<THEN><PROCSTEP><STEPNOTES><LI>JOHNNY CAN EAT SOLID
FOOD.</LI>
<LI>CONTINUE AT <XREF REFID="FROZSTK">.</LI>
</STEPNOTES></PROCSTEP>
</THEN>
<ELSE><PROCSTEP ID="BOTTLE"><PROCCMND>
<DESC>WARM A BOTTLE.</DESC>
</PROCCMND><PROCCMND>
<DESC>FEED JOHNNY.</DESC>
</PROCCMND><PROCEXIT>JOHNNY NEEDED A BOTTLE.</PROCEXIT>
</PROCSTEP>
</ELSE>
</DECISIONPNT></PROCSTEP>
</THEN>
<ELSE><PROCSTEP><PROCCMND>
<DESC>ROCK JOHNNY TO SLEEP.</DESC>
</PROCCMND><PROCEXIT>JOHNNY WAS SLEEPY.</PROCEXIT>
</PROCSTEP>
</ELSE>
</DECISIONPNT>
</PROCSTEP></PROC>
```

Contexts

Children: “Cond (procedure result)” on page 235, “Else (other procedure path to follow)” on page 259, “Then (procedure action to take)” on page 430.

Parents: “ProcStep (procedure step)” on page 392.

Defn (definition of a term)

Purpose

The Defn element contains the definition of a term.

Examples

```
<dl>
<dlentry><term>gopher</term>
<defn>A burrowing rodent that feeds on roots of plants.
</defn>
</dlentry>
<dlentry><term>lawn</term>
<defn>Gopher highway. <p>Can be identified by dinner-plate-sized
mounds of dirt where grass used to be.</p></defn>
</dlentry>
<dlentry><term>agapanthus</term>
<defn>Lovely flowering plant, the roots of which are
the preferred food of gophers. <p>If your flourishing
agapanthus suddenly keels over, it means a gopher
has had a feast.</p></defn>
</dlentry>
</dl>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Definition lists” on page 26.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Attention (safety notice)” on page 214, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Danger (danger notice)” on page 243, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display

screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “DLEntry (definition list entry)” on page 255, “GLEntry (glossary list entry)” on page 275, “Parm (parameter list entry)” on page 375.

DefnHd (definition description heading)

Purpose

The DefnHd element contains the heading for the description portion of a definition or parameter list.

Examples

```
<dl><termhd>Setting</termhd>
<defnhd>Description</defnhd>
<dentry><term>Low</term>
<defn>A good setting for simmering soups.</defn>
</dentry>
<dentry><term>Medium</term>
<defn>After the water has boiled, use this setting
for cooking the spaghetti.</defn>
</dentry>
<dentry><term>High</term>
<defn>Use this setting to get water boiling fast.
</defn>
</dentry>
</dl>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Definition lists” on page 26.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “DL (definition list)” on page 253, “ParmL (parameter list)” on page 376.

Delim (syntax delimiter)

Purpose

The Delim tag specifies a delimiter that is to indicate the start or end of keywords, variables, operators, or groups. The delimiter can consist of one or more characters.

Examples

```
<syntax>
<group>
<kwd>FRED</kwd>
```

```
<delim>+</delim>
<kwd>WILMA</kwd>
</group>
</syntax>
```

Attributes

OPTREQ=REQ | OPT

Indicates whether or not the delimiter is optional or required. REQ (required) is the default.

STARTEND=START | END

The STARTEND attribute has a value of START or END, depending upon whether the Delim element is the starting or ending delimiter in the syntax.

CONVAR=CONSTANT | VAR

Indicates whether the content of the element is a constant or a variable in the context where it is used.

LINKEND=reference-id

Contains an ID reference that enables a link to an arbitrary location in the document.

See “Common Element Attributes (large set)” on page 204.

Usage

See “The Delim (delimiter) element” on page 131.

Contexts

Children: text (#pcdata).

Parents: “Group” on page 277, “SynPh (syntax phrase)” on page 419.

Desc (element description)

Purpose

The Desc element contains a description of an element. Many elements allow Desc in their content, usually as the first element, or following the title. Desc is intended to contain a description of the content of the element within which Desc appears. The presentation effect of a given Desc element is determined by the style of the element that contains the Desc. For example, within a figure or table, the Desc may be presented as part of the caption; in another element the Desc may not be presented at all.

Examples

```
<fig ID="figa">
  <cap>Your CPU with the Whantoozler 3.0 Installed</cap>
  <desc>This figure shows the elegance of the Whantoozler when properly
  installed.</desc>
</FIG>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), "Address (address)" on page 208, "APL (APL data)" on page 211, "Bin (binary data)" on page 221, "Char (character data)" on page 227, "Cit (document citation)" on page 229, "Date" on page 244, "Dec (decimal number)" on page 247, "DL (definition list)" on page 253, "Fig (figure)" on page 262, "Hex (hexadecimal)" on page 277, "L (explicit link)" on page 300, "MD (marked deletion)" on page 326, "MMObj (multi-media object; artwork)" on page 333, "MV (message variable)" on page 355, "Note" on page 362, "NoteList (ordered note list)" on page 363, "Num (number)" on page 366, "Oct (octal number)" on page 369, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "PBlk (paragraph block)" on page 380, "Ph (Phrase)" on page 382, "PK (programming keyword)" on page 385, "PV (parameter variable)" on page 400, "Q (quotation phrase)" on page 402, "RefKey (reference key)" on page 405, "StepRef (procedure step reference)" on page 416, "SynPh (syntax phrase)" on page 419, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "Authors" on page 216, "BibEntry (bibliographic entry)" on page 218, "CritDate (critical date for a document)" on page 240, "Else (other procedure path to follow)" on page 259, "Fig (figure)" on page 262, "IBMBibEntry (IBM bibliographic entry)" on page 279, "IBMLibEntry (IBM document library definition)" on page 287, "LERSDef (LERS property definition)" on page 310, "LibEntry (document library definition)" on page 312, "Mark (marked note definition)" on page 320, "ModInfo (modular information)" on page 338, "ModInfoDef (modular information property definition)" on page 340, "ModItemDef (item class definitions)" on page 341, "MsgItemDef (definition of message description items)" on page 350, "ObjLib (object library)" on page 367, "Proc (procedure)" on page 388, "ProcCmdn (procedure command)" on page 389, "PropDef (property set definition)" on page 395, "PropDesc (property description)" on page 397, "PropGroup (property group)" on page 397, "Qualif (qualification)" on page 403, "Rev (revision)" on page 408, "Table" on page 423, "Then (procedure action to take)" on page 430.

DIntro (division introduction)

Purpose

The DIntro element contains the introduction to a hierarchical division D.

Note that regular D elements are used to subdivide the DIntro section, but that they are not numbered like the divisions in the division body. This is possible because the divisions within the division introduction are structurally distinguished from the divisions in the division body.

Examples

```
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dintro>
<p>My little division introductory sentence.</p>
</dintro>
<dbody>
<p>Here's the beginning of my chapter.</p>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Division introductions” on page 21.

Contexts

Children: “Annot (annotation)” on page 209, “AsmList (list of parts assemblies)” on page 214, “Attention (safety notice)” on page 214, “BibList (bibliography entry list)” on page 220, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “D (hierarchical division)” on page 242, “Danger (danger notice)” on page 243, “DL (definition list)” on page 262, “FNList (footnote list)” on page 266, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MarkList (marked note list)” on page 321, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Screen (display screen)” on page 411, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “TitleBlk (title information)” on page 432, “TOC (table of contents)” on page 435, “UL (unordered list)” on page 436, “Xmp (example)” on page 440.

Parents: “Abbrev (abbreviations)” on page 207, “Abstract (abstract)” on page 207, “Bibliog (bibliography)” on page 219, “D (hierarchical division)” on page 242, “Glossary” on page 276, “Legend” on page 306, “MasterIndex (master index)” on page 323, “Part (major document part)” on page 378, “Preface” on page 387, “SOA (summary of amendments)” on page 413.

DL (definition list)

Purpose

The DL element contains a list of pairs of terms and definitions. Use definition lists as a generic definition structure for defining things other than glossary terms. Entries can be organized within a definition list using DLBlk elements. Bridge elements can also be used to create transitions or connections between blocks of entries.

Examples

```
<dl>
<dlentry><term>gopher</term>
<defn>A burrowing rodent that feeds on roots of plants.
</defn>
</dlentry>
<dlentry><term>lawn</term>
<defn>Gopher highway. <p>Can be identified by dinner-plate-sized
mounds of dirt where grass used to be.</p></defn>
</dlentry>
<dlentry><term>agapanthus</term>
<defn>Lovely flowering plant, the roots of which are
the preferred food of gophers. <p>If your flourishing
```

```

agapanthus suddenly keels over, it means a gopher
has had a feast.</p></defn>
</dlenry>
</dl>

```

Attributes

TERMWIDTH= SMALL | MEDIUM | LARGE | 1 | 2

You can use the TERMWIDTH attribute to determine the indentation size of the definition list. The valid choices are: small (.5 inch, the default), medium (1 inch), and large (2 inches). The value "1" is for 1-character width; "2" is for a 2-character width.

LINESPACE=SPACE | COMPACT

Specifies whether the items in the list should be compacted or have space between the items. Nested lists automatically inherit the setting of the outer list, but can override that default with their own setting.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Definition lists" on page 26.

Contexts

Children: "Bridge (bridge between concepts)" on page 223, "DefnHd (definition description heading)" on page 250, "DLBlk (definition list block)", "DLEntry (definition list entry)" on page 255, "TermHd (term heading)" on page 426.

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "BackCover (back cover)" on page 217, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "Cond (procedure result)" on page 235, "CopyR (copyrights)" on page 237, "Danger (danger notice)" on page 243, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "EdNotices (edition notices)" on page 259, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "Fn (footnote)" on page 265, "FrontCover" on page 270, "LeDesc (language element description)" on page 304, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "NItem (notice item)" on page 359, "NoteBody (note body)" on page 362, "Notices (contains notices)" on page 364, "P (paragraph)" on page 374, "PBlk (paragraph block)" on page 380, "ProcEntry (procedure entry point)" on page 390, "ProcExit (procedure exit point)" on page 391, "ProcIntro (procedure introduction)" on page 391, "Safety (safety notices)" on page 411, "Sem (semantic meaning)" on page 412, "SynNote (syntax note)" on page 418, "TextAlt (text alternative)" on page 427, "Warning (warning notice)" on page 439.

DLBlk (definition list block)

Purpose

The DLBlk element is used to organize definition list entries within a definition list.

Examples

```
<d1>
<d1blk>
<d1entry><term>Cat</term>
<defn>A house pet
that purrs when happy.</defn></d1entry>
<d1entry><term>Dog</term>
<defn>A house pet that wags
its tail when happy.</defn></d1entry>
</d1blk>
<d1blk>
<d1entry><term>Fish</term>
<defn>A house pet
with scales that swims.</defn></d1entry>
<d1entry><term>Turtle</term>
<defn>A house pet with
scales that swims and walks slowly.</defn></d1entry>
</d1blk>
</d1>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Definition lists” on page 26.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “DLEntry (definition list entry)”, “Title” on page 431.

Parents: “DL (definition list)” on page 253.

DLEntry (definition list entry)

Purpose

The DLEntry element contains a single term and its definition. Use the DLEntry element within a definition list (DL) to define information that is not glossary information. Glossary entries (GLEntry) should be used for formal, dictionary-style definitions of words.

Examples

```
<d1>
<d1entry><term>gopher</term>
<defn>A burrowing rodent that feeds on roots of plants.
</defn>
</d1entry>
<d1entry><term>lawn</term>
<defn>Gopher highway. <p>Can be identified by dinner-plate-sized
mounds of dirt where grass used to be.</p></defn>
</d1entry>
<d1entry><term>agapanthus</term>
<defn>Lovely flowering plant, the roots of which are
the preferred food of gophers. <p>If your flourishing
agapanthus suddenly keels over, it means a gopher
has had a feast.</p></defn>
</d1entry>
</d1>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Definition lists” on page 26.

Contexts

Children: “Defn (definition of a term)” on page 249, “Term” on page 425.

Parents: “DL (definition list)” on page 253, “DLBlk (definition list block)” on page 254.

DocTitle (document title)

Purpose

The DocTitle element contains document title information. You should always provide a short title for IBM documentation. The short title will be used in places like bookshelf lists, citations, and the book’s spine. The complete title that appears on the document title page or cover page is defined by the document style and may include data from other prolog elements.

Examples

```
<prolog>
<ibmbibentry><doctitle><titleblk>
<title>My Cute, Little Document</title>
</titleblk></doctitle>
<ibmdocnum>SC99-1234-01</ibmdocnum>
<authors>
<author><person>
<name>Fred Mertz</name>
<address>East Overshoe, SD</address>
</person></author>
</authors>
</ibmbibentry>
</prolog>
```

Usage

See “Document title” on page 76.

Contexts

Children: “Library” on page 314, “TitleBlk (title information)” on page 432.

Parents: “BibEntry (bibliographic entry)” on page 218, “IBMBibEntry (IBM bibliographic entry)” on page 279.

DProlog (division prolog)

Purpose

The DProlog element contains metainformation about a division, which is information that describes the division, such as the division title, the author, and so on. It also contains many different types of markup definitions used to define classes and properties for the division.

Examples

```
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk>
<revdefs>
<rev id="v3r4" ident="use">
<date>June 5th</date>
<desc>Something happened...</desc>
</rev>
</revdefs></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
<p rev="v3r4">Something that changed on June 5th.
</p>
</dbody></d>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Division prologs” on page 21.

Contexts

Children: “Approvers (document approvers)” on page 212, “Authors” on page 216, “BibEntryDefs (contains bibliographic entries)” on page 219, “CopyRDefs (copyright definitions)” on page 237, “CritDates (set of critical dates)” on page 240, “GIDefs (glossary definitions)” on page 275, “IBMProdInfo (IBM product information)” on page 291, “IdxDefs (central index entries)” on page 292, “LDescs (link descriptions)” on page 302, “Maintainer (reader comment)” on page 320, “MasterIndexInfo (master index information)” on page 324, “ObjLib (object library)” on page 367, “Owners” on page 373, “ProdInfo (product information)” on page 393, “PropDefs (property definitions)” on page 396, “QualifDefs (qualification definitions)” on page 404, “RetKey (retrieval key)” on page 407, “RevDefs (revision tracking information)” on page 409, “TitleBlk (title information)” on page 432.

Parents: “D (hierarchical division)” on page 242, “Part (major document part)” on page 378.

DSum (division summary)

Purpose

DSum contains a summary of the informational content of the division.

Examples

```
<d>
<dprolog>
<titleblk><title>Wantoozler features</title>
</titleblk>
</dprolog>
<dbody>
...
</dbody>
<dsum>
```

<p>In summary, the Whantoozler can triple the normal operating speed of your system.</p>
</dsum>
</d>

Usage

See “Creating divisions (D element)” on page 18.

Contexts

Children: “Annot (annotation)” on page 209, “AsmList (list of parts assemblies)” on page 214, “Attention (safety notice)” on page 214, “BibList (bibliography entry list)” on page 220, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “D (hierarchical division)” on page 242, “Danger (danger notice)” on page 243, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “FNList (footnote list)” on page 266, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MarkList (marked note list)” on page 321, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Screen (display screen)” on page 411, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “TitleBlk (title information)” on page 432, “UL (unordered list)” on page 436, “Xmp (example)” on page 440.

Parents: “Abbrev (abbreviations)” on page 207, “Abstract (abstract)” on page 207, “Bibliog (bibliography)” on page 219, “D (hierarchical division)” on page 242, “Glossary” on page 276, “Legend” on page 306, “MasterIndex (master index)” on page 323, “Part (major document part)” on page 378, “Preface” on page 387, “SOA (summary of amendments)” on page 413.

DVCFObj (DVCF Migration Element)

Purpose

This element should only be temporarily used when migrating BookMaster DVCF coding to IBMIDDoc. Do not continue to use this; you will be warned with messages when your document is processed.

Contexts

Children: any element.

Parents:.

EdNotices (edition notices)

Purpose

The EdNotices element contains the edition notices for the document, including any legally required statements about intended use, updates, and the like.

Examples

```
<ibmddoc ibmcopyr="1996, 1999">
...
<ednotices><title>First Edition (June 1997)</title>
<p>This edition applies to the IBMDDoc language,
Version 4.2, and to all subsequent releases
and modifications until otherwise indicated in new
editions.</p>
</ednotices>
```

Attributes

SPEC=MAN

Specifies whether the edition notice is generated automatically or manually. At this time, MAN is the only supported value.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Notices and Edition notices” on page 86.

Contexts

Children: “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Table” on page 423, “Title” on page 431, “UL (unordered list)” on page 436.

Parents: “FrontM (front matter)” on page 270.

Else (other procedure path to follow)

Purpose

The Else element contains the step or steps to follow if the Then condition is not met.

Examples

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “Desc (element description)” on page 251, “Proc (procedure)” on page 388, “ProcStep (procedure step)” on page 392.

Parents: “DecisionPnt (decision point)” on page 247.

Entry (table entry)

Purpose

The Entry element contains an entry within a row.

Examples

```
<table pgwide="0" id="tablesample">
<cap>Sample table caption</cap>
<tgroup cols="1">
<colspec colname="col1">
<tbody>
<row>
<entry colname="col1">my little</entry>
</row>
<row>
<entry colname="col1">sample table</entry>
</row>
</tbody>
</tgroup>
</table>
```

Attributes

COLNAME=*column_name*

Specifies the column name to which the Entry belongs.

NAMEST=*start_name*

Specifies the name of the leftmost column of a horizontal span.

NAMEEND=*end_name*

Specifies the name of the rightmost column of a horizontal span.

SPANNAME=*span_name*

Specifies the name of a horizontal span in a TGroup.

MOREROWS=*number*

Specifies the number of additional rows to add in a vertical span.

VALIGN=**TOP** | **MIDDLE** | **BOTTOM**

This attribute specifies the vertical alignment of the text contained in the Entry elements.

TOP

specifies alignment of the text at the top of the Entry elements.

MIDDLE

specifies alignment of the text at the middle of the Entry elements.

BOTTOM

specifies alignment of the text at the bottom of the Entry elements (the default).

ALIGN=**LEFT** | **RIGHT** | **CENTER** | **JUSTIFY** | **CHAR**

This attribute specifies the horizontal positioning of the text contained in the column:

LEFT

specifies left alignment (the default).

RIGHT

specifies right alignment.

CENTER

Specifies center alignment.

CHAR

Specifies the character that is used for alignment.

CHAROFF=number

Specifies the character offset for Entry elements in this column.

COLSEP 0 (NO) | 1 (YES)

This attribute's value specifies that the internal column rules should be:

- drawn to the right of each Entry element that ends a column (1)
- not displayed at all (0)

ROWSEP 0 (NO) | 1 (YES)

This attribute's value specifies that the internal row rules should be:

- drawn below each Entry element that ends a row (1)
- not displayed at all (0)

ROTATE 0 (NO) | 1 (YES)

Specifies whether the entry should be rotated.

Usage

See "Chapter 7. Creating IBMIDDoc Tables" on page 57.

Contexts

Children: text (#pcdata), "Address (address)" on page 208, "Annot (annotation)" on page 209, "APL (APL data)" on page 211, "Attention (safety notice)" on page 214, "Bin (binary data)" on page 221, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "CGraphic (character graphic)" on page 226, "Char (character data)" on page 227, "Cit (document citation)" on page 229, "Danger (danger notice)" on page 243, "Date" on page 244, "Dec (decimal number)" on page 247, "DL (definition list)" on page 253, "Fig (figure)" on page 262, "Formula (math formula)" on page 267, "GL (glossary list)" on page 272, "Hex (hexadecimal)" on page 277, "L (explicit link)" on page 300, "Lines (text with line boundaries)" on page 315, "Litdata (literal data)" on page 316, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "MMObj (multi-media object; artwork)" on page 333, "MV (message variable)" on page 355, "Note" on page 362, "NoteList (ordered note list)" on page 363, "Num (number)" on page 366, "Oct (octal number)" on page 369, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "Ph (Phrase)" on page 382, "PK (programming keyword)" on page 385, "PV (parameter variable)" on page 400, "Q (quotation phrase)" on page 402, "RefKey (reference key)" on page 405, "Screen (display screen)" on page 411, "SynPh (syntax phrase)" on page 419, "Syntax (syntax diagram)" on page 420, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "Row (table row)" on page 409.

ExternalFileName

Purpose

This specifies the file name of this document. This is used for PDF cross-referencing.

Examples

```
<externalfilename>iddugref</externalfilename>
```

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “BibEntry (bibliographic entry)” on page 218, “IBMBibEntry (IBM bibliographic entry)” on page 279.

Fig (figure)

Purpose

The Fig element contains and identifies figures, such as images, examples, and formulas. The figure serves to contain the exhibits and associate a caption and a description with them. It also allows those exhibits to be referenced.

Use FigSeg elements to break long figures into smaller chunks to enable breaking of figures at logical points. In a code sample, for example, you can use one FigSeg for each subroutine to ensure that no subroutines are broken in the middle.

Examples

```
<fig style="bkm:(place=inline width=column)">
<lines>Here are some lines
in the sample, simple figure.</lines>
</fig>
```

Attributes

FRAME=NONE | BOX | RULES

Causes the figure to have a frame. The default is none — no frame.

Box Causes a box to be placed around the figure.

Rules Causes a line to be placed above and below the figure; to visually separate it from the surrounding text.

STYLE="bkm:(place=inline width=page)"

This style override ensures the following:

place=inline

This causes the figure to be placed inline. Normally, BookMaster formats figures by floating them to the top of the next page.

width=page

This causes the figure to be formatted page-wide. You can also specify **width=column** to have the figure formatted column-wide.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Figures” on page 47.

Contexts

Children: “Annot (annotation)” on page 209, “Bridge (bridge between concepts)” on page 223, “Cap (caption)” on page 225, “CGraphic (character graphic)” on page 226, “Desc (element description)” on page 251, “DL (definition list)” on page 253, “FigSeg (figure segment)” on page 264, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “Syntax (syntax diagram)” on page 420, “UL (unordered list)” on page 436, “Xmp (example)” on page 440.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “EdNotices (edition notices)” on page 259, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “LeDesc (language element description)” on page 304, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NItem (notice item)” on page 359, “NoteBody (note body)” on page 362, “Notices (contains notices)” on page 364, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “Safety (safety notices)” on page 411, “Sem (semantic meaning)” on page 412, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

FigList (list of figures)

Purpose

The FigList element causes a figure list to be generated.

Examples

```
<figlist><gendtitle></figlist>
```

Attributes

SPEC= AUTO | MAN

Specifies that the content of the element is generated. If SPEC=AUTO is specified, a list of all figures in the document is generated.

LAYOUT=Default-Layout | OneCol | TwoCol | ThreeCol

Specifies the column-style for this portion of the book.

OneCol

The entries format across the entire page.

TwoCol

The entries format in two columns.

ThreeCol

The entries format in three columns.

Usage

See “List of figures” on page 87.

Contexts

Children: “CLE (content list entry)” on page 231, “GendTitle (default title specification)” on page 272, “RetKey (retrieval key)” on page 407, “TitleBlk (title information)” on page 432.

Parents: “FrontM (front matter)” on page 270.

FigSeg (figure segment)

Purpose

The FigSeg element organizes the content of a figure into logical segments. The primary intent of FigSeg is to contain parts of a figure that must be kept together when the figure is presented.

Use multiple figure segments to break long figures into smaller chunks to enable breaking of figures at logical points. In a code sample, for example, you might use one figure segment for each subroutine, ensuring that no subroutines will be broken in the middle.

Examples

```
<fig id="samplefigdesc" style="bkm:(place=inline width=column)">
<cap>Here's a sample figure with a caption and description
</cap>
<desc>This figure has a description. Note that descriptions
have punctuations like sentences.</desc>
<figseg>
<xmp>Here is the first part
of a coding example</xmp>
</figseg>
<figseg>
<xmp>Here is the second part
of a coding example</xmp>
</figseg>
</fig>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “Annot (annotation)” on page 209, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “DL (definition list)” on page 253, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315,

"Litdata (literal data)" on page 316, "LQ (stand-alone quotation)" on page 318, "MkNote (marked note)" on page 331, "MMObj (multi-media object; artwork)" on page 333, "Note" on page 362, "NoteList (ordered note list)" on page 363, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "RefKey (reference key)" on page 405, "Screen (display screen)" on page 411, "Syntax (syntax diagram)" on page 420, "UL (unordered list)" on page 436, "Xmp (example)" on page 440.

Parents: "Fig (figure)" on page 262.

FileNum (file number)

Purpose

The FileNum element contains the file number of the document. File numbers are unique to the IBMBibEntry element, and are only used for IBM products.

Examples

```
<FileNum>
444-4444-44
</FileNum>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See "Document title" on page 76.

Contexts

Children: text (#pcdata), "Ph (Phrase)" on page 382.

Parents: "IBMBibEntry (IBM bibliographic entry)" on page 279.

Fn (footnote)

Purpose

Use Fn to annotate text with notes that are not appropriate for inclusion in-line or to indicate the source for facts or other material used in the text. Footnotes are associated with the content of the element containing the footnote.

Examples

```
<p>There's a footnote<fn>While some folks do not like
footnotes; they sometimes contain a nugget of priceless
lore. Did you know IBMIDDoc's grandmother was named
ISIL?</fn> around here somewhere.</p>
```

Attributes

refid=id

Refers to another footnote identifier. If this attribute is specified, this footnote must be empty.

See "Common Element Attributes (large set)" on page 204.

Usage

See “Footnotes” on page 38.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “FNList (footnote list)”.

FNList (footnote list)

Purpose

The FnList element contains a list of footnotes.

Examples

```
<FNLIST spec="auto">
```

Attributes

SPEC=AUTO

Specifies that the content of the element is generated. If SPEC=AUTO is specified, all Fn elements in the document are presented.

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “Fn (footnote)” on page 265.

Parents: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “LEDI (language element description item)” on page 304, “MsgItem (message description item)” on page 348, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391.

Formula (math formula)

Purpose

The Formula element contains or references a mathematical formula.

The NOTATION attribute must not be used when the OBJ attribute is used. If neither the OBJ or NOTATION attribute is specified, the SGML processor assumes that the inline formula data is encoded using the Script Mathematical Formula Formatter (SMFF) mathematics formula language. Although SMFF is implied if the notation type is not specified, the preferred IBMIDDoc form is to explicitly specify the NOTATION=SMFF, when applicable.

The content of Formula must be inspected and any occurrences of `</` must be modified to use a symbol name `<sl` to insure the formula element is not prematurely ended by an unintended end tag delimiter in the data.

Examples

The first example illustrates how the entity is encoded. The second example shows how to use a formula element that contains the `wave1` entity reference.

```
<!ENTITY wave1 PUBLIC '+//ISBN 0-933186::IBM//ENTITY formula//EN' SDATA SMFF >
```

...

```
<FORMULA ID="pwave1" OBJ="wave1">
```

The next example shows a formula element that contains a formula using SMFF notation.

```
<FORMULA NOTATION="smff">
integral from 0 to infinity of d x
</FORMULA>
```

Attributes

OBJ=*file-entity-name*

Refers to an external file that contains mathematical formula specifications. The attribute value is the name of a declared entity. This attribute is only used when the formula data is in an external file, and the Formula element must be empty when it is used. Note that the entity declaration must include the notation of the mathematical formula information.

When using the OBJ attribute, the formula element must be empty. The entity declaration that defines the entity referred to by this attribute must include the notation used to encode the mathematical formula information.

Notation=SMFF

Refers to the notation that is used to encode the mathematical formula data that is included inline in the document. It should be specified if the formula data is inline in the document. It should not be specified if the Object attribute is used to refer to an external entity containing the formula data.

Future Enhancement

At this time, only the SMFF is supported. Other formats will be supported at a later date.

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (cdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439.

Fragment (syntax fragment)

Purpose

The Fragment element contains a labeled subpart of a syntax definition. Use syntax fragments to organize subparts of a large syntax definition that either appear multiple times or are recursively defined, or parts that are too complicated to appear in place. Fragments are referred to with the FragRef element.

Examples

```
<syntax>
<fragref><title>Common attributes</title></fragref>
<fragment><title>Common attributes</title>
<group optreq="opt" style="bkm:(composite)">
<kwd>ID</kwd>
<oper>=</oper>
<var>identifier</var>
</group>
<group optreq="opt" style="bkm:(composite)">
<kwd>STYLE</kwd>
<oper>=</oper>
<var>style stuff</var>
</group>
</fragment>
</syntax>
```

Attributes

LINKEND=*element_id*

The ID value of the element being linked to or the ID of a NameLoc element.

See “Common Element Attributes (large set)” on page 204.

Usage

See “The FRAGMENT and FRAGREF (fragment reference) element” on page 133.

Contexts

Children: “FragRef (syntax fragment reference)”, “Group” on page 277, “SynNote (syntax note)” on page 418, “Title” on page 431.

Parents: “SynBlk (syntax block)” on page 417, “Syntax (syntax diagram)” on page 420.

FragRef (syntax fragment reference)

Purpose

The FragRef element provides a logical reference to a syntax definition fragment. Use fragment references to create symbolic references to subparts of a syntax definition or to abstract constructs that are not explicitly defined. For example, you can reduce the complexity of a definition by replacing complex subparts with fragment references with meaningful titles. You can also use fragment references to define recursive constructs or to refer to abstract constructs that are not explicitly defined.

Examples

```
<syntax>
<fragref><title>Common attributes</title></fragref>
<fragment><title>Common attributes</title>
<group optreq="opt" style="bkm:(composite)">
<kwd>ID</kwd>
<oper>=</oper>
<var>identifier</var>
</group>
<group optreq="opt" style="bkm:(composite)">
<kwd>STYLE</kwd>
<oper>=</oper>
<var>style stuff</var>
</group>
</fragment>
</syntax>
```

Attributes

OPTREQ=REQ | OPT

Indicates whether the fragment is optional or required. REQ (required) is the default.

FRAGID=fragment_ID

Refers to a Fragment element. When FRAGID is specified, it is an error to specify any content or the FragRef end tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “The FRAGMENT and FRAGREF (fragment reference) element” on page 133.

Contexts

Children: “Title” on page 431.

Parents: “Fragment (syntax fragment)” on page 268, “Group” on page 277, “SynBlk (syntax block)” on page 417, “Syntax (syntax diagram)” on page 420.

FrontCover

Purpose

The FrontCover element contains a reference to the art used for the document's front cover.

Examples

```
<ibmbibentry><doctitle><titleblk>
<title>My Document</title>
</titleblk></doctitle>
<ibmdocnum></ibmdocnum>
<coverdef><frontcover><mobj><objref obj="front1">
<textalt></textalt>
</mobj></frontcover><backcover><mobj>stago.objref obj="back1">
<textalt></textalt>
</mobj></backcover></coverdef></ibmbibentry>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Adding to the front or back cover (CoverDef)” on page 78.

Contexts

Children: “BibList (bibliography entry list)” on page 220, “CGraphic (character graphic)” on page 226, “DL (definition list)” on page 253, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “MMObj (multi-media object; artwork)” on page 333, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Table” on page 423, “UL (unordered list)” on page 436, “Xmp (example)” on page 440.

Parents: “CoverDef (cover definition)” on page 239.

FrontM (front matter)

Purpose

The FrontM element contains the material that precedes the body of a document, such as the preface or table of contents.

Examples

```
<ibmidoc>
<prolog>
...
</prolog>
<frontm style='display="tipage cover spine"'>
...
</frontm>
<body>
...
</ibmidoc>
```

Attributes

STYLE="display='tipage cover spine' "

Sets style items such as title page, covers, and spine. See "Front matter (FrontM)" on page 85 for the values to use.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Front matter (FrontM)" on page 85.

Contexts

Children: "Abbrev (abbreviations)" on page 207, "Abstract (abstract)" on page 207, "Bibliog (bibliography)" on page 219, "D (hierarchical division)" on page 242, "DBlk (Division block)" on page 245, "EdNotices (edition notices)" on page 259, "FigList (list of figures)" on page 263, "Glossary" on page 276, "IBMSafety (IBM safety notices)" on page 291, "Legend" on page 306, "Notices (contains notices)" on page 364, "Preface" on page 387, "RCF (reader comment form)" on page 404, "Safety (safety notices)" on page 411, "SOA (summary of amendments)" on page 413, "TList (list of tables)" on page 432, "TOC (table of contents)" on page 435.

Parents: "IBMIDDoc (IBM-specific product documentation)" on page 281.

GendTitle (default title specification)

Purpose

The GendTitle element causes the system default title for certain specialized divisions to be used at processing time. The GendTitle element has no content. The processing application and document style determine the title that will be generated.

Examples

```
<toc><gendtitle></toc>
```

Attributes

ID The ID attribute identifies an element within an SGML document. IDs must be unique within a single document. Any element that has an ID can be cross-referenced or linked to. IDs can be up to 64 characters long. IDs must start with an alphabetic character and can contain letters, numbers, dashes (-), or periods (.).

Style

The Style attribute contains either a reference to a separate style specification for an element or a set of element-specific presentation style definitions when the processing system does not support separate styles for elements. Assigning a value to the STYLE attribute modifies how an element is presented when it is output.

HyTime

ignored by processes

InfoMast

A fixed attribute used to classify the element.

Contexts

Children: empty.

Parents: "FigList (list of figures)" on page 263, "IBMSafety (IBM safety notices)" on page 291, "Index" on page 293, "PNIndex (part number index)" on page 386, "RCF (reader comment form)" on page 404, "Safety (safety notices)" on page 411, "SpecDProlog (special section division prolog)" on page 415, "TList (list of tables)" on page 432, "TOC (table of contents)" on page 435.

GL (glossary list)

Purpose

The GL element contains glossary entries. A glossary list contains one or more glossary entries (GLEntry), which in turn contain a term and one or more definitions. Entries can be organized within a glossary list using glossary block (GLBlk) elements. Bridge elements can also be used to create transitions or connections between blocks of entries.

Glossary entries can also be used within normal element content to be collected automatically, or placed within document or division prologs when the terms apply to an entire document or a to specific division.

Unlike definition list entries, glossary entries can associate multiple definitions with a single term.

Glossary lists are normally contained by a Glossary division in the BackM.

Examples

```
<backm>
...
<glossary>
<specdprolog><gendtitle></specdprolog>
<dbody>
  <gl>...</gl>
</dbody>
</glossary>
...
</backm>
```

Attributes

SPEC=**AUTO**

Specifies that the content of the element is generated.

LINESPACE=**SPACE** | **COMPACT**

Specifies whether the items in the list should be compacted or have space between the items. Nested lists automatically inherit the setting of the outer list, but can override that default with their own setting.

RETKEY=**None** | **NoDup**

Use the RetKey attribute to enable automatic running headings for glossary lists. NoDup indicates that the first and last non-blank glossary terms on the page are to be used. The two terms are joined together, separated by a bullet or other character, and the combined string is used as the retrieval subject for the page. If the first and last glossary terms on the page are the same, only the last glossary term is displayed in the running heading or footing.

If you code any explicite RetKey elements, they are honored and will appear. If you nest elements that can generate a running heading (for example, a MsgList inside Lers), only the outer active generated heading is used. That is, if you specified automated RetKey generation for LERS and MSGLIST, a Msgno inside Lers will not be used in the RetKey area. But if you had an explicit RetKey inside the Msg, then the RetKey is honored as an explicit override.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 12. Glossaries” on page 115.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “GLBlk (glossary list block)” on page 274, “GLEntry (glossary list entry)” on page 275.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264

+ page 264, “Fn (footnote)” on page 265, “LEDI (language element description item)”
+ on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on
+ page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content
+ description)” on page 343, “ModItem (module description item)” on page 344,
+ “MsgItem (message description item)” on page 348, “NoteBody (note body)” on
+ page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380,
+ “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on
+ page 418, “Warning (warning notice)” on page 439.

GLBlk (glossary list block)

Purpose

The GLBlk element organizes glossary list entries within a glossary list. For example, you can use GLBlk to create logical subdivisions within a long glossary list.

Examples

```
<GL>
<GLBLK>
<TITLE>A</TITLE>
<GENTRY>
<TERM>acopy</TERM>
<DEFN>A transaction program which provides a command line interface to
the APPC File Transfer Protocol (AFTP) facility.
</DEFN>
</GENTRY>
<GENTRY>
<TERM>advanced program-to-program communication (APPC)</TERM>
<DEFN>The general facility characterizing
the LU 6.2 architecture and its
various implementations in products.
</DEFN>
<DEFN>Sometimes used to refer to the LU 6.2
architecture and its product
implementations as a whole, or to an LU 6.2 product feature in
particular, such as an APPC application program interface.
</DEFN>
</GENTRY>
</GLBLK>
</GL>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Separating letter groups in a glossary” on page 116.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “GEntry (glossary list entry)” on page 275, “Title” on page 431.

Parents: “GL (glossary list)” on page 272.

GIDefs (glossary definitions)

Purpose

The GIDefs element contains GLEntrys that can be referred to from other places in the document or division.

Contexts

Children: “GLEntry (glossary list entry)”.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

GLEntry (glossary list entry)

Purpose

The GLEntry element contains a single term and its definition. Use the GLEntry element within a glossary list (GL) to define information.

Examples

```
<gl>
<glentry><term>gopher</term>
<defn>A burrowing rodent that feeds on roots of plants.
</defn>
</glentry>
<glentry><term>lawn</term>
<defn>Gopher highway. <p>Can be identified by dinner-plate-sized
mounds of dirt where grass used to be.</p></defn>
</glentry>
<glentry><term>agapanthus</term>
<defn>Lovely flowering plant, the roots of which are
the preferred food of gophers. <p>If your flourishing
agapanthus suddenly keels over, it means a gopher
has had a feast.</p></defn>
</glentry>
</gl>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Definition lists” on page 26.

Contexts

Children: “Defn (definition of a term)” on page 249, “Term” on page 425.

Parents: “GL (glossary list)” on page 272, “GLBlk (glossary list block)” on page 274, “GIDefs (glossary definitions)”.

Glossary

Purpose

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

Purpose

The DL element contains a list of pairs of terms and definitions. Use definition lists as a generic definition structure for defining things other than glossary terms. Entries can be organized within a definition list using DLBlk elements. Bridge elements can also be used to create transitions or connections between blocks of entries.

Examples

```
<gl>
<glentry><term>gopher</term>
<defn>A burrowing rodent that feeds on roots of plants.
</defn>
</glentry>
<glentry><term>lawn</term>
<defn>Gopher highway. <p>Can be identified by dinner-plate-sized
mounds of dirt where grass used to be.</p></defn>
</glentry>
<glentry><term>agapanthus</term>
<defn>Lovely flowering plant, the roots of which are
the preferred food of gophers. <p>If your flourishing
agapanthus suddenly keels over, it means a gopher
has had a feast.</p></defn>
</glentry>
</gl>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 12. Glossaries” on page 115.

Contexts

Children: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “SpecDProlog (special section division prolog)” on page 415.

Parents: “BackM (back matter)” on page 217, “FrontM (front matter)” on page 270.

Group

Purpose

The Group element defines the syntax group and lets you give the group a name in a Title element. The Title element enables the Group to be automatically fragmented if it is too large to fit the current area.

Examples

```
<syntax>  
<group>  
<kwd>FORM</kwd>  
<kwd>PROC</kwd>  
</group>  
</syntax>
```

Attributes

REPID=*repeat-ID*

Specifies the group repeats. the REPID points to a REPSEP element in the same syntax diagram.

OPTREQ=DEF | REQ | OPT

Indicates whether or not the group is a default, optional, or required. REQ is assumed.

CHOICESEQ=CHOICE | SEQ

Indicates whether the items for this group are choices (you select one of the items) or sequential (you enter each of them in order).

See “Common Element Attributes (large set)” on page 204.

Usage

See “The Group element” on page 128.

Contexts

Children: “Delim (syntax delimiter)” on page 250, “FragRef (syntax fragment reference)” on page 269, “Group”, “Kwd (syntax keyword)” on page 299, “Oper (syntax operator)” on page 371, “Sep (syntactic separator)” on page 412, “SynNote (syntax note)” on page 418, “Title” on page 431, “Var (syntax variable)” on page 437, “XRef (cross reference)” on page 442.

Parents: “Fragment (syntax fragment)” on page 268, “Group”, “SynBlk (syntax block)” on page 417, “Syntax (syntax diagram)” on page 420.

Hex (hexadecimal)

Purpose

Use the Hex element to identify hexadecimal data.

Examples

```
<BIN>1100 0001</BIN> = <HEX>C1</HEX>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See Table 1 on page 36.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439.

IBMBibEntry (IBM bibliographic entry)

Purpose

Use IBMBibEntry to define the bibliographic information about an IBM document.

IBMBibEntry elements can be specified in a BibEntryDefs element or object container and used by reference from within a document, for example, from Cit and Q elements. When IBMBibEntry is specified within Cit, Q, and LQ, the IBMBibEntry elements are collected for use in a generated bibliography.

Contained IBMBibEntry elements and IBMLibEntry elements are normally used within re-used information in order to ensure that the re-used information is completely self-contained. In other words, these elements should be used within the scope of the information that is being re-used. For example, if a Division element is re-used, the IBMBibEntry and IBMLibEntry elements should be contained within that same division's DProlog element. This allows these elements to be completely contained, and thus completely re-usable, along with the division that is being re-used.

Examples

```
<ibmiddoc>
<prolog><ibmbibentry><doctitle>
<library><titleblk>
<title>My Library</title>
</titleblk></library>
<titleblk>
<title>My Document of Interesting Things</title>
<stitle>Interesting things</stitle>
<subtitle>Or, Cool Things I Like</subtitle>
</titleblk></doctitle></ibmbibentry>
```

Attributes

DOCLINK=*ID*

The DocLink attribute specifies the ID of the URL defined on a Notloc element.

DOCNAME=*entity_name*

Contains a reference to the ID or name of an entity that is defined in the document that must also be referenced by a NameLoc element. This indicates a cross-document target with the specified ID value.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Chapter 13. Bibliographies and citations" on page 119.

Contexts

Children: "Authors" on page 216, "CoverDef (cover definition)" on page 239, "Desc (element description)" on page 251, "DocTitle (document title)" on page 256, "ExternalFileName" on page 262, "FileNum (file number)" on page 265, "IBMDocNum (IBM document number)" on page 280, "IBMPartNum (IBM part number)" on page 290, "ISBN (document ISBN number)" on page 295, "OrigIBMDocNum (original IBM document number)" on page 372, "PrtLoc (country where printed)" on page 398, "PublicID (public identifier)" on page 399, "Publisher (document publisher)" on page 400, "RetKey (retrieval key)" on page 407, "VolId (volume identifier)" on page 438.

Parents: “BibEntryDefs (contains bibliographic entries)” on page 219, “BibList (bibliography entry list)” on page 220, “Cit (document citation)” on page 229, “Prolog (document metainformation)” on page 395.

IBMBOFNum (bill of forms number)

Purpose

The IBMBOFNum element contains the IBM Bill of Forms number for the described library.

Examples

```
<bibentrydefs>
  <ibmlibentry>
    <library><titleblk><title>BS/300</title></titleblk>
  </library>
  <ibmbofnum>SB0F-1234-0</ibmbofnum>
</ibmlibentry>
</bibentrydefs>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “IBMLibEntry (IBM document library definition)” on page 287.

IBMDocNum (IBM document number)

Purpose

The IBMDocNum element contains the assigned IBM document number for the document.

Examples

```
<ibmbibentry><doctitle><titleblk>
  <title>My Document</title>
</titleblk></doctitle>
<ibmdocnum>SC99-1234-00</ibmdocnum>
<ibmpartnum>1234F99</ibmpartnum>
</ibmbibentry><ibmproinfo>
  <prodname>My Product</prodname>
  <ibmpgmnum>1234-XX1</ibmpgmnum>
  <ibmfeatnum>1754</ibmfeatnum>
</ibmproinfo>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Document number” on page 76.

Contexts

Children: text (#pcdata), "Ph (Phrase)" on page 382.

Parents: "IBMBibEntry (IBM bibliographic entry)" on page 279.

IBMFeatNum (IBM feature number)

Purpose

The IBMFeatNum element contains the assigned IBM feature number for the document.

Examples

```
<ibmbibentry><doctitle><titleblk>
<title>My Document</title>
</titleblk></doctitle>
<ibmdocnum>SC99-1234-00</ibmdocnum>
<ibmpartnum>1234F99</ibmpartnum>
</ibmbibentry><ibmproinfo>
<prodname>My Product</prodname>
<ibmpgmnum>1234-XX1</ibmpgmnum>
<ibmfeatnum>1754</ibmfeatnum>
</ibmproinfo>
```

Contexts

Children: text (#pcdata), "Ph (Phrase)" on page 382.

Parents: "IBMProdInfo (IBM product information)" on page 291.

Attributes

See "Common Element Attributes (large set)" on page 204.

IBMIDDoc (IBM-specific product documentation)

Purpose

The IBMIDDoc element contains IBM product information. The IBMIDDoc document type is used within IBM to create information deliverables for IBM products. This document type conforms to the IBM InfoMast Architecture and the HyTime standard (ISO/IEC 10744). The IBMIDDoc element contains an entire IBMIDDoc document.

An IBMIDDoc document is divided into four main elements: Prolog, FrontM, Body, and BackM. Only the Body element is required. The Prolog contains all the information that describes the document itself, such as the document title, author, document numbers, property definitions, and so on. The other sections contain the content of the document organized into divisions, either by D elements or by specialized divisions, such as Preface or Bibliog.

Examples

```
<IBMIDDOC SEC="IXM Confidential" LANGUAGE="USEGLISH" "COPYR="ibmprimary" "lotus"
"IMCOPR="1994, 1995" ID="edf10mstv2r1">
...
</IBMIDDOC>
```

Attributes

In addition to support for general attributes, IBMIDDoc can also have several other attributes:

AppPrefix

Controls the automatic generation of the word "Appendix" from the heading text.

DEFAULT-APP

Use the default for this style. (This is the default value.)

TEXT-APP

Add text and the number.

NONE-APP

Do not add any prefix.

NUMONLY-APP

Add the division number as a prefix.

ChapPrefix

Controls the automatic generation of the word "Chapter" from the heading text.

DEFAULT-CHAP

Use the default for this style. (This is the default value.)

TEXT-CHAP

Add text and the number.

NONE-CHAP

Do not add any prefix.

NUMONLY-CHAP

Add the division number as a prefix.

Class

Has a few values for HTML-published information. No entry generates a HTML-like book; running feet connect the pages together; headings link back to the table of contents.

article

Specified HTML article format. A table of contents is generated. The articles have no automatic running footing that connect the pages together. Use this with the HTML frame option to generate a frame-based set of articles with a "twisty" table of contents.

articles

Also specifies HTML article format. No table of contents is generated.

CLASSIF= CONFRES | RES | LIC

Identifies the classification of restricted materials.

CONFRES

Confidential restricted material

RES

Restricted material

LIC

Licensed material

CLASSIF=LIC for the style TIV8x11, causes that Tivoli style to include the licensed statement on each page and on the cover.

Copyr=reference-ID

References the ID for a copyright definition defined in an IBM BibEntry element.

DOCSTYLE=document-style

Specifies the style of the document. You can specify these styles:

IBM8X11

8-1/2 by 11 inch style. Replaces BookMaster style IBMXAGD.

IBM7X9

7 by 9 inch style. Replaces BookMaster style IBMXGGD.

IBM2COL

8.5x11 style (2 column layout)

IBMCD

4.75x4.75 style (for CD Jewel Case booklets)

IBMREFC

Reference cards (3-5/8x9in.).

IBM5X8

5.5x8.5 style (for hardware).

IBM4X6

4.25x6.25 style (for hardware).

IBM8X5

5.5x8.5 landscape style (for hardware).

IBM9X7

7x9 landscape style.

If you create a PDF from this style, the pages may switch between landscape and portrait presentation in Adobe Acrobat Reader or Exchange. Add the following lines to your PostScript file before distilling it to prevent this from occurring:

```
/currentdistillerparams where {pop}
{userdict /currentdistillerparams {1 dict} put} ifelse
/setdistillerparams where {pop}
{userdict /setdistillerparams {pop} put} ifelse
<< /AutoRotatePages /All >> setdistillerparams
```

IBMLAND

Printer System's landscape books. (Not for BookMaster)

IBMXAGD

User Guides (8.5x11in., A4); old BookMaster style.

IBMXARF

Reference (8.5x11in., A4); old BookMaster style.

IBMXGGD

Summary Guides (7-3/8x9in.); old BookMaster style.

TIV7X9

7x9 style for Tivoli

TIV8X11

8.5x11 style for Tivoli

OBIPORT

5.5x8.5 style (for Options by IBM)

OBIWWA6P

4.25x5.75 style (for Options by IBM)

SMALLFLG

3.625x8.5 style (for hardware)

IBMCopyr=*current-year* | *first-year, current-year*

Specifies the copyright date year for IBM publications. You enter either one date 1999 or two 1999, 2000.

IBMSEC=UNC | **IC**

Specifies the IBM security classification for the document. Note that you should use the SEC attribute instead of IBMSEC, with the security classification typed out. If you specify both IBMSEC and SEC, the SEC attribute is used.

unc

Unclassified

ic

IBM Confidential

Language

Specifies the language in which the document is written.

The valid values for the Language attribute on IBMIDDoc element are:

- ENGLISH, en_US, or USENGLISH
- UKENGLISH or en_GB
- DUTCH or nl_NL
- GERMAN or de_DE
- ITALIAN or it_IT
- FRENCH or fr_FR
- SPANISH or es_ES
- PORTUGUESE or pt_PT
- DANISH or da_DK
- FINNISH or fi_FI
- NORWEGIAN or no_NO
- SWEDISH or sv_SE
- CFRENCH or fr_CA
- BFRENCH or fr_BE
- BDUTCH or nl_BE
- BPORTUGUESE or pt_BR
- CENGLISH or en_CA
- ICELANDIC or is_IS
- SGERMAN or de_CH
- SFRENCH or fr_CH
- SITALIAN or it_CH
- KOREAN or ko_KR
- TCHINESE or zh_TW
- SCHINESE or zh_CN
- JAPANESE or ja_JP
- CATALAN or ca_ES
- TURKISH or tr_TR
- GREEK or el_GR

- POLISH or pl_PL
- CZECH or cs_CZ
- SLOVAK or sk_SK
- HUNGARIAN or hu_HU
- CROATIAN or hr_HR
- SLOVENIAN or sl_SI
- RUSSIAN or ru_RU
- ROMANIAN or ro_RO
- BULGARIAN or bg_BG
- ESTONIAN or et_EE
- LATVIAN or lv_LV
- LITHUANIAN or lt_LT
- MACEDONIAN or mk_MK
- SERBIAN or sr_SP

MAXTOC=*number*

Specifies the maximum heading level to be included in the table of contents. The default for the style IBM8X11 is 3. Specifying maxtoc=4 will include divisions to heading level 4.

MULTIVOL=OneVol | Index-Folio

This indicates whether the book is part of a multiple-volume set. Specifying "Index-Folio" adds "X-" as a prefix for the page numbers in the index and starts the page numbering from 1.

PageNumber

There are three options for the PageNumber attribute: FBC, SEQ, and Default-Folio.

SEQ

Sequential page numbering: The front matter page numbers use roman numerals. The body and back matter use arabic numerals.

Note: The 3.2 GA version of the Xyvision code will only support PageNumber=Seq.

FBC

Folio-by-Chapter numbering: The front matter page numbers use roman numerals, body and appendixes add the chapter or appendix number as a prefix and restart the page number at every new level-1 division. For example, chapters and sections are numbered 1-1, 1-2, 2-1, 2-2, etc. Appendixes are numbered A-1, A-2, etc. Non-appendix back matter sections are numbered X-1, X-2, etc. and are not reset at new level-1 divisions.

Default-Folio

This is the default value for PageNumber. Use the default page numbering style for this document style. Currently, all document styles default to SEQ.

PartPrefix

Controls the automatic generation of the word "Part" in the heading text.

DEFAULT-PART

Use the default for this style. (This is the default value.)

TEXT-PART

Add text and the number.

NONE-PART

Do not add any prefix.

NUMONLY-PART

Add the division number as a prefix.

SEC

Specifies the security classification. Note that you should use the SEC attribute instead of IBMSEC, with the security classification typed out. If you specify both IBMSEC and SEC, the SEC attribute is used.

STYLE=*overrides*

This allows specific style overrides.

keepblanks

In non literal-text elements (everything except xmp, lines, litdata, screen, and cgraphic), if you have a space immediately before the ending tag, that space is normally removed by the processing transforms. If you have a **lot** of these spaces and you do not want them removed, you can specify this attribute to keep the blanks. Note that coding this could result in spaces being added between words or extra lines being generated in hardcopy and other types of output.

As an example, the default for this markup is to remove the blanks inside the phrase tags. So this markup:

```
Hi <ph style="bold"> there </ph> handsome.
```

Effectively becomes this when the output is formatted:

Hi **there** handsome.

Through migration from BookMaster or other means, several times the phrase tags were done this way:

```
Hi<ph style="bold"> there </ph>handsome.
```

With the default to remove the blanks, the text contatenates like this:

Hi **there**handsome.

With the keepblanks setting, the text has extra spaces like this:

Hi **there** handsome.

Here is an example example of when the keepblanks setting could cause a problem:

```
<p>Text followed by a screen.  
<screen>  
Screen contents  
</screen> </p>
```

With the default setting to remove the blanks, the space between the </screen> and </p> is removed. With the keepblanks setting, the space remains. This can cause continued paragraph in BookMaster output and a potential “bad” blank in Windows help.

xpp:(bookmarks)

Causes the bookmarks in Acrobat PDF files to match the table of contents. This creates a most excellent way of navigating the PDF. This is now the default setting.

| **xpp:(justify)**

| For DBCS languages only, this causes the formatting for flowed text
| items to be left and right-justified. The opposite setting is `nojustify` or
| `ragged`.

| **MLSPrefix=YES | NO**

| Indicates the document is part of a multiple-language safety book. The page
| number prefixes are determined by the document's language attribute.

| **BRAND**

| Specifies the type of product identification branding to be used for the
| document.

| **DefaultBrand**

| This is the default; no special branding information is produced.

| **eserver-white**

| For the IBM @server brand, this specifies black cover text on a white
| background.

| **eserver-black**

| For the IBM @server brand, this specifies white cover text on a black
| background.

| **NewBrand**

| Indicates the value of the NEWBRAND attribute should be used.

| **NEWBRAND=*brand-name***

| This is where you would enter a special brand name; as defined by the ID
| Workbench team, to handle a future brand in the middle of a release.

| **UNMSPACE=Separate | unify**

| Currently not used.

| **ID** Allows you to assign an identifier to the entire document.

| **Company**

| Specifies the company for non-IBM trademarks.

| **DTDVersion**

| A fixed attribute that indicates the level of the DTD.

Contexts

Must occur in same entity (file) as the IBMIDDoc document type declaration (DTD) and must be highest-level element in the document. Children: "BackM (back matter)" on page 217, "Body (document body)" on page 222, "FrontM (front matter)" on page 270, "Prolog (document metainformation)" on page 395.

Parents:.

IBMLibEntry (IBM document library definition)

Purpose

The IBMLibEntry element contains an IBM-specific library definition. Library entries contain information about a library or collection of documents.

Use IBMLibEntry to define the bibliographic information about a library or other collection of IBM documents. You can use the Class attribute and the ClassDef element to define different classes of LibEntry to correspond to different classes of collection. For example, you may have product libraries that are themselves

collected into larger libraries of libraries. You could define a class of "CollectionKit" for libraries of libraries and then use IBMLibEntry elements to define the contents of a given collection kit.

IBMLibEntry can also be specified in a BibEntryDefs section or ObjLib and used by reference from within a document, for example, from Cit elements.

When IBMLibEntry elements are specified within Cit. the IBMLibEntry elements are collected for use in a generated bibliography.

Contained IBM BibEntry elements and IBMLibEntry elements are normally specified within re-used information in order to ensure that the re-used information is completely self-contained. In other words, these elements should be used within the scope of the information that is being re-used. For example, if a Division element is re-used, the IBM BibEntry and IBMLibEntry elements should be contained within that same division's DProlog element. This allows these elements to be completely contained, and thus completely re-usable, within the division that is being re-used.

Examples

```
<IBMLIBENTRY ID=IBMIDDocLIB>
<LIBRARY>IBMIDDoc Library</LIBRARY>
<PUBLISHER>IBM Corporation
  <ADDRESS>
  </ADDRESS>
</PUBLISHER>
<PRTLOC>USA
<IBMBOFNUM>SBOF-6000-00
<PUBLICID>+//ISBN 0-19-9999
//LIB SBOF-6000-00
/IBMIDDoc Library
/rickd@nando.net
/Networking Software
//EN
<CONTAINEDDOCS BIBIDS="IBMIDDocUG IBMIDDocTUT">
<DESC>Documentation for the IBMIDDoc language.
IBMIDDoc is an SGML language for creating printed and online
technical information.
</IBMLIBENTRY>
```

Attributes

Library

The library name.

Publisher

Contains the name of the publisher followed by an optional address element.

PrtLoc

Contains the name location where the document was printed.

IBMBOFNum

The IBM bill of forms (BOF) number assigned to this library.

IBMPartNum

The IBM part number assigned to this library.

ProdName

The product name with which this library is associated.

ISBN

The ISBN number assigned to this library.

PublicID

The SGML public identifier assigned to this library. This is the same public identifier used in entity declarations for the system object that represents this library.

ContainedDocs

Defines the documents contained in this library and the default order for presenting the documents when the library definition is presented.

The contained documents can be identified directly using Cit elements, or indirectly by specifying BibEntry or LibEntry IDs.

The order the contained documents are specified in ContainedDocs defines the default order for presenting the library contents.

Desc

Contains a description of the library.

Contexts

Children: "ContainedDocs (documents in IBMLibEntry and LibEntry)" on page 236, "Desc (element description)" on page 251, "IBMBOFNum (bill of forms number)" on page 280, "IBMPartNum (IBM part number)" on page 290, "ISBN (document ISBN number)" on page 295, "Library" on page 314, "ProdName (product name)" on page 394, "PrtLoc (country where printed)" on page 398, "PublicID (public identifier)" on page 399, "Publisher (document publisher)" on page 400.

Parents: "BibEntryDefs (contains bibliographic entries)" on page 219, "BibList (bibliography entry list)" on page 220, "Cit (document citation)" on page 229.

IBMMail (IBMMail e-mail address)
Purpose

The IBMMail element contains an IBMMail email address.

Examples

```
<IBMIDDOC>
<PROLOG>

:
<OWNERS>
<CORP>
  <CORPNAME>IBM CORPORATION</CORPNAME>
  <ADDRESS>INFORMATION DEVELOPMENT
    IBM RTP
    DEPT. E14D
    500/D162
    ATTN: Rick Dennis
  <INTERNET>rickd@nando.net</INTERNET>
  <PHONE>919-254-4062</PHONE>
  <VNET>RICKD@RTPNOTES</VNET>
  <IBMMAIL>IBMMail Exchange: USIBM3345 at IBMMAIL</VNET>
  <POSTALCODE>27614</POSTALCODE>
</ADDRESS>
</CORP>
</OWNERS>
<MAINTAINER>
<PERSON>
```

```

        <NAME>Rick Dennis</NAME>
    </PERSON>
</MAINTAINER>
</PROLOG>

```

Attributes

#PCDATA

Contains the IBMMail email address.

Contexts

Children: text (#pcdata).

Parents: “Address (address)” on page 208.

IBMPartNum (IBM part number)

Purpose

The IBMPartNum element contains the IBM part number of the document.

Examples

```

<ibmbibentry><doctitle><titleblk>
<title>My Document</title>
</titleblk></doctitle>
<ibmdocnum>SC99-1234-00</ibmdocnum>
<ibmpartnum>1234F99</ibmpartnum>
</ibmbibentry><ibmprodinfo>
<prodname>My Product</prodname>
<ibmpgmnum>1234-XX1</ibmpgmnum>
<ibmfeatnum>1754</ibmfeatnum>
</ibmprodinfo>

```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMLibEntry (IBM document library definition)” on page 287.

IBMPgmNum (IBM program number)

Purpose

The IBMPgmNum element contains the IBM program number of an IBM program product that is described by the document.

Examples

```

<ibmbibentry><doctitle><titleblk>
<title>My Document</title>
</titleblk></doctitle></ibmbibentry>
<ibmprodinfo>

```

```
<prodname>My Product</prodname>
<ibmpgmnum>1234-XX1</ibmpgmnum>
<ibmfeatnum>1754</ibmfeatnum>
</ibmprodinfo>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using IBMProdInfo” on page 79.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “IBMProdInfo (IBM product information)”.

IBMProdInfo (IBM product information)

Purpose

The IBMProdInfo element contains information about an IBM product that is associated with the document.

Examples

```
<ibmbibentry><doctitle><titleblk>
<title>My Document</title>
</titleblk></doctitle></ibmbibentry>
<ibmprodinfo>
<prodname>My Product</prodname>
<ibmfeatnum>1754</ibmfeatnum>
</ibmprodinfo>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using IBMProdInfo” on page 79.

Contexts

Children: “IBMFeatNum (IBM feature number)” on page 281, “IBMPgmNum (IBM program number)” on page 290, “ModLvl (modification level)” on page 346, “ProdName (product name)” on page 394, “Release (product release identifier)” on page 406, “Version (product version number)” on page 437.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

IBMSafety (IBM safety notices)

Purpose

The IBMSafety element is designed to contain IBM-specific safety notices about safe hardware practices. This is not yet implemented.

Attributes

SPEC=AUTO|MAN

Specifies that the content of the element is generated. SPEC=AUTO is the default value, and causes the appropriate generated text to be included.

See “Common Element Attributes (large set)” on page 204.

Examples

```
<frontm>
<ibmsafety spec="auto"><gendtitle></ibmsafety>
...
</frontm>
```

Contexts

Children: “GendTitle (default title specification)” on page 272, “RetKey (retrieval key)” on page 407, “TitleBlk (title information)” on page 432.

Parents: “FrontM (front matter)” on page 270.

IdxDefs (central index entries)

Purpose

The IdxDefs element contains central index entries for the document or division.

Examples

```
<prolog><ibmbibentry><doctitle><titleblk>
<title>Index test</title>
</titleblk></doctitle></ibmbibentry>
<idxdefs>
<i1 id="becha"><idxterm>a bechamel sauce</idxterm></i1>
<i1 seeid="becha"><idxterm>a white sauce</idxterm></i1>
</idxdefs></prolog>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Defining index entries (central indexing)” on page 102.

Contexts

Children: “I1 (primary index entry)” on page 296, “I2 (secondary index entry)” on page 297, “I3 (tertiary index entry)” on page 298, “IRef (index entry reference)” on page 294.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document meta-information)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

IdxTerm (index term)

Purpose

The IdxTerm element contains a term that is to be included in the index.

Examples

```
<i1><idxterm>dessert  sauces</idxterm></i1>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 10. Indexing” on page 97.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “ClassDef (element class definition)” on page 230, “I1 (primary index entry)” on page 296, “I2 (secondary index entry)” on page 297, “I3 (tertiary index entry)” on page 298.

Index

Purpose

The Index element contains a title for the index, if one is specified.

The normal use of Index is to contain an index that is automatically generated from the index entries within the document content.

Contexts

Children: “GendTitle (default title specification)” on page 272, “RetKey (retrieval key)” on page 407, “TitleBlk (title information)” on page 432.

Parents: “BackM (back matter)” on page 217.

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

SPEC=AUTO | MAN

Specifies that the content of the element is generated. SPEC=AUTO is the default value.

LAYOUT=Default-Layout | OneCol | TwoCol | ThreeCol

Specifies the column-style for this portion of the book.

- + **OneCol**
- + The entries format across the entire page.
- + **TwoCol**
- + The entries format in two columns.
- + **ThreeCol**
- + The entries format in three columns.

+ See “Common Element Attributes (large set)” on page 204.

+ **Usage**

+ See “Generating the index” on page 105.

+ **Examples**

+ `<backm>`
 + `<index>`
 + `<gendtitle>`
 + `</index>`
 + `</backm>`

| **Internet (internet e-mail address)**

| **Purpose**

| The Internet element contains an Internet email address.

| **Examples**

| `<address>`
 | ATTN Dept 245
 | 3605 Hwy 52 N
 | Rochester MN
 | `<postalcode>55901-9986</postalcode>`
 | `<internet>fred@us.ibm.com</internet>`
 | `</address>`

| **Attributes**

| See “Common Element Attributes (large set)” on page 204.

| **Contexts**

| Children: text (#pcdata).

| Parents: “Address (address)” on page 208.

| **IRef (index entry reference)**

| **Purpose**

| The IRef element associates an element with an index entry by referring to an
 | index entry defined elsewhere in the document, either in content or in an IdxDefs
 | element.

| Index entries specified in the information content are normally used to generate a
 | traditional index. When index entries are specified in an IdxDefs element within
 | Prolog or DProlog, they define index structures that can be used by reference from

within the document content. Index entries within IdxDefs will not appear in a generated index unless specifically referred to.

Examples

```
<i1><idxterm>sauces</idxterm>
<i2 id="mayo"><idxterm>mayonnaise</idxterm></i2></i1>
...
<iref refids="mayo">
...
<iref refids="mayo">
```

Attributes

REFIDS=*index_entry_ids*

Refers to one or more index entries (I1, I2, or I3) to be associated with the element that contains this IRef.

PRIMARY=**PRIMARY**

Indicates that this entry is a primary entry for the term. The primary term is usually given some form of emphasis. There should be only one primary entry for each term.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Cross referencing index entries” on page 100.

Contexts

Children: empty.

Parents: “IdxDefs (central index entries)” on page 292.

ISBN (document ISBN number)

Purpose

The ISBN element contains a document’s ISBN number.

Examples

```
<P>To better understand the intricacies of SGML, see
<CIT>
<BIBENTRY>
<DOCTITLE>
<TITLEBLK>
<TITLE>The SGML Handbook</TITLE>
</TITLEBLK>
<AUTHOR>
<NAME>Charles F. Goldfarb
</NAME>
</AUTHOR>
<PUBLISHER>
<CORPNAME>
Oxford University Press
</CORPNAME>
<ADDRESS>
Walton St
Oxford OX2 6DP
</ADDRESS>
```

```

</PUBLISHER>
<PRTLOC>Printed and Bound in Great Britain
<ISBN>0-19-835737-9
<PUBID>+//ISBN 0-19-853737-9//DOCUMENT The SGML Handbook//EN
</BIBENTRY>
</CIT>
for more information.

```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

For more information about the ISBN element, see “An example of using BibEntry and BibEntryDefs” on page 122.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “BibEntry (bibliographic entry)” on page 218, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

I1 (primary index entry)

Purpose

The I1 element contains an index entry and related secondary or tertiary (third-level) entries. An index entry is associated with the element that directly contains it.

Index entries specified in the information content will be used to generate a traditional index. When index entries are specified in an IdxDefs element within Prolog or DProlog, they define index structures that can be used by reference from within the document content. Index entries within IndexDefs will not appear in a generated index unless specifically referred to.

Examples

```
<i1><idxterm>dessert sauces</idxterm></i1>
```

Attributes

SEEID=*i1_ids*

Defines a SEE or SEE ALSO reference for cross-referencing.

SEETEXT=*see_also_text*

Contains the text of a see or see also reference. For example, under an I1 entry of Poultry, you can use SEETEXT="Chicken, Turkey, Quail, Duck, and Goose". When you specify the SEETEXT attribute, you must ensure that there are index entries for each word or phrase mentioned in the SEETEXT content. SEEID and SEETEXT can both be specified, if desired. SEEID takes precedence over SEETEXT.

SORTKEY=*sortkey text*

When specified, the SortKey= text is used to sort the entry, rather than the index entry text itself.

| **PRIMARY=Primary**

| Indicates that this entry is a primary entry for the term. The primary term is
| usually given some form of emphasis. There should be only one primary entry
| for each term.

| See “Common Element Attributes (large set)” on page 204.

| **Usage**

| See “Chapter 10. Indexing” on page 97.

| **Contexts**

| Children: “I2 (secondary index entry)”, “IdxTerm (index term)” on page 293.

| Parents: “IdxDefs (central index entries)” on page 292.

| **I2 (secondary index entry)**

| **Purpose**

| The I2 element contains an index entry and any related tertiary (third-level) entries.
| An index entry is associated with the element that directly contains it. I2 outside
| the context of I1 must use the I1ID attribute to refer to an I1 element.

| Index entries specified in the information content will be used to generate a
| traditional index. When index entries are specified in an IdxDefs element within
| Prolog or DProlog, they define index structures that can be used by reference from
| within the document content. Index entries within IndDefs will not appear in a
| generated index unless specifically referred to.

| **Examples**

| <i1><idxterm>dessert sauces</idxterm>
| <i2><idxterm>butterscotch</idxterm></i2>
| <i2><idxterm>hot fudge</idxterm>
| <i3><idxterm>microwave method</idxterm></i3>
| <i3><idxterm>stovetop method</idxterm></i3>
| </i2>
| <i2><idxterm>strawberry</idxterm></i2>
| </i1>

| **Attributes**

| **I1ID=*i1 id***

| Refers to the first level entry for this second level entry. I1ID is required when
| I2 occurs outside the context of an I1 element.

| **SEEID=*i1 ids***

| Defines a cross reference to one or more other first-level index entries.

| **SEETEXT=*see also text***

| Contains the text of a see also reference. SEEID and SEETEXT can both be
| specified, if desired.

| **SORTKEY=*sortkey text***

| When specified, the SORTKEY text is used to sort the entry, rather than the
| index entry text itself.

PRIMARY

Indicates that this entry is a primary entry for the term. The primary term is usually given some form of emphasis. There should be only one primary entry for each term.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 10. Indexing” on page 97.

Contexts

Children: “I3 (tertiary index entry)”, “IdxTerm (index term)” on page 293.

Parents: “I1 (primary index entry)” on page 296, “IdxDefs (central index entries)” on page 292.

I3 (tertiary index entry)

Purpose

The I3 element contains a third-level index entry. An index entry is associated with the element that directly contains it. I3 outside the context of I2 must use the I2ID attribute to refer to an I2 element.

Index entries specified in the information content will be used to generate a traditional index. When index entries are specified in an IdxDefs element within Prolog or DProlog, they define index structures that can be used by reference from within the document content. Index entries within IndexDefinition will not appear in a generated index unless specifically referred to.

Examples

```
<i1><idxterm>dessert sauces</idxterm>
<i2><idxterm>butterscotch</idxterm></i2>
<i2><idxterm>hot fudge</idxterm>
<i3><idxterm>microwave method</idxterm></i3>
<i3><idxterm>stovetop method</idxterm></i3>
</i2>
<i2><idxterm>strawberry</idxterm></i2>
</i1>
```

Attributes

I2ID=*i2 id*

Refers to the first level entry for this second level entry. I2ID is required when I3 occurs outside the context of an I2 element.

SORTKEY=*sortkey text*

When specified, the SORTKEY text is used to sort the entry, rather than the index entry text itself.

PRIMARY=**Primary**

Indicates that this entry is a primary entry for the term. The primary term is usually given some form of emphasis. There should be only one primary entry for each term.

Usage

See “Chapter 10. Indexing” on page 97.

Contexts

Children: “IdxTerm (index term)” on page 293.

Parents: “I2 (secondary index entry)” on page 297, “IdxDefs (central index entries)” on page 292.

Kwd (syntax keyword)

Purpose

Use Kwd to define keywords within a syntax definition. Keywords are literal values that must be specified exactly as shown in the diagram.

Examples

```
<syntax>
<group>
<kwd>FORM</kwd>
<kwd optreq="opt">PROC</kwd>
</group>
</syntax>
```

Attributes

ABBREVS=*abbreviations*

Lists the valid blank delimited abbreviations for the keyword.

OPTREQ=REQ | OPT | DEF

Indicates whether or not the keyword is optional. REQ (required) is the default.

LINKEND=*id*

Contains an ID reference that enables a link to an arbitrary location in the document.

See “Common Element Attributes (large set)” on page 204.

Usage

See “The KWD (keyword) element” on page 130.

Contexts

Children: text (#pcdata).

Parents: “Group” on page 277, “SynPh (syntax phrase)” on page 419.

L (explicit link)

Purpose

The L element links a phrase to any place in a document, another document, or a non-text object, such as a multimedia presentation.

L can point to either another element in the same document, or it can point to a NameLoc element, through which it can link to almost anything. What you can link to via NameLoc is limited only by the online presentation system you use.

Note: IBMIDDoc neither defines nor limits the types of things you can link to from a document. The linking you can do is determined by the online presentation system you are using.

As a rule, it is worth using NameLoc when something will be linked to more than twice within the same document, because the indirection provided by NameLoc makes maintaining those links easier.

Examples

```
<d id="xrefhyl">
<dprolog><titleblk>
<title>All about linking</title>
</titleblk></dprolog>
<dbody>
<p>Hypertext links (we'll just call them links from
now on) connect elements in one part of an online
document to elements in another part of the same document
or a separate online document. </p>
...
<p>Sometimes you need to <l linkend="xrefhyl">link</l> to
other topics.</p>
```

Attributes

LINKEND=*element_id*

The ID value of the element being linked to or a NameLoc element that ultimately locates the object or objects being linked to.

SPEC=**AUTO**

When **AUTO** is specified, the link behaves like a cross reference and the link text is generated automatically. The generated text for the hotspot is dependent upon the target element.

CLASS

You can use this to affect whether that link will replace the content of a frame; or whether the link will launch a new browser window.

NewWindow

This opens the link in a new, unnamed window. This is the same as the HTML coding: target="_blank"

FullWindow

This opens the link into the full, original window, cancelling all frames. This is the same as the HTML coding: target="_top"

SameWindow

This opens the link into the same window. This is the same as the HTML coding: target="_self"

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 11. All about linking” on page 109.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “Formula (math formula)” on page 267, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “MD (marked deletion)” on page 326, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term” on page 425, “TM (Trademark)” on page 433, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “Address (address)” on page 208, “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Cap (caption)” on page 225, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “CI (component item)” on page 228, “CLE (content list entry)” on page 231, “CompCmt (component comment)” on page 234, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DefnHd (definition description heading)” on page 250, “Desc (element description)” on page 251, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “EdNotices (edition notices)” on page 259, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LeDesc (language element description)” on page 304, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “ModName (modular information element name)” on page 346, “MsgItem (message description item)” on page 348, “MsgText (message text)” on page 354, “NItem (notice item)” on page 359, “NoteBody (note body)” on page 362, “Notices (contains notices)” on page 364, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “Q (quotation phrase)” on page 402, “Safety (safety notices)” on page 411, “Screen (display screen)” on page 411, “Sem (semantic meaning)” on page 412, “STitle (shortened title)” on page 416, “SubTitle (descriptive subtitle)” on page 417, “SynNote (syntax note)” on page 418, “Term” on page 425, “TermHd (term heading)” on page 426, “TextAlt (text alternative)” on page 427, “Title” on page 431, “Warning (warning notice)” on page 439, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441.

LDescs (link descriptions)

Purpose

Use LDescs to contain descriptions of links that need to be referenced from more than one place in the document.

Examples

```
<PROLOG>
<IBMBIBENTRY ID="BOOKMSG">
  <DOCTITLE><TITLEBLK><TITLE>IBM BOOKMASTER USER'S GUIDE </TITLE>
</TITLEBLK>
</DOCTITLE>
<IBMDOCNUM>SC34-5107</IBMDOCNUM>
  <DESC>DESCRIBE HOW TO USE BOOKMASTER</DESC>
</IBMBIBENTRY>
<LDESCS>
  <NAMELOC ID="ABC1" OBJTYPE="BOOK">
    <NMLIST></NMLIST>
  </NAMELOC>
  <NAMELOC ID="ABC0" OBJTYPE="HEAD">
    <NMLIST DOCNAME="BOOKUG">SYMB</NMLIST>
  </NAMELOC>
</LDESCS>
<BIBENTRYDEFS>
  <IBMBIBENTRY ID="BOOKUG">
    <DOCTITLE><TITLEBLK><TITLE>TITLE</TITLE></TITLEBLK>
  </DOCTITLE>
  <IBMDOCNUM>SN23-0059</IBMDOCNUM>
  <DESC>BOOK DESCRIPTION</DESC>
</IBMBIBENTRY>
</BIBENTRYDEFS>
</PROLOG>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using LDescs and NameLoc” on page 82.

Contexts

Children: “AreaDef (defines graphic hot spot area)” on page 213, “NameLoc (named location)” on page 357, “Notloc (notation-specific location)” on page 365.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document meta-information)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

LE (language element)

Purpose

The LE element contains the description of a computer language element, such as a command, within the context of a language element reference section. Use LE and LERS to create reference information for computer languages such as command sets, programming languages, and the like. The presentation of language elements is as headed sections.

Examples

```
<LERS>
  <LE>
    <LEN>aname
  </LEN>
  <LEDI CLASS="PURPOSE">
    <P>Use this command to request help information on the APPC NameServer
      facility.
  <LEDI CLASS="FORMAT">
    <SYNTAX>
      <GROUP>
        <KWD OPTREQ="REQ">aname
        <KWD OPTREQ="REQ"> -h
      </GROUP>
    </SYNTAX>
  <LEDI CLASS="PARMS">
    <PARML STYLE="BKM:(BREAK='NONE' TSIZE='&bigt.')">
      <PARM>
        <TERM>-h</TERM>
        <DEFN>An explicit request for help information on the ANAME command.
      </DEFN>
    </PARML>
  <LEDI CLASS="EXAMPLES">
    <P>This example requests general help on the ANAME command.
    <XMP>
      aname -h
    </XMP>
  </LE>
</LERS>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 15. Developing Programming Language Reference Materials” on page 141.

Contexts

Children: “LeDesc (language element description)” on page 304, “LEDI (language element description item)” on page 304, “LEN (language element name)” on page 307, “RetKey (retrieval key)” on page 407.

Parents: “LERS (language element reference section)” on page 308.

LeDesc (language element description)

Purpose

LeDesc can contain the description of the command. This is usually a short abstract. The LEDI element's Purpose class should be used to contain the main purpose of the language element.

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See "Chapter 15. Developing Programming Language Reference Materials" on page 141.

LEDI (language element description item)

Purpose

The LEDI element contains a description of one aspect of a language element within a LERS section. Each language element can have several LEDI elements.

For each LEDI, the title text can be defined separately on a LERSDEF element. Each LEDI class can have a different generated title or no title. The default presentation style for language element descriptions is as normal divisions.

Examples

```
<LERS>
  <LE>
    <LEN>aname
  </LEN>
  <LEDI CLASS="PURPOSE">
    <P>Use this command to request help information on the APPC NameServer
      facility.
  </LEDI CLASS="FORMAT">
    <SYNTAX>
      <GROUP>
        <KWD OPTREQ="REQ">aname
        <KWD OPTREQ="REQ"> -h
      </GROUP>
    </SYNTAX>
  <LEDI CLASS="PARMS">
    <PARML STYLE="BKM:(BREAK='NONE' TSIZE='&bigt.')">
      <PARM>
        <TERM>-h</TERM>
        <DEFN>An explicit request for help information on the ANAME command.
      </DEFN>
    </PARML>
  <LEDI CLASS="EXAMPLES">
    <P>This example requests general help on the ANAME command.
    <XMP>
      aname -h
    </XMP>
  </LE>
</LERS>
```

Attributes

Class Values

Indicates the class of the LEDI. The class values define what type of information each LEDI contains:

AUTH

Authorization level; for example, user or superuser.

COMMENTS

Comments about the language element.

CONTEXT

The context in which the language element is used.

DEFAULTS

The defaults for the language element.

ERRCOND

Error conditions.

EXAMPLES

Examples of using the language element.

FLAGS

Control flags.

FORMAT

The format or syntax of the language element.

INTREP

Internal representation, such as control blocks or data structures.

MESSAGES

Any messages related to the language element.

OTHER

An open category.

PARMS

Parameters.

PROCESS

Processing related to the language element.

PURPOSE

The purpose of the language element.

RESTRICT

Restrictions on the use of the language element.

RESULTS

The results of using the language element.

RETCODES

Return codes.

SYSENV

The system environments to which the language element applies.

USAGE

How to use the language element.

VERSION

Any version information for the language element.

See “Using LDescs and Nameloc” on page 82.

Usage

See “Chapter 15. Developing Programming Language Reference Materials” on page 141.

Contexts

Children: “Annot (annotation)” on page 209, “AsmList (list of parts assemblies)” on page 214, “Attention (safety notice)” on page 214, “BibList (bibliography entry list)” on page 220, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “D (hierarchical division)” on page 242, “Danger (danger notice)” on page 243, “Dblk (Division block)” on page 245, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “FNList (footnote list)” on page 266, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “LERS (language element reference section)” on page 308, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MarkList (marked note list)” on page 321, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MsgList (list of message or code descriptions)” on page 352, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PartAsm (part assembly)” on page 379, “Pblk (paragraph block)” on page 380, “Proc (procedure)” on page 388, “Screen (display screen)” on page 411, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “UL (unordered list)” on page 436, “Xmp (example)” on page 440.

Parents: “LE (language element)” on page 302.

Legend

Purpose

The Legend element contains an explanation of any special notations used in the document, such as within graphics, tables, figures, or other specialized information.

Examples

```
<FRONTM>
<LEGEND>
  <SPECDPROLOG>
    <GENDTITLE>
  </SPECDPROLOG>
  <DBODY>
    <P>The following symbols have special meaning in this document:

    :
  </DBODY>
</LEGEND>
</FRONTM>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level

(such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Special sections” on page 88.

Contexts

Children: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “SpecDProlog (special section division prolog)” on page 415.

Parents: “FrontM (front matter)” on page 270.

LEN (language element name)

Purpose

Use LEN for the name of the language element, for example, the command name.

Examples

```
<le id="len">
<len>LEN (language element name)</len>
<ledi class="PURPOSE">
<p>Use LEN for the name of the language element, for
example, the command name.</p>
</ledi>
<ledi class="EXAMPLES">
<xmp></xmp>
</ledi>
<ledi class="PARMS">
<p conloc="commattr">
</ledi>
<ledi class="USAGE">
<p>See <xref refid="langref">.</p>
</ledi>
<ledi class="CONTEXT"><pblk conloc="context_len">
</ledi>
</le>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 15. Developing Programming Language Reference Materials” on page 141.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382, “TM (Trademark)” on page 433.

Parents: “LE (language element)” on page 302.

LERS (language element reference section)

Purpose

Use LERS to contain reference information for computer languages such as programming languages, command sets, and the like. Use the more generic ModInfo elements for other sorts of modular information. A LERS section contains one or more language element descriptions (LE). The default presentation style for LERS is as normal divisions.

Examples

```
<le id="len">
<len>LEN (language element name)</len>
<ledi class="PURPOSE">
<p>Use LEN for the name of the language element, for
example, the command name.</p>
</ledi>
<ledi class="EXAMPLES">
<xmp></xmp>
</ledi>
<ledi class="PARMS">
<p conloc="commattr">
</ledi>
<ledi class="USAGE">
<p>See <xref refid="langref">.</p>
</ledi>
<ledi class="CONTEXT"><pblk conloc="context_len">
</ledi>
</le>
```

Attributes

COMPLANG

Specifies the computer language described in this LERS section.

class

references the CLASSNAME attribute on a LersDef element.

RETKEY=None | First

Use the RetKey attribute to enable or disable automatic running headings for this tag.

NONE

indicates that nothing is to be used. NODUP is like FIRSTLAST, except that if the first and last glossary terms on the page are the same, only the last glossary term is displayed in the running heading or footing.

FIRST

indicates that the first non-blank item on the page is to be used.

If you code any explicit RetKey elements, they are honored and will appear. If you nest elements that can generate a running heading (for example, a MsgList inside Lers), only the outer active generated heading is used. That is, if you specified automated RetKey generation for LERS and MSGLIST, a Msgno inside Lers will not be used in the RetKey area. But if you had an explicit RetKey inside the Msg, then the RetKey is honored as an explicit override.

classname=title_text

For each LEDI class, you can define the generated title for the LEDI elements.

+	AUTH
+	Authorization level; for example, user or superuser.
+	COMMENTS
+	Comments about the language element.
+	CONTEXT
+	The context in which the language element is used.
+	DEFAULTS
+	The defaults for the language element.
+	ERRCOND
+	Error conditions.
+	EXAMPLES
+	Examples of using the language element.
+	FLAGS
+	Control flags.
+	FORMAT
+	The format or syntax of the language element.
+	INTREP
+	Internal representation, such as control blocks or data structures.
+	MESSAGES
+	Any messages related to the language element.
+	OTHER
+	An open category; define your own title.
+	PARMS
+	Parameters.
+	PROCESS
+	Processing related to the language element.
+	PURPOSE
+	The purpose of the language element.
+	RESTRICT
+	Restrictions on the use of the language element.
+	RESULTS
+	The results of using the language element.
+	RETCODES
+	Return codes.
+	SYSENV
+	The system environments to which the language element applies.
+	USAGE
+	How to use the language element.
+	VERSION
+	Any version information for the language element.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 15. Developing Programming Language Reference Materials” on page 141.

Contexts

Children: “LE (language element)” on page 302, “RetKey (retrieval key)” on page 407.

Parents: “Appendix” on page 212, “Body (document body)” on page 222, “DBody (division body)” on page 246, “LEDI (language element description item)” on page 304, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348.

LERSTDef (LERS property definition)

Purpose

Use LERSTDef to define the titles to be generated for different classes of LEDI elements, and to define values for common properties.

Examples

```
<lersdef classname="taglers" comments="Usage" context="Contexts"
defaults="Style Values" examples="Examples" format="Syntax"
other="SGML Markup" parms="Attributes" results="More Information"
usage="Usage">
</lersdef>
```

Attributes

CLASSNAME = *classname*

Defines the name of the class. This name is the name referenced by the CLASS attribute on the Lers element.

classname=*title_text*

For each LEDI class, you can define the generated title for the LEDI elements.

AUTH

Authorization level; for example, user or superuser.

COMMENTS

Comments about the language element.

CONTEXT

The context in which the language element is used.

DEFAULTS

The defaults for the language element.

ERRCOND

Error conditions.

EXAMPLES

Examples of using the language element.

FLAGS

Control flags.

FORMAT

The format or syntax of the language element.

INTREP

Internal representation, such as control blocks or data structures.

MESSAGES

Any messages related to the language element.

OTHER

An open category; define your own title.

PARMS

Parameters.

PROCESS

Processing related to the language element.

PURPOSE

The purpose of the language element.

RESTRICT

Restrictions on the use of the language element.

RESULTS

The results of using the language element.

RETCODES

Return codes.

SYSENV

The system environments to which the language element applies.

USAGE

How to use the language element.

VERSION

Any version information for the language element.

See “Common Element Attributes (large set)” on page 204.

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 15. Developing Programming Language Reference Materials” on page 141.

Contexts

Children: “Desc (element description)” on page 251.

Parents: “PropDefs (property definitions)” on page 396, “PropGroup (property group)” on page 397.

LI (list item)

Purpose

The LI element contains a single list item within a list. List items can be grouped together with LiBlk..

Examples

```
<ul>
<li>This is an item in an unordered list. To separate
it from other items in the list, the formatter puts
a bullet beside it.</li>
<li>The paragraph that is contained in the LI element
is part of the list item which contains it. <p>This
```

```
is the contained paragraph.</p></li>
<li>This is a separate list item in our unordered
list.</li>
</ul>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Unordered lists” on page 23.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Attention (safety notice)” on page 214, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Danger (danger notice)” on page 243, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “LIBlk (list item block)” on page 314, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “StepNotes (step notes)” on page 416, “UL (unordered list)” on page 436.

LibEntry (document library definition)

Purpose

Use LibEntry to define the bibliographic information about a library or other collection of documents. You can use the CLASS attribute and the ClassDef element to define different classes of LibEntry to correspond to different classes of collection. For example, you may have product libraries that are themselves collected into larger libraries of libraries. You could define a class of “CollectionKit” for libraries of libraries and then use LibEntry elements to define the contents of a given collection kit.

LibEntry elements can also be specified in a BibEntryDefs section or object container and used by reference from within a document, for example, from Cit

elements. When LibEntry elements are specified within Cit, the LibEntry elements are collected for use in a generated bibliography.

Contained BibEntry elements and LibEntry elements are normally used within re-used information in order to ensure that the re-used information is completely self-contained. In other words, these elements should be used within the scope of the information that is being re-used. For example, if a Division element is re-used, the BibEntry and LibEntry elements should be contained within that same division's DProlog element. This allows these elements to be completely contained, and thus completely re-usable, within the division that is being re-used.

It is intended that the public identifier of the library entity be used by the presentation system to locate the actual system object, but specific presentation systems may define application-specific data to be specified as the system identifier of the library entity if they do not support the use of public identifiers. The public identifier can be included in the LibEntry itself as a way of keeping a library's formal public identifier definition with the rest of its bibliographic information. This could allow, for example, the automatic generation of entity declarations for libraries described by LibEntry elements.

Examples

```
<bibentrydefs>
<ibmlibentry>
<library><titleblk><title>BS/300</title></titleblk>
</library>
<ibmbofnum>SB0F-1234-0</ibmbofnum>
<containeddocs bibids="booka bookb"></ibmlibentry>
<ibmbibentry id="booka"><doctitle><titleblk><title>
BS/300 Guide</title></titleblk></doctitle></ibmbibentry>
<ibmbibentry id="bookb"><doctitle><titleblk><title>
BS/300 Reference</title></titleblk></doctitle></ibmbibentry>
<libentry>
<library><titleblk><title>Back'n'Recovery</title>
</titleblk></library>
</libentry>
</ibmlibentry>
</bibentrydefs>
```

Usage

See "Defining library entries" on page 121.

Contexts

Children: "BOFNum (bill of forms number)" on page 222, "ContainedDocs (documents in IBMLibEntry and LibEntry)" on page 236, "Desc (element description)" on page 251, "ISBN (document ISBN number)" on page 295, "Library" on page 314, "OrderNum (order number)" on page 372, "ProdName (product name)" on page 394, "PrtLoc (country where printed)" on page 398, "PublicID (public identifier)" on page 399, "Publisher (document publisher)" on page 400.

Parents: "BibEntryDefs (contains bibliographic entries)" on page 219, "BibList (bibliography entry list)" on page 220, "Cit (document citation)" on page 229.

LIBlk (list item block)

Purpose

The LIBlk element groups items within a list. The reasons for the grouping are determined by the author. The grouping may be logical, and can be indicated by including an optional title. LiBlk can also be used define blocks of list items to be connected with a Bridge element.

If you want to have all the text in the LIBLK to be on the same page, use `style="xpp:(keep)"`. Be cautious when using this feature. If the text does not fit on a page, remove the `style="xpp:(keep)"` or the pages will not format correctly.

Migration Note

The use of LIBlk, its title, or the use of Bridge, replaces the List Part (LP) element from BookMaster.

Examples

```
<ol>
<liblk>
<li>1 GIG SCSI-2 Hard Disk</li>
<li>32 MB RAM</li>
<li>128-Bit 8MB VRAM Video</li>
<li>21-Inch Monitor</li>
</liblk>
<liblk>
<li>Great Word Processor</li>
<li>Best Multimedia App</li>
<li>Voice Mail</li>
</liblk>
</ol>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Grouping list items” on page 32.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “LI (list item)” on page 311, “Title” on page 431.

Parents: “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “StepNotes (step notes)” on page 416, “UL (unordered list)” on page 436.

Library

Purpose

The Library element contains the name of a library.

Examples

```
See these books for a good read and then a weird read: <cit>
<bibentry><doctitle><titleblk><title>Tom Sawyer</title>
</titleblk></doctitle></bibentry></cit> and <cit>
<ibmbibentry><doctitle>
<library><titleblk><title>System/36</title></titleblk>
</library>
<titleblk><title>Concepts and Programmer's Guide</title>
</titleblk></doctitle></ibmbibentry></cit>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

For more information about the Library element, see “An example of using BibEntry and BibEntryDefs” on page 122.

Contexts

Children: “TitleBlk (title information)” on page 432.

Parents: “DocTitle (document title)” on page 256, “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

Lines (text with line boundaries)

Purpose

The Lines element contains text for which the input line (record) boundaries are significant and must be preserved or indicated when presented.

Examples

```
<LINES>
a partridge in a pear tree
two turtledoves
three French hens
four calling birds
five golden rings
six geese a-laying
seven swans a-swimming
eight maids a-milking
nine ladies dancing
ten lords a-leaping
eleven pipers piping
twelve drummers drumming
</LINES>
```

Attributes

NOTATION=LINESPEC

Specifies that LINESPEC is the default value for the NOTATION attribute.

OBJ=*entity_name*

The name of the external data entity that contains the line-specific data. When OBJ is specified, it is an error to specify any data or the Lines end tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Just plain lines” on page 43.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “Formula (math formula)” on page 267, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “MD (marked deletion)” on page 326, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term” on page 425, “TM (Trademark)” on page 433, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BackCover (back cover)” on page 217, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “FrontCover” on page 270, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

Litdata (literal data)

Purpose

The Litdata element contains or refers to literal data in a specific notation. Use Litdata to contain or refer to data that is not to be parsed by the SGML parser and that may need special processing in order to properly present the data, such as character translation or use of special code pages.

When the OBJ attribute is used to refer to an external entity, the Litdata end tag cannot be specified. Do not specify a notation when referring to an external entity because the entity will have a notation defined on its entity declaration.

The typical use of Litdata is to contain or refer to samples of programming code.

The only SGML markup recognized within Litdata is the end tag open delimiter (</). When literal data is included inline in the document, the end tag open delimiter ends the Litdata element, regardless of the context.

Examples

```
<!ENTITY testprg SYSTEM "testprg.c" ndata c>
:
```

```

<FIG>
<CAP>A basic C++ program.
<LITDATA OBJ="TESTPRG">
</FIG>

```

Attributes

NOTATION=*notation_name*

The name of the notation that the data is in. The notations supported are defined by the specific implementation of IBMIDDoc you are using, but typical notations include:

LINESPEC

Specifies that the line ends are respected. This is the default.

C

Cpp

C and C++ program source.

The processing application uses the notation name to determine what special processing is needed to present the literal data. For example, in the case of C program code, the square bracket and curly bracket characters may need special treatment.

OBJ=*entity_name*

Specifies the name of an external data entity that contains the literal data. When OBJ is specified, the Litdata end tag cannot be specified.

CDATA

Is the literal data.

Usage

See "Literal text data" on page 45.

Contexts

Children: text (CDATA).

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "BackCover (back cover)" on page 217, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "CGraphic (character graphic)" on page 226, "Danger (danger notice)" on page 243, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "Fn (footnote)" on page 265, "FrontCover" on page 270, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "PBlk (paragraph block)" on page 380, "ProcIntro (procedure introduction)" on page 391, "Screen (display screen)" on page 411, "SynNote (syntax note)" on page 418, "Warning (warning notice)" on page 439, "Xmp (example)" on page 440.

LQ (stand-alone quotation)

Purpose

Use the LQ element to contain material excerpted from another source that can stand alone without a defining context. This is usually the case when the quotation is of considerable length. The LQ may be of considerable length and can contain almost any element, including divisions.

Examples

```
<lq>The only thing we have to fear is fear itself.  
</lq>
```

Attributes

BibId=*bibentry_id*

The ID of the BibEntry element that defines the source of the quotation. You must either specify BIBID or include a BibEntry element within the LQ element.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Quotes and excerpts” on page 39.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Attention (safety notice)” on page 214, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Danger (danger notice)” on page 243, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table

| entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on
| page 264, "Fn (footnote)" on page 265, "LEDI (language element description item)"
| on page 304, "LI (list item)" on page 311, "ModDesc (modular content
| description)" on page 343, "ModItem (module description item)" on page 344,
| "MsgItem (message description item)" on page 348, "NoteBody (note body)" on
| page 362, "P (paragraph)" on page 374, "PBlk (paragraph block)" on page 380,
| "ProcIntro (procedure introduction)" on page 391, "SynNote (syntax note)" on
| page 418, "Warning (warning notice)" on page 439.

Maintainer (reader comment)

Purpose

The content of the Maintainer element is used for tracking control information. It is also used in generating a Reader Comment Form. If this information is not included, the Reader Comment Form cannot be generated by the output processor.

Usage

See “Using reader’s comment form (RCF)” on page 90.

Examples

```
<maintainer>
<corp>
<corpname>IBM Corporation</corpname>
<address>ATTN: Dept 542
3605 HWY 52 N
Rochester, MN
<postalcode>55901-9986</postalcode></address>
</corp>
</maintainer>
```

Contexts

Children: “Corp (enterprise name and address)” on page 238, “Person (person’s name and address)” on page 381.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

Mark (marked note definition)

Purpose

Use Mark to define a collection of marked notes. To use marked notes, you must define at least one collection in a document. All marked notes must be associated with a collection.

You can define different collections of notes to correspond to different releases of a document or to different types of notes. Different collections may use the same classes and actions. For example, if you use marked notes to track changes and generate a summary of changes, you can define different collections for different releases or drafts of a document. Or if you are tracking changes from different sources, you can define a collection for each different source.

Examples

```
<mark id="mkv5r4" ident="use">
<desc>V5R4 marked changes</desc>
</mark>
```

Attributes

ID=*collection_id*
Defines the mark collection ID.

IDENT=USE | IGNORE

Determines whether to process the specified collection of marked notes as follows:

USE

Process the collection of marked notes.

IGNORE

Ignore the collection of marked notes.

Conloc

The CONLOC attribute specifies that the content of another element of the same type is to be used as the content of the referencing element. This enables reuse of information. When the CONLOC attribute is specified, you cannot specify the element's end tag. The result of using CONLOC is exactly the same as if the element being referred to had occurred at that point in the document. See "Reusing elements from an object library" on page 166.

Attributes on the element are now passed to the element with the CONLOC; this is a feature that began with IDWB release 3.4, patch IDWXF036.

Class

The Class attribute associates an element class with an element. This attribute must contain an SGML name that has been defined as a class name in a ClassDef element. This attribute only applies to elements specified on the ClassDef's ELETYPE attribute. Element classes are defined with ClassDef elements within PropDefs. See "Chapter 19. Property and Class Definition" on page 177 for more information.

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See "Using Revisions" on page 91.

Status

ignored by processes

InfoMast

A fixed attribute used to classify the element.

Usage

See "Creating Collections of Marked Notes" on page 93.

Contexts

Children: "Desc (element description)" on page 251.

Parents: "RevDefs (revision tracking information)" on page 409.

MarkList (marked note list)

Purpose

Use MarkList to include a list of marked notes. Only notes that are members of the specified collections and that have the specified class and action values are included in the list.

Examples

```
<marklist mkids="mkv5r4" classes="msg abend" actions="change del rep"
display="item action location desc" classhd="Msg"
actionhd="Reason" lochd="Loc" descd="Desc">
```

Attributes

ID=*marklist_id*

Specifies the ID of this element.

DISPLAY=**NAMES**

CLASSES=*class_name*

Defines the mark class or classes to include in this list. Only notes with a specified class will be included.

ACTIONS=*action_name*

Defines the action or actions to include in this list. Only notes with a specified action will be included.

SPEC=**AUTO** | **MAN**

Specifies that the content of the element is generated.

Future Enhancement

At this time, MAN is not supported.

MKIDS=*mark_id*

Defines which collections to search for notes to present.

DISPLAY=**CLASS** | **ACTION** | **LOC** | **ITEM** | **DESC**

Specifies the type of information to display in the generated list. You can choose one or more items from the group, but you can choose each item only once. The items in the list are displayed in the order they are specified on the Display attribute.

CLASS

Specifies that the CLASS attribute values for the marked notes are to be included.

ACTION

Specifies that the ACTION attribute values for the marked notes are to be displayed.

ITEM

Specifies that the ITEM attribute values for the marked notes are to be displayed.

LOC

Specifies that the locations of the marked notes, either page numbers or online equivalents, are to be displayed.

DESC

Specifies that the contents of the marked notes are to be displayed.

CLASSHd=*column_heading*

ACTIONHd=*column_heading*

ITEMHd=*column_heading*

LOCHd=*column_heading*

DESCHd=*column_heading*

Defines the heading text associated with each type of information in the generated list. These values override the default headings defined in the document style.

TOC=**toc** | **notoc**

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating Collections of Marked Notes” on page 93.

Contexts

Children: empty.

Parents: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “LEDI (language element description item)” on page 304, “MsgItem (message description item)” on page 348, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391 .

MasterIndex (master index)

Purpose

The Master Index is a way to combine the indexes of several documents. Rather than having to look in the index of several documents, the user can look in the master index for the correct document and page number where the index entry is located.

Examples

```
<backm>
<masterindex>
<specdprolog><gendtitle></specdprolog>
<masterindexobj obj="gsugidx">
<masterindexobj obj="planidx">
<masterindexobj obj="instidx">
</masterindex></backm>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “MasterIndexObj (master index object)” on page 324, “SpecDProlog (special section division prolog)” on page 415.

Parents: “BackM (back matter)” on page 217.

MasterIndexInfo (master index information)

Purpose

MasterIndexInfo is used in conjunction with MasterIndexPrefix. Use these tags when preparing to merge the indexes of several documents into a master index. These tags should be used in the content of each document containing an index you wish to have merged into a master index.

Examples

```
<ibmbibentry><doctitle>
<library><titleblk>
<title>ID Workbench</title>
</titleblk></library>
<titleblk>
<title>Getting Started and User's Guide</title>
</titleblk></doctitle>
<externalfilename>idfgsmst</externalfilename>
</ibmbibentry>
<masterindexinfo>
<masterindexprefix>GSUG</masterindexprefix>
</masterindexinfo>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “MasterIndexPrefix (master index prefix)” on page 325.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

MasterIndexObj (master index object)

Purpose

MasterIndexObj is used with the MasterIndex tag. The MasterIndexObj tag provides a reference to the index of a document to be included in the Master Index. This is a separate document in itself. It is separate from the documents with indexes you are merging.

Examples

```
<!ENTITY guide SYSTEM "rweguide.mdx" ndata mindex>
<!ENTITY ref SYSTEM "rweref.mdx" ndata mindex>
...
<masterindex>
<specdprolog><gendtitle></specdprolog>
<masterindexobj obj="guide">
<masterindexobj obj="ref">
</masterindex>
```

Attributes

OBJ

Specifies the name of the master index entity.

Class

The Class attribute associates an element class with an element. This attribute must contain an SGML name that has been defined as a class name in a ClassDef element. This attribute only applies to elements specified on the ClassDef's ELETYPE attribute. Element classes are defined with ClassDef elements within PropDefs. See "Chapter 19. Property and Class Definition" on page 177 for more information.

ID The ID attribute identifies an element within an SGML document. IDs must be unique within a single document. Any element that has an ID can be cross-referenced or linked to. IDs can be up to 64 characters long. IDs must start with an alphabetic character and can contain letters, numbers, dashes (-), or periods (.).

Props

The Props attribute is used to specify the condition under which the information contained within the element appears. See "Property-Based Retrieval" on page 171.

PropSrc

Points to an element whose properties are to be used as the properties of the referencing element. See "Chapter 19. Property and Class Definition" on page 177.

Status

ignored by processes

Style

The Style attribute contains either a reference to a separate style specification for an element or a set of element-specific presentation style definitions when the processing system does not support separate styles for elements. Assigning a value to the STYLE attribute modifies how an element is presented when it is output.

Usage

See "Creating a master index" on page 105.

Contexts

Children: empty.

Parents: "MasterIndex (master index)" on page 323.

MasterIndexPrefix (master index prefix)

Purpose

MasterIndexPrefix is used in conjunction with MasterIndexInfo. Use these tags when preparing to merge the indexes of several documents into a master index. These tags should be used in the content of each document containing an index you wish to have merged into a master index.

Examples

```
<masterindexinfo>
<masterindexprefix>GSUG</masterindexprefix>
</masterindexinfo>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating a master index” on page 105.

Contexts

Children: text (#pcdata).

Parents: “MasterIndexInfo (master index information)” on page 324.

MD (marked deletion)

Purpose

The MD element identifies data that no longer applies. The REV attribute associates the MD element with a specific revision.

MD can contain any other phrase-like elements. The MD element cannot be used to mark data that contains paragraphs or division elements. In these cases, the revision and status attributes provided for those elements must be used to indicate the revision level and deletion status of the data.

Examples

The following data no longer applies:
<MD>This data no longer applies.</MD>,
as you can clearly see.

Attributes

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See “Using Revisions” on page 91.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Marking text for deletion” on page 93.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “Formula (math formula)” on page 267, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term” on page 425, “TM (Trademark)” on page 433, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

MkAction (marked note action definition)

Purpose

Use MkAction to define actions that can be associated with marked notes. You can define as many actions as you need. Any mark action can be used with any mark class. These actions are used within a MKNote element to describe the action that is associated with that note. The MarkList element uses the MKAction to determine which marked notes are to be presented.

See “Creating Collections of Marked Notes” on page 93.

Examples

```
<propdefs>
<mkdesc>
<mkclass name="msg">Msg</mkclass>
<mkclass name="abend">Abend</mkclass>
<mkaction name="new">New</mkaction>
<mkaction name="changed">Changed</mkaction>
<mkaction name="deleted">Deleted</mkaction>
<mkaction name="replaced">Replaced</mkaction>
</mkdesc>
</propdefs>
```

Attributes

NAME=*action_name*

Specifies the short name of the action, which is the value specified in the Action attribute of MkNote and the Actions attribute of MarkList.

Conloc

The CONLOC attribute specifies that the content of another element of the same type is to be used as the content of the referencing element. This enables reuse of information. When the CONLOC attribute is specified, you cannot specify the element’s end tag. The result of using CONLOC is exactly the same as if the element being referred to had occurred at that point in the document. See “Reusing elements from an object library” on page 166.

Attributes on the element are now passed to the element with the CONLOC; this is a feature that began with IDWB release 3.4, patch IDWXF036.

ID The ID attribute identifies an element within an SGML document. IDs must be unique within a single document. Any element that has an ID can be

cross=referenced or linked to. IDs can be up to 64 characters long. IDs must start with an alphabetic character and can contain letters, numbers, dashes (-), or periods (.).

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See “Using Revisions” on page 91.

Status

ignored by processes

Style

The Style attribute contains either a reference to a separate style specification for an element or a set of element-specific presentation style definitions when the processing system does not support separate styles for elements. Assigning a value to the STYLE attribute modifies how an element is presented when it is output.

HyTime

ignored by processes

InfoMast

A fixed attribute used to classify the element.

Reftype

ignored by processes

Usage

See “Creating Collections of Marked Notes” on page 93.

Contexts

Children: text (#pcdata).

Parents: “MkDesc (mark description)” on page 329.

MkClass (marked note class definition)

Purpose

The MkClass element defines a marked note class. These class codes are used in a MarkNote element to specify the class of the note.

Examples

```
<propdefs>
<mkdesc>
<mkclass name="msg">Msg</mkclass>
<mkclass name="abend">Abend</mkclass>
<mkaction name="new">New</mkaction>
<mkaction name="changed">Changed</mkaction>
<mkaction name="deleted">Deleted</mkaction>
<mkaction name="replaced">Replaced</mkaction>
</mkdesc>
</propdefs>
```

Attributes

NAME=*class_name*

Specifies the short name of the class, which is the value specified in the Class attribute of MkNote and the Classes attribute of MarkList.

Conloc

The CONLOC attribute specifies that the content of another element of the same type is to be used as the content of the referencing element. This enables reuse of information. When the CONLOC attribute is specified, you cannot specify the element's end tag. The result of using CONLOC is exactly the same as if the element being referred to had occurred at that point in the document. See "Reusing elements from an object library" on page 166.

Attributes on the element are now passed to the element with the CONLOC; this is a feature that began with IDWB release 3.4, patch IDWXF036.

ID The ID attribute identifies an element within an SGML document. IDs must be unique within a single document. Any element that has an ID can be cross-referenced or linked to. IDs can be up to 64 characters long. IDs must start with an alphabetic character and can contain letters, numbers, dashes (-), or periods (.).

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See "Using Revisions" on page 91.

Status

ignored by processes

Style

The Style attribute contains either a reference to a separate style specification for an element or a set of element-specific presentation style definitions when the processing system does not support separate styles for elements. Assigning a value to the STYLE attribute modifies how an element is presented when it is output.

HyTime

ignored by processes

InfoMast

A fixed attribute used to classify the element.

Reftype

ignored by processes

Usage

See "Creating Collections of Marked Notes" on page 93.

Contexts

Children: text (#pcdata).

Parents: "MkDesc (mark description)".

MkDesc (mark description)

Purpose

The MkDesc element describes the classes and actions that can be used with marked notes.

Use MkDesc to define the classes and actions that are valid for marked notes. Any mark action can be used with any mark class.

Examples

```
<propdefs>
<mkdesc>
<mkclass name="msg">Msg</mkclass>
<mkclass name="abend">Abend</mkclass>
<mkaction name="new">New</mkaction>
<mkaction name="changed">Changed</mkaction>
<mkaction name="deleted">Deleted</mkaction>
<mkaction name="replaced">Replaced</mkaction>
</mkdesc>
</propdefs>
```

Attributes

Conloc

The CONLOC attribute specifies that the content of another element of the same type is to be used as the content of the referencing element. This enables reuse of information. When the CONLOC attribute is specified, you cannot specify the element's end tag. The result of using CONLOC is exactly the same as if the element being referred to had occurred at that point in the document. See "Reusing elements from an object library" on page 166.

+ Attributes on the element are now passed to the element with the CONLOC;
+ this is a feature that began with IDWB release 3.4, patch IDWXF036.

| **ID** The ID attribute identifies an element within an SGML document. IDs must be
| unique within a single document. Any element that has an ID can be
| cross=referenced or linked to. IDs can be up to 64 characters long. IDs must
| start with an alphabetic character and can contain letters, numbers, dashes (-),
| or periods (.).

Rev

The REV attribute references the Rev element which describes the last revision level of the information. See "Using Revisions" on page 91.

Status

ignored by processes

Style

The Style attribute contains either a reference to a separate style specification for an element or a set of element-specific presentation style definitions when the processing system does not support separate styles for elements. Assigning a value to the STYLE attribute modifies how an element is presented when it is output.

HyTime

ignored by processes

InfoMast

A fixed attribute used to classify the element.

Reftype

ignored by processes

Usage

See "Creating Collections of Marked Notes" on page 93.

Contexts

Children: "MkAction (marked note action definition)" on page 327, "MkClass (marked note class definition)" on page 328.

Parents: “PropDefs (property definitions)” on page 396, “PropGroup (property group)” on page 397.

MkNote (marked note)

Purpose

The MkNote element contains a marked note, which is a specialized annotation element that can be associated with problem and change tracking information such as change requests or problem numbers. Use marked notes to collect information about the content of your document such as notes about changes, notes to yourself as an author, or references to specific locations as navigation aides to readers. Marked notes are presented in marked note lists as defined by the MarkList element.

For example, you can automatically create a summary of changes by using marked notes to record changes within your document and using a MarkList element to generate a list of those notes.

Examples

```
<ibmddoc docstyle="ibmxagd">
<prolog><ibmbibentry><doctitle><titleblk>
<title>My Marked Changes Document for Messages</title>
</titleblk></doctitle></ibmbibentry>
<propdefs>
<mkdesc>
<!--Define two classes for marked lists - notes and abends-->
<mkclass name="msg">Msg</mkclass>
<mkclass name="abend">Abend</mkclass>
<!--Define the actions for the changed info-->
<mkaction name="new">New</mkaction>
<mkaction name="change">Changed</mkaction>
<mkaction name="del">Deleted</mkaction>
<mkaction name="rep">Replaced</mkaction>
</mkdesc>
</propdefs>
<revdefs>
<rev id="revv4r5" ident="use">
<date></date>
<desc></desc>
</rev>
<mark id="mkv4r5" ident="use">
<desc>v4r5 marked message changes</desc>
</mark>
</revdefs>
</prolog>
<body>
<d>
<dprolog><titleblk>
<title>List of changed items</title>
</titleblk></dprolog>
<dbody>
<marklist mkids="mkv4r5" classes="msg abend" actions="change del rep"
display="item action page desc" classhd="Msg" actionhd="Reason"
itemhd="Msg" lochd="Page" descd="Message text"></dbody>
</d>
<msglist>
<msg rev="revv4r5">
<msgnum>IDW0012</msgnum>
<msgtext>Hi there!</msgtext>
<msgitem class="xpl">
<p>This is a friendly message.<mknote class="msg"
```

```

action="change" mkids="mkv4r5" item="IDW0012">Hi there!
</mknote></p>
</msgitem>
</msg>
<msg rev="revv4r5">
<msgnum>IDW0013</msgnum>
<msgtext>Farewell!</msgtext>
<msgitem class="xpl">
<p>This unlucky message was removed.<mknote class="msg"
action="del" mkids="mkv4r5" item="IDW0013">Farewell!
</mknote></p>
</msgitem>
</msg>
</msglist></body>
</ibmidoc>

```

Attributes

CLASS=*class_name*

Defines one or more mark classes to which this marked note belongs.

ACTION=*action_name*

Defines one or more actions associated with this marked note.

MKIDS=*mark_id*

Contains the IDs of one or more Mark elements, which define collections of marked notes.

ITEM=*item_value*

Defines an identifying label for the note, such as a message number, function name, or error report. The Item attribute enables you to distinguish among marks of the same class and action. You can also use the Item attribute to closely associate a note with things in your document that have unique identifiers such as message numbers.

DISPLAY=*items-to-display*

Specifies the items to display and the order to display them in. Values are: item, action, loc or page, and desc.

CLASSHD=*class-heading*

Specifies the heading for the Class column.

ACTIONHD=*action-heading*

Specifies the heading for the Action column.

ITEMHD=*item-heading*

Specifies the heading for the Item column.

LOCHD=*location-heading*

Specifies the heading for the Location column.

DESHD=*description-heading*

Specifies the heading for the Description column.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating Collections of Marked Notes” on page 93.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge

(bridge between concepts)" on page 223, "CGraphic (character graphic)" on page 226, "Char (character data)" on page 227, "Cit (document citation)" on page 229, "Date" on page 244, "Dec (decimal number)" on page 247, "DL (definition list)" on page 253, "Fig (figure)" on page 262, "Formula (math formula)" on page 267, "GL (glossary list)" on page 272, "Hex (hexadecimal)" on page 277, "L (explicit link)" on page 300, "Lines (text with line boundaries)" on page 315, "Litdata (literal data)" on page 316, "MD (marked deletion)" on page 326, "MMObj (multi-media object; artwork)" on page 355, "Note" on page 362, "NoteList (ordered note list)" on page 363, "Num (number)" on page 366, "Oct (octal number)" on page 369, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "Ph (Phrase)" on page 382, "PK (programming keyword)" on page 385, "PV (parameter variable)" on page 400, "Q (quotation phrase)" on page 402, "RefKey (reference key)" on page 405, "Screen (display screen)" on page 411, "SynPh (syntax phrase)" on page 419, "Syntax (syntax diagram)" on page 420, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "Danger (danger notice)" on page 243, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "PBlk (paragraph block)" on page 380, "ProcIntro (procedure introduction)" on page 391, "Warning (warning notice)" on page 439.

MMObj (multi-media object; artwork)

Purpose

The MMObj element refers to a non-text object such as an image, vector graphic, or video clip. The types of objects supported depends on your processing and presentation systems. For print, non-text objects are usually image or vector graphics such as EPS graphics. For online information, they would be BMPs, GIFs, or JPEGs. MMObj also contains a text description of the object, intended to provide an alternative to the object itself.

Depending on the style definition, the object can be integrated with the text. It is normally presented inline with the text. The online presentation of non-text objects depends on the presentation system being used. When you need to refer to a single non-text object, you can use the OBJ attribute of the MMObj element to specify the entity that represents the object.

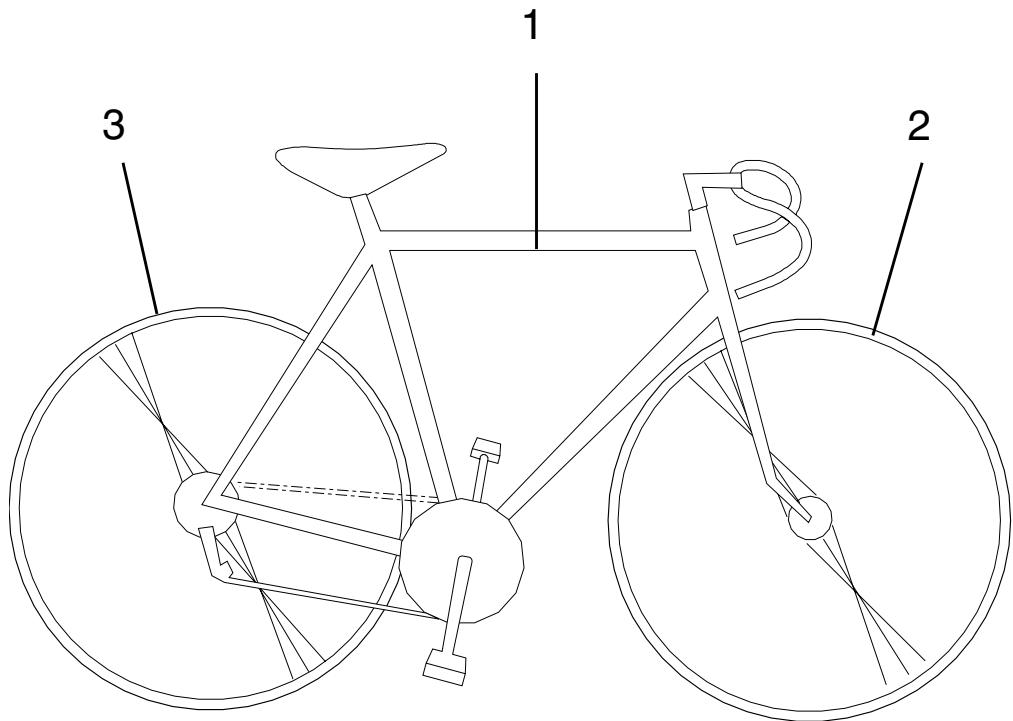
Migration Note

MMObj replaces the ARTWORK and ARTALT tags. Use MMObj as you would ARTALT. The presentation attributes on the ARTWORK tag are now notation attributes specified on the entity declaration used to declare the artwork file.

Examples

Here's the declaration and the markup for an illustration of a bike:

```
<!ENTITY bike system "bike.gif" ndata graphics>
...
<MMOBJ>
<OBJREF OBJ="bike">
<TEXTALT>This is a two-wheeled bicycle.</TEXTALT>
</MMOBJ>
```



Attributes

LABEL=*position*

Defines a text label to use for a graphic object when the object cannot be presented, or to use as a link button.

Placement=*position*

Controls where the graphic appears on the page. Valid values include: standalone, inline, and margin.

Standalone

Places artwork on a separate line. This is the default value.

Inline

Places artwork in the current text stream without a preceding or following line break. This replaces the previously used style override `bkm:(runin)`

Margin

Places artwork in the offset margin.

Halign=*alignment*

Controls the horizontal alignment of the graphic within the textline when standalone is specified. Valid values include: left, center, right, and current.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Including artwork in documents” on page 45.

Contexts

Children: “MMObjLink (multi-media object link)”, “ObjRef (object reference)” on page 368, “TextAlt (text alternative)” on page 427.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BackCover (back cover)” on page 217, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “CoverDef (cover definition)” on page 239, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “EdNotices (edition notices)” on page 259, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “FrontCover” on page 270, “LeDesc (language element description)” on page 304, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NItem (notice item)” on page 359, “NoteBody (note body)” on page 362, “Notices (contains notices)” on page 364, “P (paragraph)” on page 374, “PartAsm (part assembly)” on page 379, “PartAsmSeg (part assembly segment)” on page 380, “PBlk (paragraph block)” on page 380, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “Safety (safety notices)” on page 411, “Screen (display screen)” on page 411, “Sem (semantic meaning)” on page 412, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439.

MMObjLink (multi-media object link)

Purpose

The MMObjLink element allows you to create a graphic hot spot under an image.

Examples

```
<mobj><objref obj="part1">  
<mobjlink linkend="newdiv">  
<areadef coords="1 100">  
</areadef></mobjlink>  
<textalt></textalt>  
</mobj>
```

Attributes

LINKEND=*text_target*

Specifies the ID of the textual target of the link.

AREADEFS=*ref-ID*

Specifies the ID of the AreaDef element containing the graphic area specification of the graphic hot spot. If AREADEFS is specified, MMObjLink must be an empty element. To specify the area definition, see “AreaDef (defines graphic hot spot area)” on page 213.

If you do not specify an area definition using an ID, use an inline AreaDefs element to define the hotspot.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating graphic links” on page 46.

Contexts

Children: “AreaDef (defines graphic hot spot area)” on page 213.

Parents: “MMObj (multi-media object; artwork)” on page 333.

Mod (information module)

Purpose

The Mod element contains a module of information within a collection of modules. The modules usually have the same structure.

Use modular information to create reference information. You can define as many different modular information classes as you want for information that is similarly structured.

Examples

```
<PROPDEF>  
<MODINFODEF CLASSNAME="CUST">  
<DESC>CUSTOMER INFORMATION</DESC>  
<MODITEMDEF CLASSNAME="NAME">  
<TITLE>NAME</TITLE>  
<DESC>THIS IS THE FIRST AND LAST NAME OF THE CUSTOMER</DESC>  
</MODITEMDEF>  
<MODITEMDEF CLASSNAME="CUSTID">  
<TITLE>ID</TITLE>  
<DESC>THIS IS THEIR CUSTOMER ID NUMBER</DESC>  
</MODITEMDEF>  
<MODITEMDEF CLASSNAME="INC">  
<TITLE>INCOME</TITLE>
```

```

    <DESC>THIS IS THEIR ANNUAL ESTIMATED INCOME</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="LPD">
    <TITLE>LAST PURCHASE DATE</TITLE>
    <DESC>THIS IS THEIR LAST PURCHASE DATE</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="NOTES">
    <TITLE>NOTES</TITLE>
    <DESC>PERSONAL NOTES</DESC>
  </MODITEMDEF>
</MODINFODEF>
</PROPDEFS>
:
<MODINFO CLASS="CUST" STYLE="TABLE">
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">FRED SMITH</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1000</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><NUM BASE="10">40000</NUM></MODITEM>
    <MODITEM CLASS="LPDATE"><P><DATE>12/25/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>BIG SPENDER</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">SUZANNE STANLEY</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1001</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">50000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>11/22/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>LIKES GAME SOFTWARE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">JEFF GEORGE</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1002</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">60000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/02/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">MIKE GIDENTO</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1003</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">35000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/12/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
</MODINFO>

```

Attributes

CLASS=*classname*

Defines the class of this information module. Classes are defined with a ModInfoDef element within the document or division prolog.

Usage

See “Chapter 16. Defining Modular Information” on page 151.

Contexts

Children: “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “ModName (modular information element name)” on page 346, “RetKey (retrieval key)” on page 407.

Parents: “ModInfo (modular information)” on page 338.

ModInfo (modular information)

Purpose

The ModInfo element contains a set of information modules for a single class of modular information.

Use a ModInfo element to contain a set of information modules that describe a type of information that is specific to your document. For example, in a document describing a complex program, you can define a class of information module for describing data structures.

Note: If your information describes the elements of a computer language, use the LERS element for your information.

Information modules can also be organized within object libraries and used by reference from within a modular information section using the CONLOC attribute.

The default presentation style for modular information is as divisions, but other presentation styles are defined. For example, you can present each module as a row in a table for quick reference.

MODINFO with any style except STYLE=TABLE becomes a series of nested divisions. When STYLE=TABLE is used, it is mapped to a table. The title of MODINFO becomes CAP; DESC becomes DESC; MODINFOTITLE becomes ENTRY; MOD becomes ROW; MODNAME becomes ENTRY, and the TITLE in MODITEM is suppressed.

To use information modules, you must define an information module class using the ModInfoDef element. To better facilitate reuse, you should consider containing the ModInfoDef element within your modular information unit. You can specify the modular information class on the ModInfo element, which applies to all contained Mod elements, or you can specify a class on each Mod element.

For example, suppose you define several related classes of information module that you want to group into a single ModInfo group for presentation. In that case, you need to specify the class for each module rather than a global class for the ModInfo element.

Examples

```
<PROPDEFS>
<MODINFODEF CLASSNAME="CUST">
<DESC>CUSTOMER INFORMATION</DESC>
  <MODITEMDEF CLASSNAME="NAME">
    <TITLE>NAME</TITLE>
    <DESC>THIS IS THE FIRST AND LAST NAME OF THE CUSTOMER</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="CUSTID">
    <TITLE>ID</TITLE>
    <DESC>THIS IS THEIR CUSTOMER ID NUMBER</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="INC">
    <TITLE>INCOME</TITLE>
    <DESC>THIS IS THEIR ANNUAL ESTIMATED INCOME</DESC>
  </MODITEMDEF>
  <MODITEMDEF CLASSNAME="LPD">
    <TITLE>LAST PURCHASE DATE</TITLE>
    <DESC>THIS IS THEIR LAST PURCHASE DATE</DESC>
```

```

</MODITEMDEF>
<MODITEMDEF CLASSNAME="NOTES">
  <TITLE>NOTES</TITLE>
  <DESC>PERSONAL NOTES</DESC>
</MODITEMDEF>
</MODINFODEF>
</PROPDEFS>
:
<MODINFO CLASS="CUST" STYLE="TABLE">
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">FRED SMITH</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1000</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">40000</NUM></P></MODITEM>
    <MODITEM CLASS="LPDATE"><P><DATE>12/25/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>BIG SPENDER</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">SUZANNE STANLEY</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1001</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">50000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>11/22/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>LIKES GAME SOFTWARE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">JEFF GEORGE</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1002</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">60000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/02/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">MIKE GIDENTO</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1003</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">35000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/12/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
</MODINFO>

```

Attributes

Title

Contains the Title for the ModInfo element.

RetKey

Contains the text to be used as a retrieval aid.

Desc

Contains a description of the information contained in the ModInfo element.

ModInfoDef

Contains the definition of classes of modular information.

Mod

Contains an information module.

Usage

See “Chapter 16. Defining Modular Information” on page 151.

Contexts

Children: “Desc (element description)” on page 251, “Mod (information module)” on page 336, “ModInfoDef (modular information property definition)” on page 340, “RetKey (retrieval key)” on page 407, “Title” on page 431.

Parents: “AnnotBody (annotation body)” on page 210, “Appendix” on page 212, “Attention (safety notice)” on page 214, “Body (document body)” on page 222, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Fn (footnote)” on page 265, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

ModInfoDef (modular information property definition)

Purpose

The ModInfoDef element defines a set of modular information properties.

Use ModInfoDef to define a class of modular information specific to your information. You can apply specific presentation styles to a given class of modular information using the STYLE attribute.

All classes referenced by a ModInfo element must be defined. These classes are usually defined in a central location that is accessed by many documents. This centralization allows control over the class definitions for modular information used at a publishing site.

The scope of a definition is determined by the location of the definition.

Examples

```
<PROPDEFS>
<MODINFODEF CLASSNAME="CUST">
<DESC>CUSTOMER INFORMATION</DESC>
<MODITEMDEF CLASSNAME="NAME">
  <TITLE>NAME</TITLE>
  <DESC>THIS IS THE FIRST AND LAST NAME OF THE CUSTOMER</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="CUSTID">
  <TITLE>ID</TITLE>
  <DESC>THIS IS THEIR CUSTOMER ID NUMBER</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="INC">
  <TITLE>INCOME</TITLE>
  <DESC>THIS IS THEIR ANNUAL ESTIMATED INCOME</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="LPD">
  <TITLE>LAST PURCHASE DATE</TITLE>
  <DESC>THIS IS THEIR LAST PURCHASE DATE</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="NOTES">
  <TITLE>NOTES</TITLE>
  <DESC>PERSONAL NOTES</DESC>
</MODITEMDEF>
</MODINFODEF>
</PROPDEFS>
:
<MODINFO CLASS="CUST" STYLE="TABLE">
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">FRED SMITH</MODNAME>
```

```

<MODITEM CLASS="ID"><P><NUM BASE="10">1000</NUM></P></MODITEM>
<MODITEM CLASS="INC"><NUM BASE="10">40000</NUM></MODITEM>
<MODITEM CLASS="LPDATE"><P><DATE>12/25/93</DATE></P></MODITEM>
<MODITEM CLASS="NOTES"><P>BIG SPENDER</P></MODITEM>
</MOD>
<MOD CLASS="CUST">
  <MODNAME CLASS="NAME">SUZANNE STANLEY</MODNAME>
  <MODITEM CLASS="ID"><P><NUM BASE="10">1001</NUM></P></MODITEM>
  <MODITEM CLASS="INC"><P><NUM BASE="10">50000</NUM></P></MODITEM>
  <MODITEM CLASS="LPD"><P><DATE>11/22/92</DATE></P></MODITEM>
  <MODITEM CLASS="NOTES"><P>LIKES GAME SOFTWARE</P></MODITEM>
</MOD>
<MOD CLASS="CUST">
  <MODNAME CLASS="NAME">JEFF GEORGE</MODNAME>
  <MODITEM CLASS="ID"><P><NUM BASE="10">1002</NUM></P></MODITEM>
  <MODITEM CLASS="INC"><P><NUM BASE="10">60000</NUM></P></MODITEM>
  <MODITEM CLASS="LPD"><P><DATE>12/02/93</DATE></P></MODITEM>
  <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
</MOD>
<MOD CLASS="CUST">
  <MODNAME CLASS="NAME">MIKE GIDENTO</MODNAME>
  <MODITEM CLASS="ID"><P><NUM BASE="10">1003</NUM></P></MODITEM>
  <MODITEM CLASS="INC"><P><NUM BASE="10">35000</NUM></P></MODITEM>
  <MODITEM CLASS="LPD"><P><DATE>12/12/92</DATE></P></MODITEM>
  <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
</MOD>
</MODINFO>

```

Attributes

CLASSNAME=*classname*

Specifies the class of modular information that is being defined.

Desc

Describes the purpose of the modular information class.

ModItemDef

Defines the classes of module description items valid for the class of modular information that is being defined.

Usage

See “Chapter 16. Defining Modular Information” on page 151.

Contexts

Children: “Desc (element description)” on page 251, “ModItemDef (item class definitions)”.

Parents: “ModInfo (modular information)” on page 338, “PropDefs (property definitions)” on page 396, “PropGroup (property group)” on page 397.

ModItemDef (item class definitions)

Purpose

The ModItemDef element defines the classes of module description items valid for a class of modular information.

Examples

```
<PROPDEF>
<MODINFODEF CLASSNAME="CUST">
<DESC>CUSTOMER INFORMATION</DESC>
<MODITEMDEF CLASSNAME="NAME">
  <TITLE>NAME</TITLE>
  <DESC>THIS IS THE FIRST AND LAST NAME OF THE CUSTOMER</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="CUSTID">
  <TITLE>ID</TITLE>
  <DESC>THIS IS THEIR CUSTOMER ID NUMBER</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="INC">
  <TITLE>INCOME</TITLE>
  <DESC>THIS IS THEIR ANNUAL ESTIMATED INCOME</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="LPD">
  <TITLE>LAST PURCHASE DATE</TITLE>
  <DESC>THIS IS THEIR LAST PURCHASE DATE</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="NOTES">
  <TITLE>NOTES</TITLE>
  <DESC>PERSONAL NOTES</DESC>
</MODITEMDEF>
</MODINFODEF>
</PROPDEF>
:
<MODINFO CLASS="CUST" STYLE="TABLE">
<MOD CLASS="CUST">
  <MODNAME CLASS="NAME">FRED SMITH</MODNAME>
  <MODITEM CLASS="ID"><P><NUM BASE="10">1000</NUM></P></MODITEM>
  <MODITEM CLASS="INC"><NUM BASE="10">40000</NUM></MODITEM>
  <MODITEM CLASS="LPDATE"><P><DATE>12/25/93</DATE></P></MODITEM>
  <MODITEM CLASS="NOTES"><P>BIG SPENDER</P></MODITEM>
</MOD>
<MOD CLASS="CUST">
  <MODNAME CLASS="NAME">SUZANNE STANLEY</MODNAME>
  <MODITEM CLASS="ID"><P><NUM BASE="10">1001</NUM></P></MODITEM>
  <MODITEM CLASS="INC"><P><NUM BASE="10">50000</NUM></P></MODITEM>
  <MODITEM CLASS="LPD"><P><DATE>11/22/92</DATE></P></MODITEM>
  <MODITEM CLASS="NOTES"><P>LIKES GAME SOFTWARE</P></MODITEM>
</MOD>
<MOD CLASS="CUST">
  <MODNAME CLASS="NAME">JEFF GEORGE</MODNAME>
  <MODITEM CLASS="ID"><P><NUM BASE="10">1002</NUM></P></MODITEM>
  <MODITEM CLASS="INC"><P><NUM BASE="10">60000</NUM></P></MODITEM>
  <MODITEM CLASS="LPD"><P><DATE>12/02/93</DATE></P></MODITEM>
  <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
</MOD>
<MOD CLASS="CUST">
  <MODNAME CLASS="NAME">MIKE GIDENTO</MODNAME>
  <MODITEM CLASS="ID"><P><NUM BASE="10">1003</NUM></P></MODITEM>
  <MODITEM CLASS="INC"><P><NUM BASE="10">35000</NUM></P></MODITEM>
  <MODITEM CLASS="LPD"><P><DATE>12/12/92</DATE></P></MODITEM>
  <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
</MOD>
</MODINFO>
```

Attributes

CLASSNAME=*classname*

Defines the class of module description item that is valid for a class of modular information.

Title

Contains the title for a given class of ModItem.

Desc

Describes the purpose or meaning of a given ModItem class.

Usage

See “Chapter 16. Defining Modular Information” on page 151.

Contexts

Children: “Desc (element description)” on page 251, “Title” on page 431.

Parents: “ModInfoDef (modular information property definition)” on page 340, “PropDefs (property definitions)” on page 396, “PropGroup (property group)” on page 397.

ModDesc (modular content description)

Purpose

The ModDesc element contains a description of the content of the Mod element of which it is a part.

Examples

```
<MODINFO ID="CMDEXIT">
<TITLE>AFTP Commands Exit the AFTP Environment</TITLE>
<MOD CLASS="CMDNORM">
  <MODNAME Class="COMMAND">bye</MODNAME>
  <MODDESC Class="DESCRIP">Alias for exit.</MODDESC>
  <MODITEM Class="PAGE">
    <XREF REFID="BYE" STYLE="BKM: (FORM='PAGEONLY') "></MODITEM>
  </MOD>
  <MOD CLASS="CMDNORM">
    <MODNAME CLASS="COMMAND">exit</MODNAME>
    <MODDESC CLASS="DESCRIP">Exits the AFTP environment.</MODDESC>
    <MODITEM CLASS="PAGE">
      <XREF REFID="EXIT" STYLE="BKM: (FORM='PAGEONLY') "></MODITEM>
    </MOD>
    :
  </MODINFO>
```

Attributes**%PhAndPLike.NoModInfo**

Contains a short description of the Mod element.

Usage

See “Chapter 16. Defining Modular Information” on page 151.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Attention (safety notice)” on page 214, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Danger (danger notice)” on page 243, “Date” on page 244, “Dec (decimal

number)" on page 247, "DL (definition list)" on page 253, "Fig (figure)" on page 262, "Formula (math formula)" on page 267, "GL (glossary list)" on page 272, "Hex (hexadecimal)" on page 277, "L (explicit link)" on page 300, "Lines (text with line boundaries)" on page 315, "Litdata (literal data)" on page 316, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "MMObj (multi-media object; artwork)" on page 333, "MV (message variable)" on page 355, "Note" on page 362, "NoteList (ordered note list)" on page 363, "Num (number)" on page 366, "Oct (octal number)" on page 369, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "Ph (Phrase)" on page 382, "PK (programming keyword)" on page 385, "PV (parameter variable)" on page 400, "Q (quotation phrase)" on page 402, "RefKey (reference key)" on page 405, "Screen (display screen)" on page 411, "SynPh (syntax phrase)" on page 419, "Syntax (syntax diagram)" on page 420, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "Mod (information module)" on page 336.

ModItem (module description item)

Purpose

The ModItem element contains one item in a module of information.

The ModItem elements within a Mod element contain the different kinds of information that are applicable for a given class of information module. The classes for the ModItem elements are defined in the ModItemDef element.

The style used to present ModItems is determined by the style specification of the Mod element which contains it.

Examples

```
<PROPDEFS>
<MODINFODEF CLASSNAME="CUST">
<DESC>CUSTOMER INFORMATION</DESC>
<MODITEMDEF CLASSNAME="NAME">
  <TITLE>NAME</TITLE>
  <DESC>THIS IS THE FIRST AND LAST NAME OF THE CUSTOMER</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="CUSTID">
  <TITLE>ID</TITLE>
  <DESC>THIS IS THEIR CUSTOMER ID NUMBER</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="INC">
  <TITLE>INCOME</TITLE>
  <DESC>THIS IS THEIR ANNUAL ESTIMATED INCOME</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="LPD">
  <TITLE>LAST PURCHASE DATE</TITLE>
  <DESC>THIS IS THEIR LAST PURCHASE DATE</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="NOTES">
  <TITLE>NOTES</TITLE>
  <DESC>PERSONAL NOTES</DESC>
</MODITEMDEF>
</MODINFODEF>
</PROPDEFS>
:
```

```

<MODINFO CLASS="CUST" STYLE="TABLE">
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">FRED SMITH</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1000</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><NUM BASE="10">40000</NUM></MODITEM>
    <MODITEM CLASS="LPDATE"><P><DATE>12/25/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>BIG SPENDER</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">SUZANNE STANLEY</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1001</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">50000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>11/22/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>LIKES GAME SOFTWARE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">JEFF GEORGE</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1002</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">60000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/02/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">MIKE GIDENTO</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1003</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">35000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/12/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
</MODINFO>

```

Attributes

CLASS=*classname*

Indicates which class of information the ModItem contains. The class values for a modular information item are defined on the ModItemDef element.

%PhAndPLike.NoModInfo

%DivLikeNoDiv.NoModInfo

Contains the description of the information for this ModItem.

Usage

See “Chapter 16. Defining Modular Information” on page 151.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Attention (safety notice)” on page 214, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Danger (danger notice)” on page 243, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “LERS (language element reference section)” on page 308, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “MsgList (list of message or code descriptions)” on page 352, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366

on page 364, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PartAsm (part assembly)” on page 379, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “Proc (procedure)” on page 388, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “Mod (information module)” on page 336.

ModLvl (modification level)

Purpose

The ModLvl element contains a program’s modification level.

Examples

```
<IBMPRODINFO>
<PRODDNAME>Test Prod</PRODDNAME>
<VERSION>2</VERSION>
<RELEASE>3</RELEASE>
<MODLVL>1</MODLVL>
<IBMPROGNUM>223-3330</IBMPROGNUM>
</IBMPRODINFO>
```

Attributes

#PCDATA

Contains the modification level.

Usage

See “Other prolog elements” on page 77.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “IBMProdInfo (IBM product information)” on page 291.

ModName (modular information element name)

Purpose

The ModName element contains the name of the information contained by the Mod element of which it is a part.

Examples

```
<PROPDEF>
<MODINFODEF CLASSNAME="CUST">
<DESC>CUSTOMER INFORMATION</DESC>
<MODITEMDEF CLASSNAME="NAME">
<TITLE>NAME</TITLE>
<DESC>THIS IS THE FIRST AND LAST NAME OF THE CUSTOMER</DESC>
</MODITEMDEF>
```

```

<MODITEMDEF CLASSNAME="CUSTID">
  <TITLE>ID</TITLE>
  <DESC>THIS IS THEIR CUSTOMER ID NUMBER</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="INC">
  <TITLE>INCOME</TITLE>
  <DESC>THIS IS THEIR ANNUAL ESTIMATED INCOME</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="LPD">
  <TITLE>LAST PURCHASE DATE</TITLE>
  <DESC>THIS IS THEIR LAST PURCHASE DATE</DESC>
</MODITEMDEF>
<MODITEMDEF CLASSNAME="NOTES">
  <TITLE>NOTES</TITLE>
  <DESC>PERSONAL NOTES</DESC>
</MODITEMDEF>
</MODINFODEF>
</PROPDEFS>
:
<MODINFO CLASS="CUST" STYLE="TABLE">
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">FRED SMITH</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1000</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><NUM BASE="10">40000</NUM></MODITEM>
    <MODITEM CLASS="LPDATE"><P><DATE>12/25/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>BIG SPENDER</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">SUZANNE STANLEY</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1001</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">50000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>11/22/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>LIKES GAME SOFTWARE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">JEFF GEORGE</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1002</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">60000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/02/93</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
  <MOD CLASS="CUST">
    <MODNAME CLASS="NAME">MIKE GIDENTO</MODNAME>
    <MODITEM CLASS="ID"><P><NUM BASE="10">1003</NUM></P></MODITEM>
    <MODITEM CLASS="INC"><P><NUM BASE="10">35000</NUM></P></MODITEM>
    <MODITEM CLASS="LPD"><P><DATE>12/12/92</DATE></P></MODITEM>
    <MODITEM CLASS="NOTES"><P>NONE</P></MODITEM>
  </MOD>
</MODINFO>

```

Attributes

%Title

Contains the module name.

Usage

For more information about the ModName element, see “Examples of Using Modular Information” on page 152.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “Mod (information module)” on page 336.

Msg (message or code description)

Purpose

The Msg element contains a message or code and its description.

Examples

```
<msglist>
<msg>
<msgnum>DJI7832E</msgnum>
<msgtext>This message is issued when no data set of
the name <mv>file-name</mv> is found.</msgtext>
<msgitem class="xpl">
<p>The processor could not locate the data set named <mv>
file-name</mv>.</p>
</msgitem>
<msgitem class="severity">
<p>8</p>
</msgitem>
<msgitem class="probd">
<p>You would appear to have a problem.</p>
</msgitem>
<msgitem class="uresp">
<p>Search high and low for the data set.</p>
</msgitem>
</msg>
<msg>
<msgtext>This message has no number</msgtext>
<msgitem class="xpl">
<p>This message has no message number; only text.
These are really insidious because it makes finding
the message very hard.</p>
</msgitem>
</msg>
</msglist>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Message and code lists” on page 29.

Contexts

Children: “Code (message code number)” on page 232, “MsgItem (message description item)”, “MsgNum (message identifier)” on page 353, “MsgText (message text)” on page 354, “Title” on page 431.

Parents: “MsgList (list of message or code descriptions)” on page 352.

MsgItem (message description item)

Purpose

The MsgItem element contains part of the description of a message or code.

Examples

```
<msglist>
<msg>
<msgnum>DJI7832E</msgnum>
<msgtext>This message is issued when no data set of
the name <mv>file-name</mv> is found.</msgtext>
<msgitem class="xpl">
<p>The processor could not locate the data set named <mv>
file-name</mv>.</p>
</msgitem>
<msgitem class="severity">
<p>8</p>
</msgitem>
<msgitem class="probd">
<p>You would appear to have a problem.</p>
</msgitem>
<msgitem class="uresp">
<p>Search high and low for the data set.</p>
</msgitem>
</msg>
<msg>
<msgtext>This message has no number</msgtext>
<msgitem class="xpl">
<p>This message has no message number; only text.
These are really insidious because it makes finding
the message very hard.</p>
</msgitem>
</msg>
</msglist>
```

Attributes

CLASS=*author-defined_class* | **DEST** | **XPL** | **EXPLANATION** | **MODULE** |
NUMBYTES | **ORESP** | **PREFIX** | **PRESP** | **PROBD** | **SEVERITY** | **SPRESP** |
SYSACT | **URES**

Indicates the class of the MsgItem as follows:

author-defined_class

Specifies an author-defined class, which must be defined using a
MsgItemDef element.

DEST

Specifies the message destination.

XPL or **EXPLANATION**

Specifies the message explanation.

MODULE

Specifies the issuing module.

NUMBYTES

Specifies the number of error bytes.

ORESP

Specifies the operator response.

PRESP

Specifies the programmer response.

PROBD

Specifies problem-determination information.

SEVERITY

Specifies the message severity.

SPRESP

Specifies the system-programmer response.

SYSACT

Specifies the system action.

URESP

Specifies the user response.

Usage

See “Message and code lists” on page 29.

Contexts

Children: “Annot (annotation)” on page 209, “AsmList (list of parts assemblies)” on page 214, “Attention (safety notice)” on page 214, “BibList (bibliography entry list)” on page 220, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “D (hierarchical division)” on page 242, “Danger (danger notice)” on page 243, “Dblk (Division block)” on page 245, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “FNList (footnote list)” on page 266, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “LERS (language element reference section)” on page 308, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MarkList (marked note list)” on page 321, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MsgList (list of message or code descriptions)” on page 352, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PartAsm (part assembly)” on page 379, “Pblk (paragraph block)” on page 380, “Proc (procedure)” on page 388, “Screen (display screen)” on page 411, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “UL (unordered list)” on page 436, “Xmp (example)” on page 440.

Parents: “Msg (message or code description)” on page 348.

MsgItemDef (definition of message description items)

Purpose

The MsgItemDef element defines the classes of message description items that are valid for a message list.

Examples

```
<msglist>
  <msgitemdef classname="xpl"><title>Why</title></msgitemdef>
  <msgitemdef classname="uresp"><title>What to do</title>
</msgitemdef>
  <msg>
    <msgnum>A12</msgnum>
    <msgtext>Closet full: Insufficient storage to proceed.
  </msgtext>
    <msgitem class="xpl">
      <p>There are too many clothes in the closet.</p>
    </msgitem>
    <msgitem class="uresp">
      <p>Remove some clothes from the closet and restart.
```

</p>
</msgitem>
</msg>
</msglist>

Attributes

CLASSNAME=*author-defined_class* | **DEST** | **EXPLANATION** | **MODULE** |
NUMBYTES | **ORESP** | **PREFIX** | **PRESP** | **PROBD** | **SEVERITY** | **SPRESP** |

SYSACT | **URESP**

Specifies the MsgItem class to which the definition applies as follows:

author-defined_class

Specifies an author-defined class, which must be defined using a
MsgItemDef element.

DEST

Specifies the message destination.

EXPLANATION

Specifies the message explanation.

MODULE

Specifies the issuing module.

NUMBYTES

Specifies the number of error bytes.

ORESP

Specifies the operator response.

PRESP

Specifies the programmer response.

PROBD

Specifies problem-determination information.

SEVERITY

Specifies the message severity.

SPRESP

Specifies the system-programmer response.

SYSACT

Specifies the system action.

URESP

Specifies the user response.

Usage

See “Message and code lists” on page 29.

Contexts

Children: “Desc (element description)” on page 251, “Title” on page 431.

Parents: “MsgList (list of message or code descriptions)” on page 352, “PropDefs
(property definitions)” on page 396, “PropGroup (property group)” on page 397.

MsgList (list of message or code descriptions)

Purpose

Use MsgList to contain descriptions of messages and codes. A MsgList contains one or more Msg elements, which contain the message or code and any explanatory text associated with it.

Examples

```
<msglist>
<msg>
<msgnum>DJI7832E</msgnum>
<msgtext>This message is issued when no data set of
the name <mv>file-name</mv> is found.</msgtext>
<msgitem class="xpl">
<p>The processor could not locate the data set named <mv>
file-name</mv>.</p>
</msgitem>
<msgitem class="severity">
<p>8</p>
</msgitem>
<msgitem class="probd">
<p>You would appear to have a problem.</p>
</msgitem>
<msgitem class="uresp">
<p>Search high and low for the data set.</p>
</msgitem>
</msg>
<msg>
<msgtext>This message has no number</msgtext>
<msgitem class="xpl">
<p>This message has no message number; only text.
These are really insidious because it makes finding
the message very hard.</p>
</msgitem>
</msg>
</msglist>
```

Attributes

RETKEY=None | NoDup

Use the RetKey attribute to enable or disable automatic running headings for this tag.

NONE

indicates that nothing is to be used

NODUP

indicates that the first and last non-blank items on the page are to be used. The two items are joined together, separated by a bullet or other character, and the combined string is used as the retrieval subject for the page. Use of FIRSTLAST provides the most dictionary-like retrieval of items. If the first and last items on the page are the same, only one of the items appears.

If you code any explicit RetKey elements, they are honored and will appear. If you nest elements that can generate a running heading (for example, a MsgList inside Lers), only the outer active generated heading is used. That is, if you specified automated RetKey generation for LERS and MSGLIST, a Msgno inside Lers will not be used in the RetKey area. But if you had an explicit RetKey inside the Msg, then the RetKey is honored as an explicit override.

Layout=Layout-Default | TwoCol | Page

This specifies how the message list should be formatted.

Layout-Default

Uses the layout based on the style. Most of the styles will have the default as two columns, but some of the smaller styles have a default of page-wide.

TwoCol

Composes the message list in two columns.

Page

Composes the message list page-wide.

Usage

See “Message and code lists” on page 29.

Contexts

Children: “Msg (message or code description)” on page 348, “MsgItemDef (definition of message description items)” on page 350, “RetKey (retrieval key)” on page 407.

Parents: “Appendix” on page 212, “Body (document body)” on page 222, “DBody (division body)” on page 246, “LEDI (language element description item)” on page 304, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348.

MsgNum (message identifier)

Purpose

The MsgNum element contains the number of a message.

Examples

```
<msglist>
<msg>
<msgnum>DJI7832E</msgnum>
<msgtext>This message is issued when no data set of
the name <mv>file-name</mv> is found.</msgtext>
<msgitem class="xpl">
<p>The processor could not locate the data set named <mv>
file-name</mv>.</p>
</msgitem>
<msgitem class="severity">
<p>8</p>
</msgitem>
<msgitem class="probd">
<p>You would appear to have a problem.</p>
</msgitem>
<msgitem class="uresp">
<p>Search high and low for the data set.</p>
</msgitem>
</msg>
<msg>
<msgtext>This message has no number</msgtext>
<msgitem class="xpl">
<p>This message has no message number; only text.
These are really insidious because it makes finding
```

```
the message very hard.</p>
</msgitem>
</msg>
</msglist>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Message and code lists” on page 29.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “Msg (message or code description)” on page 348.

MsgText (message text)

Purpose

The MsgText element contains the text of a message.

Examples

```
<msglist>
<msg>
<msgnum>DJI7832E</msgnum>
<msgtext>This message is issued when no data set of
the name <mv>file-name</mv> is found.</msgtext>
<msgitem class="xpl">
<p>The processor could not locate the data set named <mv>
file-name</mv>.</p>
</msgitem>
<msgitem class="severity">
<p>8</p>
</msgitem>
<msgitem class="probd">
<p>You would appear to have a problem.</p>
</msgitem>
<msgitem class="uresp">
<p>Search high and low for the data set.</p>
</msgitem>
</msg>
<msg>
<msgtext>This message has no number</msgtext>
<msgitem class="xpl">
<p>This message has no message number; only text.
These are really insidious because it makes finding
the message very hard.</p>
</msgitem>
</msg>
</msglist>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Message and code lists” on page 29.

Contexts

Children: text (#pcdata), “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Dec (decimal number)” on page 247, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “MV (message variable)”, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “SynPh (syntax phrase)” on page 419, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “Msg (message or code description)” on page 348.

MV (message variable)

Purpose

The MV element identifies data that is a placeholder for variable data in a message. It can be used either in the message text itself or in an explanation of the message.

Examples

```
<msglist>
<msg>
<msgnum>DJI7832E</msgnum>
<msgtext>This message is issued when no data set of
the name <mv>file-name</mv> is found.</msgtext>
<msgitem class="xpl">
<p>The processor could not locate the data set named <mv>
file-name</mv>.</p>
</msgitem>
<msgitem class="severity">
<p>8</p>
</msgitem>
<msgitem class="probd">
<p>You would appear to have a problem.</p>
</msgitem>
<msgitem class="uresp">
<p>Search high and low for the data set.</p>
</msgitem>
</msg>
<msg>
<msgtext>This message has no number</msgtext>
<msgitem class="xpl">
<p>This message has no message number; only text.
These are really insidious because it makes finding
the message very hard.</p>
</msgitem>
</msg>
</msglist>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Message and code lists” on page 29.

Contexts

Children: text (#pcdata).

| Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on
| page 214 , "Bridge (bridge between concepts)" on page 223, "Caution (caution
| notice)" on page 225, "CompCmt (component comment)" on page 234, "Danger
| (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "Desc
| (element description)" on page 251, "Entry (table entry)" on page 260, "Fn
| (footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311,
| "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on
| page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on
| page 331, "ModDesc (modular content description)" on page 343, "ModItem
| (module description item)" on page 344, "MsgText (message text)" on page 354,
| "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)"
| on page 382, "Q (quotation phrase)" on page 402, "SynNote (syntax note)" on
| page 418, "Term" on page 425, "Warning (warning notice)" on page 439, "Xmp
| (example)" on page 440, "XPh (example phrase)" on page 441.

Name (person's name)

Purpose

The Name element must always contains a person's name. It should not contain the name of a company (use the Company element for those).

Examples

```
<authors>
<author><person>
<name>Fred Mertz</name>
<address>
127 East Main Street,
East Overshoe, SD <postalcode>59134</postalcode>
</address>
</person></author>
</authors>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See "Author and Address" on page 77.

Contexts

Children: text (#pcdata), "Ph (Phrase)" on page 382.

Parents: "Person (person's name and address)" on page 381.

NameLoc (named location)

Purpose

The NameLoc element associates a local ID (defined within the same document) with other locations. These other locations may be IDs within the same document, other documents, or other entities. These are referenced using named list (NMList) elements.

NameLoc is the standard HyTime mechanism for creating indirect links. NameLoc creates indirect links and can be used with all links. The indirection provided by NameLoc:

- Associates a single ID with several objects (either other SGML elements or entities).
- Associates a local ID with objects in another document or subdocument.

NameLoc is required for linking to multiple locations.

Usually NameLoc contains a single NMList element that creates a simple cross-document or multiple-object link. However, NameLoc can contain several NMList elements. For example, you can create a link to several targets in different documents by using a new NMList for the targets in each different document.

Examples

The first example shows a simple NameLoc that creates a cross-document link. Note that the entity named on the DOCNAME attribute must be declared as a data entity in the document's SGML prolog.

```

:
<!ENTITY iddocref PUBLIC
    "-//ISBN 0-933186::IBM//NONSGML IBMIDDoc User's Guide and
    Reference//EN" NDATA SGMLDoc >
:
<LDESCS>
  <NAMELOC ID="nmlocref">
    <NMLIST DOCNAME="iddocref">iddoc</NMLIST>
  </NAMELOC>
</LDESCS>
:
<P>The <L LINKEND="nmlocref">NameLoc</L> element is
the workhorse of the HyTime architecture.
```

In this example, the NameLoc element contains a single NMList element. The NMList element associates the target ID (in this case the ID of the reference entry for NameLoc) with the document it is in (the document represented by the entity named *iddocref*). The L element itself points to the NameLoc element, which then serves to locate the actual target, the element with the ID "iddoc" in the document *iddocref*.

NItem contains information that refers the reader to a location in the document that contains information that all readers should know before using the document.

Attributes

ID=*nameloc_ID*

Specifies the local ID of the NameLoc element, for the location referenced by the NameLoc element.

OBJTYPE=*target_type*

Specifies the type of object being referenced.

This attribute must contain one of the following type names.

HEAD

A section with a heading, as defined by BookMaster heading tags or heading equivalents. If the OBJTYPE attribute is omitted, HEAD is assumed.

BOOK

For interdocument links, an entire book. Here the DOCID is used instead of the OBJECT attribute to tell BookManager what link to create. This value is not used for intradocument links.

FIG | TABLE | QUES | ANS

A figure, table, question, or answer.

STEP | CI | LI | SPOT

A step, component item, list item, or spot.

PROGRAM | ANIMATION | VIDEO | AUDIO | GRAPHIC | IMAGE

Information that is not in BookMaster documents. Access to these types of information depends on the capabilities of the user's installation.

NMList

Specifies an NMList element that contains the IDs or entity names of the location being referred to by the NameLoc element.

Usage

See “Chapter 11. All about linking” on page 109.

Contexts

Children: “NMList (named list of IDs or entities)” on page 360.

Parents: “LDescs (link descriptions)” on page 302.

NItem (notice item)

Purpose

The NItem element contains one or more elements that contain special notice information.

Examples

```

:
<FRONTM>
<NOTICES>
  <NITEM><PBLK STYLE='LBLBOX'>
    <TITLE>TAKE NOTE!</TITLE>
    <P>BEFORE USING THIS INFORMATION AND THE PRODUCT IT
      SUPPORTS, BE SURE TO READ THE GENERAL INFORMATION UNDER <XREF
      REFID="NOTICES">.</P>
  </NOTICES>
<EDNOTICES>FIFTH EDITION (AUGUST 1992)
  <P>This edition applies to Release 4 of IBM ..
  .</P>
  <P>Order publications through your IBM ...</P>
  <P>A form for reader's comments is provided ...
  </P>
</EDNOTICES>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Notices and Edition notices” on page 86.

Contexts

Children: “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Table” on page 423, “UL (unordered list)” on page 436.

Parents: “Notices (contains notices)” on page 364.

NMList (named list of IDs or entities)

Purpose

The NMList element is used within a NameLoc element to associate the NameLoc element's ID with the IDs of other elements or the IDs of other elements in other documents.

NMList also uses the DOCNAME attribute to specify the document or subdocument where the IDs are located.

NMList is used within NameLoc to contain the names (IDs or entity names) of the objects to be located, either SGML elements or entities. When the objects to be located are in the same document as the NMList element, NMList contains a list of their IDs or entity names. When the objects to be located are in another document or subdocument, specify the DocName attribute to indicate the document that defines or declares the target.

You can use several NMList elements within a single NameLoc. If objects are in the same document as the NMList element and in other documents, you need to specify a new NMList for each different document that defines or declares the target.

Examples

The first example shows a simple NMList used to create a cross-document link. Note that the entity named on the DOCNAME attribute has been declared as a data entity in the document's SGML prolog.

```

:
<!ENTITY iddocref PUBLIC
    "+//ISBN 0-933186::IBM//NONSGML IBMIDDoc User's Guide and
    Reference//EN" NDATA SGMLDoc >
:
<LDESCS>
  <NAMELOC ID="nmlocref">
    <NMLIST DOCNAME="iddocref">iddoc</NMLIST>
  </NAMELOC>
</LDESCS>
:
<P>The <L LINKEND="nmlocref">NameLoc</L> element is....
```

In this example, the NameLoc element, which is within an LDescs element, contains a single NMList element. The NameLoc element associates the target ID (in this case the ID of the reference entry for NMList) with the document it is in (the document represented by the entity named "iddocref"). The L element itself points to the NameLoc element, which then serves to locate the actual target, the element with the ID "iddoc" in the document "iddocref".

Attributes

NAMETYPE=ELEMENT | ENTITY

Indicates the contents of the NMList as follows:

ENTITY

Indicates that the names listed in the content of the NMList element are declared entity names.

ELEMENT

Indicates that the names listed in the content of the NMList element are element IDs in a document.

DOCNAME=*entity_name*

Specifies the name of the document or subdocument entity that contains the IDs or declares the entities listed in the content of the NMList element. The entity named must be declared in the current document.

If DOCNAME is not specified, the IDs in the NMList element are valid in the current document. If DOCNAME is specified, the IDs in the NMList element are valid for the document named on the DOCNAME attribute.

Note: The DOCNAME attribute is the HyTime *docorsub* attribute.

element_id

Specifies one or more element IDs of the elements to be located. When the unified name space option is in effect (either because the HyTime *unmspace* attribute was specified as Unified on the document element or because the Nametype attribute value is Unified), the IDs can also be the names of entities to be located.

DTDORLPD

Names the DTD that defines the element types or entity names used in the list. This should almost never be used. Used in conjunction with DOCNAME, it identifies a parsing context for interpreting the names in the list. The defaults will handle the vast majority of cases.

OBNAMES=OBNAMES | NOOBNAMES

OBNAMES means that referencing one location actually is an indirect reference to another location.

NOOBNAMES means that the object pointed to is the actual content of the NameLoc element that is the target.

element_id

Specifies one or more element IDs of the elements to be located.

Future Enhancement

When the unified name space option is in effect (either because the HyTime *unmspace* attribute was specified as Unified on the document element or because the NAMETYPE attribute value is Unified), the IDs can also be the names of entities to be located.

CDATA

Contains character data.

Usage

See “Chapter 11. All about linking” on page 109.

Contexts

Children: text (#pcdata).

Parents: “NameLoc (named location)” on page 357.

|

Note

Purpose

The Note element contains a information that is differentiated from the main text. Information contained in a note often further explains the meaning of the main text. Use Note to create a note of one or more paragraphs.

Examples

```
<note><notebody>Thinking of a seashore, green meadow,  
or cool mountain overlook can help you to relax and  
be more patient.</notebody></note>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Notes” on page 37.

Contexts

Children: “NoteBody (note body)”, “Title” on page 431.

Parents: “AnnotBody (annotation body)” on page 210, “Bridge (bridge between concepts)” on page 223, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “EdNotices (edition notices)” on page 259, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “LeDesc (language element description)” on page 304, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NItem (notice item)” on page 359, “Notices (contains notices)” on page 364, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “Safety (safety notices)” on page 411, “Sem (semantic meaning)” on page 412, “SynNote (syntax note)” on page 418, “TextAlt (text alternative)” on page 427.

NoteBody (note body)

Purpose

The NoteBody element contains the body of the Note information that is differentiated from the main text.

Examples

```
<note><notebody>Thinking of a seashore, green meadow,  
or cool mountain overlook can help you to relax and  
be more patient.</notebody></note>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Notes” on page 37.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “Note” on page 362.

NoteList (ordered note list)

Purpose

The NoteList element contains an ordered list of notes.

Examples

```
<notelist>
<li>Make a To Do list</li>
<li>Prioritize sensibly</li>
<li>Avoid interruptions where possible</li>
<li>Check on your progress toward monthly goals</li>
<li>Plan for the next work week</li>
<li>Do something for the fun of it</li>
<li>Spend some quality time with your pet</li>
</notelist>
```

Attributes

LINESPACE=SPACE | COMPACT

Specifies whether the items in the list should be compacted or have space between the items. Nested lists automatically inherit the setting of the outer list, but can override that default with their own setting.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Note lists” on page 38.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “LI (list item)” on page 311, “LIBlk (list item block)” on page 314, “Title” on page 431.

Parents: “AnnotBody (annotation body)” on page 210, “Bridge (bridge between concepts)” on page 223, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “EdNotices (edition notices)” on page 259, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “LeDesc (language element description)” on page 304, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NItem (notice item)” on page 359, “Notices (contains notices)”, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “Safety (safety notices)” on page 411, “Sem (semantic meaning)” on page 412, “SynNote (syntax note)” on page 418.

Notices (contains notices)

Purpose

The Notices element contains one or more NItem elements that contain special notice information.

Examples

```
<notices><pblk style="lblbox"><title>Note</title>
<p>Before using this information, be sure to read
the general information under <xref refid="notices">.
</p>
<p>This manual was produced using IBMIDDoc SGML, the
Adept editor, and processed for print and online using
the ID Workbench.</p>
</pblk></notices>
<ednotices>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Notices and Edition notices” on page 86.

Contexts

Children: “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “NItem (notice item)” on page 359, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Table” on page 423, “UL (unordered list)” on page 436.

Parents: “FrontM (front matter)” on page 270.

Notloc (notation-specific location)

Purpose

The Notloc element contains a specification of an address that uses a specific notation. It enables the use of definition of anchors in non-SGML objects using specifications that are peculiar to those objects.

In the current level of IBMIDDoc, NotLoc is used to encode URLs.

The Notloc element is a HyTime element.

Use Notloc to define link anchors in non-SGML objects such as images and vector graphics, or using query specifications in a specific notation. For example, it can contain x and y pixel locations within a bitmap or the label of a graphic object in a CAD drawing. A link anchor can be the start of a link or the target of a link or both. How a Notloc-defined anchor is used and expressed depends on your online presentation system and is not defined by IBMIDDoc.

Examples

```
<IBMIDDOC>
<PROLOG>

:
<LDESCS>
  <NOTLOC ID="ibmwww" notation="url">
    http://www.ibm.com
  </NOTLOC>
</LDESCS>

:
<BODY>

:
<P PROPS="www">Be sure to check out the
<L LINKEND="ibmwww"> for the latest IBM product information.
</P>
```

Attributes

ID=*notloc_id*

Contains the ID of this Notloc element.

NOTATION=*notation_name*

Specifies the notation of the address specification. It must be a declared SGML notation.

Usage

See “Chapter 11. All about linking” on page 109.

Contexts

Children: text (#pcdata).

Parents: “LDescs (link descriptions)” on page 302.

Num (number)

Purpose

Use the Num element to identify numbers in a base for which a more precise element, such as Hex or Bin, does not exist. You must specify the Base attribute to indicate the base of the number, for example, "36" for base 36 numbers.

Examples

<P>Nums in base 34 use the digits zero (0) to nine (9) and the letters A to Z minus I and O (or L and 0), for example, <NUM BASE="34">Z</NUM> = <DEC>33</DEC>.

Attributes

BASE=*basevalue*

Contains an integer value specifying the base of the number.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Highlighting” on page 35.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439.

ObjLib (object library)

Purpose

The ObjLib element contains elements that are to be used elsewhere in the document by reference. You can have as many different ObjLib elements as desired. Use object libraries to collect elements for reuse. Elements in object libraries are not processed at the place they occur in the source document but are processed only when referred to by another element of the same type using the CONLOC or RETALTS attributes. The ObjLib needs to be specified in the document's Prolog.

Use the optional Desc element to describe the purpose of the library.

Migration Note

ObjLib replaces the function provided by DVCF side files for organizing collections of text for re-use.

Examples

```
<OBJLIB>
  <OBJLIBBODY>
    <DLENTY ID="FILEMENUITEM" CLASS="menuitem">
      <TERM>File</TERM>
      <DEFN>Work with files.</DEFN>
    </DLENTY>
    <DLENTY ID="Editmenuitem" CLASS="menuitem">
      <TERM>Edit</TERM>
      <DEFN>Perform edit functions.</DEFN>
    </DLENTY>
  </OBJLIBBODY>
</OBJLIB>
```

Attributes

See "Common Element Attributes (large set)" on page 204.

Usage

See "Reusing elements from an object library" on page 166.

Contexts

Children: "Desc (element description)" on page 251, "ObjLibBody (object library body)".

Parents: "DProlog (division prolog)" on page 256, "Prolog (document metainformation)" on page 395, "SpecDProlog (special section division prolog)" on page 415.

ObjLibBody (object library body)

Purpose

The ObjLibBody element contains the elements in an object library.

The OBJLibBody contains all of the elements in the object library.

Examples

```
<OBJLIB>
<OBJLIBBODY>
  <DENTRY ID="FILEMENUITEM" CLASS="menuitem">
    <TERM>File</TERM>
    <DEFN>Work with files.</DEFN>
  </DENTRY>
  <DENTRY ID="Editmenuitem" CLASS="menuitem">
    <TERM>Edit</TERM>
    <DEFN>Perform edit functions.</DEFN>
  </DENTRY>
</OBJLIBBODY>
</OBJLIB>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Reusing elements from an object library” on page 166.

Contexts

Children: any element.

Parents: “ObjLib (object library)” on page 367.

ObjRef (object reference)

Purpose

The ObjRef element references a declared graphic entity. ObjRef has no content; it only references the graphic entity using the entity’s declared name.

When using an SGML editor, such as ArborText’s ADEPT*Editor, that supports the inline display of certain graphic types, the graphic entity will be displayed inline with your text.

Examples

```
<!ENTITY bike system "bike.gif" ndata graphics>
...
<MMOBJ>
  <OBJREF OBJ="bike">
    <TEXTALT>This is a two-wheeled bicycle.</TEXTALT>
  </MMOBJ>
```

Attributes

OBJ=*entity_name*

This attribute’s value is the name of the graphic entity containing the graphic to be included.

EDITSCALE=*scaling_value*

The EDITSCALE attribute specifies the scaling factor for graphics in an SGML editor capable of displaying graphics inline.

WIDTH and DEPTH

WIDTH and DEPTH attributes allow you to expand or contract a drawing. They accept the following values:

x.yin

inches. For example: 4in or 5.5in.

x.ypt

points. For example: 48pt or 39.5pt.

x.ypi

picas. y is base 12. For example, 5.11pi is 5 picas, 11 points.

x.ymm

millimeters. For example: 55mm or 47.6mm.

x.ycm

centimeters. For example: 12cm or 10.7cm.

ScaleBox=BestFit | DepthFirst | UseBoth | None

Defines how to use the width and depth values to scale an object.

BestFit

Scaled to fit specified area without skewing the artwork. This is the default value.

DepthFirst

The depth value will be chosen first if the composer must skew the object.

UseBoth

Sets to the depth and width values specified. If any one value is specified, it will be used without skewing.

None

No scaling.

Usage

See “Including artwork in documents” on page 45.

Contexts

Children: empty.

Parents: “MMObj (multi-media object; artwork)” on page 333.

Oct (octal number)

Purpose

Use the Oct element to identify octal data, which is encoded in a base-8 numbering system.

Examples

<BIN>11000001</BIN> = <DECIMAL>193</DECIMAL> = <OCT>301</OCT>

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Highlighting” on page 35.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439.

OL (ordered list)

Purpose

The OL element contains a list of items where the order of the items has particular significance, or where the ordinal number of each item is significant.

Examples

```
<ol>
<li>Cream butter and sugar together until fluffy.</li>
<li>Beat in egg yolks one at a time.</li>
<li>Add nutmeg, cinnamon, and vanilla, and mix thoroughly.
The batter should be smooth and glossy and stream
off the spoon in ribbons.</li>
<li>Fold in beaten egg whites.
<p>Do not overmix; the batter should be light and fluffy.</p></li>
</ol>
```

Attributes

LINESPACE=SPACE | COMPACT

Specifies whether the items in the list should be compacted or have space between the items. Nested lists automatically inherit the setting of the outer list, but can override that default with their own setting.

OLType

This attribute specifies the list type. Allowed values are:

Normal

Causes a normal, ordered list.

Step Causes a list with the word “Step” to appear before the step item.

Checkoff

Causes a small check-off area to appear before the step item.

CheckoffStep

Causes both a small check-off area and the word “Step” to appear before the step item.

Seq

Specifies that a sequence of ordered lists are to connect. That is, the list can

end, but when the list starts again, the numnbering continues from the previous list's last item. Use this for steps that need to cross divisions or table cells. The value START indicates the beginning of the list; the value END indicates the end of the list. The ID on the list must appear on the SEQID attributes on the continuing lists.

SEQID

Indicates the list is part of a sequence. The SEQID points to the ID of the beginning list.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Ordered lists" on page 24.

Contexts

Children: "Bridge (bridge between concepts)" on page 223, "LI (list item)" on page 311, "LIBlk (list item block)" on page 314.

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "BackCover (back cover)" on page 217, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "Cond (procedure result)" on page 235, "CopyR (copyrights)" on page 237, "Danger (danger notice)" on page 243, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "EdNotices (edition notices)" on page 259, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "Fn (footnote)" on page 265, "FrontCover" on page 270, "LeDesc (language element description)" on page 304, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "NItem (notice item)" on page 359, "NoteBody (note body)" on page 362, "Notices (contains notices)" on page 364, "P (paragraph)" on page 374, "PBlk (paragraph block)" on page 380, "ProcEntry (procedure entry point)" on page 390, "ProcExit (procedure exit point)" on page 391, "ProcIntro (procedure introduction)" on page 391, "Safety (safety notices)" on page 411, "Sem (semantic meaning)" on page 412, "SynNote (syntax note)" on page 418, "TextAlt (text alternative)" on page 427, "Warning (warning notice)" on page 439.

Oper (syntax operator)

Purpose

Use Oper to define an operator within a syntax definition. Operators usually connect two parts of a statement and imply an action such as assignment or comparison. Operators can also be applied to a single parameter, such as the negation operator. Typical operators are the equals sign (=) and the mathematical operators such as add (+) and multiply (*).

Examples

```
<syntax>
<group>
<kwd>LANGUAGE</kwd>
```

```
<oper>=</oper>
<var>language_name</var>
</group>
</syntax>
```

Attributes

OPTREQ= OPT|REQ|DEF

Indicates whether the operator is optional, required, or the default.

LINKEND=*id*

Contains an ID reference that enables a link to an arbitrary location in the document.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 14. Program Syntax Definitions” on page 125.

Contexts

Children: text (#pcdata).

Parents: “Group” on page 277, “SynPh (syntax phrase)” on page 419.

OrderNum (order number)

Purpose

The OrderNum element contains the order number assigned to a document. This element is for use by non-IBM documents. IBM documents use the IBMDocNum element (see “IBMDocNum (IBM document number)” on page 280).

Examples

```
<ORDERNUM>GC12-3456-00</ORDERNUM>
```

Attributes

#PCDATA

Contains the order number.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “BibEntry (bibliographic entry)” on page 218, “LibEntry (document library definition)” on page 312.

OrigIBMDocNum (original IBM document number)

Purpose

Use this when you have a new manual that superceeds a previous manual.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “IBMBibEntry (IBM bibliographic entry)” on page 279.

Owners

Purpose

Contains the name of the owner or owners of the information.

Examples

```
<OWNERS>
  <PERSON>
    <NAME>John Smith</NAME>
    <ADDRESS>XYZ Corp. RTP, NC 27709</ADDRESS>
  </PERSON>
  <PERSON>
    <NAME>Susan Jones</NAME>
    <ADDRESS>ABC Corp. RTP, NC</ADDRESS>
  </PERSON>
</OWNERS>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “Corp (enterprise name and address)” on page 238, “Person (person’s name and address)” on page 381.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

P (paragraph)

Purpose

The P element contains a paragraph; a block of text representing a single idea. Use paragraphs to contain flowing text and other elements associated with the text, such as lists, examples, figures, and tables.

Paragraphs with no content are not valid.

Examples

This example shows a simple paragraph:

```
<P>This is a simple paragraph.</P>
```

Use the paragraph end tag to control whether or not other elements are contained within the paragraph. In this example, the unordered list is contained within the paragraph:

```
<P>This paragraph contains a list:
  <UL>
    <LI>List item</LI>
    <LI>List item</LI>
  </UL>
</P>
```

To keep the list from being part of the paragraph, the paragraph must be ended before the list begins, as in this example:

```
<P>This paragraph does not contain the list.</P>
<UL>
  <LI>List item</LI>
</UL>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating paragraphs (P element)” on page 18.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “ParmL (parameter list)” on page 376, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax

phrase)" on page 419, "Syntax (syntax diagram)" on page 420, "Table" on page 423, "Term" on page 425, "TM (Trademark)" on page 433, "UL (unordered list)" on page 436, "Xmp (example)" on page 440, "XPh (example phrase)" on page 441, "XRef (cross reference)" on page 442.

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "BackCover (back cover)" on page 217, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "Cond (procedure result)" on page 235, "CopyR (copyrights)" on page 237, "Danger (danger notice)" on page 243, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "EdNotices (edition notices)" on page 259, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "Fn (footnote)" on page 265, "FrontCover" on page 270, "LeDesc (language element description)" on page 304, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "NItem (notice item)" on page 359, "NoteBody (note body)" on page 362, "Notices (contains notices)" on page 364, "PBlk (paragraph block)" on page 380, "ProcEntry (procedure entry point)" on page 390, "ProcExit (procedure exit point)" on page 391, "ProcIntro (procedure introduction)" on page 391, "Safety (safety notices)" on page 411, "Sem (semantic meaning)" on page 412, "SynNote (syntax note)" on page 418, "TextAlt (text alternative)" on page 427, "Warning (warning notice)" on page 439.

Parm (parameter list entry)

Purpose

The Parm element contains a single parameter and its definition within a parameter list.

Examples

```
<parml>
<parm><term>KEYWORD = <pk optreq="DEF">DEFAULT</pk>|VALUE
</term>
<defn>This is the description of the parameter above.
It could go on for many pages, if necessary. (Of course,
that means we have a very complicated parameter to
describe.)</defn>
</parm>
<parm><term>KEYWORD2 = &lbrk;ABC|XYZ&rbrk;</term>
<term>&lbrk;KEYWORD3 = GGG&rbrk;</term>
<defn>This description applies to the two parameters
above. Often in examples of programming syntax, it
is necessary to use symbols for the brackets and braces.
</defn>
</parm>
<parm><term><synph><kwd>KEYWORD3</kwd></synph></term>
<defn>Here's a term that uses the syntax phrase (SYNPH);
it allows you to use the same items as a syntax diagram.
</defn>
</parm>
</parml>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Parameter lists” on page 28.

Contexts

Children: “Defn (definition of a term)” on page 249, “Term” on page 425.

Parents: “ParmBlk (parameter list block)”, “ParmL (parameter list)”.

ParmBlk (parameter list block)

Purpose

The ParmBlk element organizes parameter list entries into meaningful groupings. For example, if you group parameters within the definition of a complex statement or command, you can use ParmBlk in the parameter list to mirror the grouping in the syntax definition.

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Grouping list items” on page 32.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “Parm (parameter list entry)” on page 375, “Title” on page 431.

Parents: “ParmL (parameter list)”.

Examples

```
<parml>
  <parmblk>
    <parm><term>one term</term>
    <defn>definition</defn>
  </parm>
  <parm><term>another term</term>
  <defn>definition</defn>
</parmblk>
<parm><term>yet another term</term>
<defn>definition</defn>
</parm>
</parml>
```

ParmL (parameter list)

Purpose

Use parameter lists to describe the parameters in a computer language statement. Parameter lists are usually associated with syntax definitions. Entries can be

organized within parameter lists using ParmBlk elements. Bridge elements can also be used to create connections between blocks of entries, including syntax definitions.

Examples

```
<parml>
<parm><term>KEYWORD = <pk optreq="DEF">DEFAULT</pk>|VALUE
</term>
<defn>This is the description of the parameter above.
It could go on for many pages, if necessary. (Of course,
that means we have a very complicated parameter to
describe.)</defn>
</parm>
<parm><term>KEYWORD2 = &lbrk;ABC|XYZ&rbrk;</term>
<term>&lbrk;KEYWORD3 = GGG&rbrk;</term>
<defn>This description applies to the two parameters
above. Often in examples of programming syntax, it
is necessary to use symbols for the brackets and braces.
</defn>
</parm>
<parm><term><synph><kwd>KEYWORD3</kwd></synph></term>
<defn>Here's a term that uses the syntax phrase (SYNPH);
it allows you to use the same items as a syntax diagram.
</defn>
</parm>
</parml>
```

Attributes

TERMWIDTH= SMALL | MEDIUM | LARGE | 1 | 2

You can use the TERMWIDTH attribute to determine the indentation size of the definition list. The valid choices are: small (.25 inch, the default), medium (.5 inch), and large (1 inch). The value "1" is for 1-character width; "2" is for a 2-character width.

LINESPACE=SPACE | COMPACT

Specifies whether the items in the list should be compacted or have space between the items. Nested lists automatically inherit the setting of the outer list, but can override that default with their own setting.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Parameter lists" on page 28.

Contexts

Children: "Bridge (bridge between concepts)" on page 223, "DefnHd (definition description heading)" on page 250, "Parm (parameter list entry)" on page 375, "ParmBlk (parameter list block)" on page 376, "TermHd (term heading)" on page 426.

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "Danger (danger notice)" on page 243, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "Fn (footnote)" on page 265, "LEDI (language element description item)" on page 304

on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

Part (major document part)

Purpose

Use Part to divide a document’s chapters into logical groupings. For example, in a document that contains both guide and reference information, you can define two parts, one containing the guide information and the other containing the reference information.

The Part element does not change the logical hierarchy of the divisions it contains. For example, if, in your document style, first-level divisions are considered to be chapters, they are still chapters when contained within Part. Thus, the enumeration of divisions contained within Parts is not affected by the presence or absence of Part elements.

Examples

```
<ibmiddoc>
<body>
  <part>
    <dprolog><titleblk>
      <title>Introduction</title>
    </titleblk></dprolog>
    <dbody>
      <d>
        <dprolog><titleblk>
          <title>Salads of our neighborhood</title>
        </titleblk></dprolog>
        <dbody></dbody></d>
      <d>
        <dprolog><titleblk>
          <title>Salads of the world</title>
        </titleblk></dprolog>
        <dbody></dbody></d>
      </dbody></part>
    <part>
      <dprolog><titleblk>
        <title>Recipes</title>
      </titleblk></dprolog>
      <dbody>
        <d>
          <dprolog><titleblk>
            <title>Egg salad</title>
          </titleblk></dprolog>
          <dbody></dbody></d>
        <d>
          <dprolog><titleblk>
            <title>Tuna fish salad</title>
          </titleblk></dprolog>
          <dbody></dbody></d>
        </dbody></part></body>
  </ibmiddoc>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using parts to organize your chapters” on page 22.

Contexts

Children: “Abstract (abstract)” on page 207, “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DProlog (division prolog)” on page 256, “DSum (division summary)” on page 257.

Parents: “Body (document body)” on page 222.

PartAsm (part assembly)

Purpose

The PartAsm element contains the elements needed to construct a parts assembly list.

Examples

```
<partasm id="bike" style="bkm:(layout=same)"><title>
Bicycle</title><mmobj><objref obj="bike">
<textalt>Bicycle</textalt>
</mmobj><compl>
<ci idxnum="1" partnum="4563423" upa="1">Bike</ci>
</compl>
<ci idxnum="1" partnum="1230987" upa="1">Frame</ci>
<ci idxnum="2" partnum="1238475" upa="1">Wheel assembly, front</ci>
<compcmt>For detailed breakdown, see <xref refid="wheelxmp">.</compcmt>
<ci idxnum="3" partnum="1234939" upa="1">Wheel assembly, rear</ci>
</compl>
</compl>
</partasm>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 21. Creating parts catalog lists” on page 191.

Contexts

Children: “CompL (component list)” on page 234, “MMObj (multi-media object; artwork)” on page 333, “PartAsmSeg (part assembly segment)” on page 380, “RetKey (retrieval key)” on page 407, “Title” on page 431.

Parents: "Appendix" on page 212, "Body (document body)" on page 222, "DBody (division body)" on page 246, "LEDI (language element description item)" on page 304, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348.

PartAsmSeg (part assembly segment)

Purpose

The PartAsmSeg element contains the component elements needed to contain the parts assembly information that is a logical division of the assembly.

Attributes

See "Common Element Attributes (large set)" on page 204.

Contexts

Children: "CompL (component list)" on page 234, "MMObj (multi-media object; artwork)" on page 333.

Parents: "PartAsm (part assembly)" on page 379.

PBlk (paragraph block)

Purpose

Use PBlk to group paragraphs and paragraph-like elements together. You can use a Title element on the PBlk element to identify or introduce the topic that the paragraphs in the PBlk address. PBlk can be used to define property values for the set of contained elements. For example, if a number of paragraphs have changed, you can put them within a PBlk element in order to define their revision status.

You can also use PBlk within an ObjLib to define groups of paragraphs for use by reference. For example, if you have a figure with an introductory paragraph that you want to use in several contexts, you can put the paragraph and the figure into a PBlk. To use the content of that PBlk in several places, you can specify a PBlk with a CONLOC attribute that refers to the PBlk you want to reuse.

PBlk also enables you to define a set of paragraphs as a single link anchor by linking to the PBlk element.

To create a labeled box, use attribute style="lblbox" on the PBlk tag:

```
<pbk style="lblbox"><title>Getting There</title>
<p>To get to...
</p></pbk>
```

To create hidden text in IPF and Windows, use attribute style="hidden" on the PBlk tag:

Attributes

style=hidden

A PBLK with style=hidden and formatted for IPF or RTF becomes a hidden division. Be sure to specify a title for the PBLK or you will get *** for the title of the generated division.

```
<pbk style="hidden"><title>Getting There</title>
<p>To get to...
</p></pbk>
```

style=lblbox

This causes a labeled box to surround the content. Use a Title tag to specify the title for the labeled box.

```
<pbk style="lblbox"><title>Getting There</title>
<p>To get to...
</p></pbk>
```

Contexts

Children: "Annot (annotation)" on page 209, "AsmList (list of parts assemblies)" on page 214, "Attention (safety notice)" on page 214, "BibList (bibliography entry list)" on page 220, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "CGraphic (character graphic)" on page 226, "Danger (danger notice)" on page 243, "DL (definition list)" on page 253, "Fig (figure)" on page 262, "FNList (footnote list)" on page 266, "GL (glossary list)" on page 272, "L (explicit link)" on page 300, "Lines (text with line boundaries)" on page 315, "Litdata (literal data)" on page 316, "LQ (stand-alone quotation)" on page 318, "MarkList (marked note list)" on page 321, "MkNote (marked note)" on page 331, "MMObj (multi-media object; artwork)" on page 333, "ModInfo (modular information)" on page 338, "Note" on page 362, "NoteList (ordered note list)" on page 363, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "ParmL (parameter list)" on page 376, "PBlk (paragraph block)" on page 380, "Screen (display screen)" on page 411, "Syntax (syntax diagram)" on page 420, "Table" on page 423, "Title" on page 431, "UL (unordered list)" on page 436, "Xmp (example)" on page 440.

Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "Bridge (bridge between concepts)" on page 223, "Caution (caution notice)" on page 225, "Cond (procedure result)" on page 235, "CopyR (copyrights)" on page 237, "Danger (danger notice)" on page 243, "DBody (division body)" on page 246, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "DIntro (division introduction)" on page 252, "DSum (division summary)" on page 257, "EdNotices (edition notices)" on page 259, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "Fn (footnote)" on page 265, "FrontCover" on page 270, "LeDesc (language element description)" on page 304, "LEDI (language element description item)" on page 304, "LI (list item)" on page 311, "LQ (stand-alone quotation)" on page 318, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "MsgItem (message description item)" on page 348, "NItem (notice item)" on page 359, "NoteBody (note body)" on page 362, "Notices (contains notices)" on page 364, "PBlk (paragraph block)" on page 380, "ProcEntry (procedure entry point)" on page 390, "ProcExit (procedure exit point)" on page 391, "ProcIntro (procedure introduction)" on page 391, "Safety (safety notices)" on page 411, "Sem (semantic meaning)" on page 412, "SynNote (syntax note)" on page 418, "TextAlt (text alternative)" on page 427, "Warning (warning notice)" on page 439.

Person (person's name and address)

Purpose

The Person element contains name and address pairs for use in Author, Approvers, and Owners where either a person or an enterprise can be meaningful.

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Author and Address” on page 77.

Contexts

Children: “Address (address)” on page 208, “Name (person’s name)” on page 357.

Parents: “Approvers (document approvers)” on page 212, “Author” on page 215, “Maintainer (reader comment)” on page 320, “Owners” on page 373.

Examples

```
<owners><person><name>Mike Temple</name></person>
</owners>
```

Ph (Phrase)

Purpose

Use the Ph element to identify a phrase for some reason not already provided for by the IBMIDDoc language. Phrases can define containment structures to associate one element with another, such as associating a footnote with a specific sentence, or they can use an author-defined element class to further specify the semantic meaning of a phrase.

The Ph element can also be used to associate a specific property with a specific phrase. For example, you can associate a revision or version level with a phrase. You can also identify a word as a particular type of data for special processing.

You can precisely identify information that is unique to your document by defining element classes with the ClassDef element and using those classes with the Ph element. For example, in the documentation for a program that supports mining operations, you may need to discuss different kinds of rocks and want to precisely identify references to different rocks to enhance the retrievability of your information. You can define element classes for the different rock types and use Ph elements with those classes to identify references to the types of rock.

Migration Note

Examples

Hey there! This is **very important!** Don't go out in the *rain* **without your galoshes!**

Here's its markup:

```
<ph style="Bold Italic">Hey there!</ph>
This is <ph style="Underlined Bold">very</ph>
<ph style="Bold">important</ph>! Don't go out in the
<ph style="Italic">rain</ph> <ph style="Underlined Bold Italic">
without your galoshes</ph>!
```

Attributes

`style=phrase-style`

The style attribute values include:

- base
- **bold**
- *italic*
- ***bold italic***
- underlined
- ^{superscript}
- _{subscript}
- monospaced
- SMALLCAPS. Note that YOU need to do the uppercase conversion yourself. This is because not all languages do proper uppercase conversion of lowercase letters.
- **underlined bold**
- *underlined italic*
- ***underlined bold italic***
- UNDERLINED SMALLCAPS

See “Common Element Attributes (large set)” on page 204.

Usage

See “Highlighting” on page 35.

When migrating a Bookmaster document, Bookmaster highlight phrases are migrated to IBMIDDoc phrases.

HP0	<PH STYLE="base">
HP1	<PH STYLE="italic">
HP2	<PH STYLE="bold">
HP3	<PH STYLE="bold italic">
HP4	<PH STYLE="smallcaps">
HP5	<PH STYLE="underlined">
HP6	<PH STYLE="underlined italic">
HP7	<PH STYLE="underlined bold">
HP8	<PH STYLE="underlined bold italic">
HP9	<PH STYLE="underlined smallcaps">

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “Formula (math formula)” on page 267, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “MD (marked deletion)” on page 326, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402,

“RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term” on page 425, “TM (Trademark)” on page 433, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “Address (address)” on page 208, “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BOFNum (bill of forms number)” on page 222, “Bridge (bridge between concepts)” on page 223, “Cap (caption)” on page 225, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “CI (component item)” on page 228, “CLE (content list entry)” on page 231, “Code (message code number)” on page 232, “CompCmt (component comment)” on page 234, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “DefnHd (definition description heading)” on page 250, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “ExternalFileName” on page 262, “FileNum (file number)” on page 265, “Fn (footnote)” on page 265, “IBMBOFNum (bill of forms number)” on page 280, “IBMDocNum (IBM document number)” on page 280, “IBMFeatNum (IBM feature number)” on page 281, “IBMPartNum (IBM part number)” on page 290, “IBMPgmNum (IBM program number)” on page 290, “IdxTerm (index term)” on page 293, “ISBN (document ISBN number)” on page 295, “L (explicit link)” on page 300, “LeDesc (language element description)” on page 304, “LEN (language element name)” on page 307, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “ModLvl (modification level)” on page 346, “ModName (modular information element name)” on page 346, “MsgNum (message identifier)” on page 353, “MsgText (message text)” on page 354, “Name (person’s name)” on page 357, “NoteBody (note body)” on page 362, “OrderNum (order number)” on page 372, “OrigIBMDocNum (original IBM document number)” on page 372, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “PrtLoc (country where printed)” on page 398, “PublicID (public identifier)” on page 399, “Q (quotation phrase)” on page 402, “Release (product release identifier)” on page 406, “RetKey (retrieval key)” on page 407, “Screen (display screen)” on page 411, “Sem (semantic meaning)” on page 412, “STitle (shortened title)” on page 416, “SubTitle (descriptive subtitle)” on page 417, “SynNote (syntax note)” on page 418, “Term” on page 425, “TermHd (term heading)” on page 426, “TextAlt (text alternative)” on page 427, “Title” on page 431, “TM (Trademark)” on page 433, “Version (product version number)” on page 437, “VolId (volume identifier)” on page 438, “Warning (warning notice)” on page 439, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441.

Phone (telephone number)

Purpose

The Phone element contains a telephone number.

Phone has the EQUIP attribute that is used to specify the type of phone equipment being described.

If you want to specify a fax and a voice phone, use two Phone elements within the Address element.

Examples

`<phone equip="fax">1-800-555-1212</phone>`

Attributes

EQUIP=FAX | VOICE | VOICEFAX

Specifies the type of telephone equipment being described.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using reader’s comment form (RCF)” on page 90.

Contexts

Children: text (#pcdata).

Parents: “Address (address)” on page 208.

PK (programming keyword)

Purpose

The PK element contains a programming keyword. A keyword is a literal string that has special significance in the context in which it is being used.

Examples

`<P>Specify the user ID and password parameters when error messages indicate that you must provide security information and specifying the <PK>-n</PK> parameter does not solve the problem. See <XREF REFID="SECURITY"> for an explanation of using security parameters in the....</P>`

Attributes

OPTREQ=OPT | REQ | DEF

Indicates if the keyword is optional, required, or the default. REQ (required) is the default value for this attribute.

#PCDATA

The parameter keyword

Usage

See “Highlighting” on page 35.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318

page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441.

PNIndex (part number index)

Purpose

The PNIndex element is a specialized list element that contains an index of all parts contained in CI (component item) elements.

Examples

```

      :
      :
<PNINDEX SPEC=AUTO>
      :
      :

```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

SPEC=AUTO | MAN

This attribute has a fixed value of AUTO.

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Getting a part number index” on page 196.

Contexts

Children: “GendTitle (default title specification)” on page 272, “RetKey (retrieval key)” on page 407, “TitleBlk (title information)” on page 432.

Parents: “BackM (back matter)” on page 217.

PostalCode (postal or zip code)

Purpose

The PostalCode element contains a zip or postal code.

Examples

```
<authors>
<author><person>
<name>Fred Mertz</name>
<address>125 West Hollywood Blvd
Tinseltown, CA <postal code>90210</postal code></address>
</person></author>
</authors>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Using reader’s comment form (RCF)” on page 90.

Contexts

Children: text (#pcdata).

Parents: “Address (address)” on page 208.

Preface

Purpose

The Preface element contains introductory information about a document, such as the purpose of the document. If you wish to enter a unique title for the Preface, use the TitleBlk element to contain the Title element and the title text.

Examples

```
<FRONTM>
<PREFACE><SPECDPROLOG><GENDTITLE></SPECDPROLOG>
<P>This information is...

:
</FRONTM>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear. The levels of headings that appear in the table of contents are determined by the MAXTOC attribute of the IBMIDDoc tag.

See “Common Element Attributes (large set)” on page 204.

Usage

See “The preface” on page 87.

Contexts

Children: “DBody (division body)” on page 246, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “SpecDProlog (special section division prolog)” on page 415.

Parents: “FrontM (front matter)” on page 270.

Proc (procedure)

Purpose

The Proc element structures information that describes a procedure or task. The markup enables a wide variety of print and online presentation styles. It contains a link element (RefKey) for connecting procedure descriptions to graphics and other multimedia elements, such as animation or tutorials.

Use the procedure element to describe procedures such as user tasks. A procedure consists of three basic parts: a procedure entry, one or more steps, and a procedure exit. The procedure entry defines the entry criteria for a procedure, such as any prerequisite tasks or related tasks. Each procedure step defines the actions to take and the expected results and can contain other procedures. The procedure exit describes the expected result of performing the task and what to do next.

Examples

```
<proc id="babymap" style="BKM:(STYLE=BASE SEP=INLINE COMPACT)">
<titleblk><title>Baby Johnny is Crying</title></titleblk>
<procentry>Six-month old baby Johnny was sleeping
peacefully. Suddenly he began to cry.</procentry>
<procstep>
<proccmnd>
<desc>Check Johnny's diaper.</desc>
</proccmnd>
<decisionpnt>
<cond>Is the diaper wet?</cond>
<then><procstep><proccmnd>
<desc>Change the diaper.</desc>
</proccmnd><procexit>Johnny was uncomfortable.</procexit>
</procstep>
</then>
<else>
<desc>Continue at <xref refid="hungry">.</desc>
</else>
</decisionpnt>
</procstep><procstep id="hungry">
<decisionpnt>
<cond>Is Johnny hungry?</cond>
<then><procstep><decisionpnt>
<cond>Does Johnny have teeth?</cond>
<then><procstep><stepnotes><li>Johnny can eat solid
food.</li>
<li>Continue at <xref refid="frozstk"></li>
</stepnotes></procstep>
</then>
<else><procstep id="bottle"><proccmnd>
<desc>Warm a bottle.</desc>
</proccmnd><proccmnd>
<desc>Feed Johnny.</desc>
```

```

</proccmnd><procexit>Johnny needed a bottle.</procexit>
</procstep>
</else>
</decisionpnt></procstep>
</then>
<else><procstep><proccmnd>
<desc>Rock Johnny to sleep.</desc>
</proccmnd><procexit>Johnny was sleepy.</procexit>
</procstep>
</else>
</decisionpnt>
</procstep><procstep id="frozstk">
<proccmnd>
<desc>Thaw and broil a steak for Johnny. Include a
baked potato with butter and sour cream.</desc>
</proccmnd>
<procexit>Johnny was really hungry.</procexit>
</procstep></proc>

```

Attributes

procnum=*procedure-number*

Assigns a specific number to a procedure.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “Desc (element description)” on page 251, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “ProcStep (procedure step)” on page 392, “ProcSumm (procedure summary)” on page 393, “RetKey (retrieval key)” on page 407, “TitleBlk (title information)” on page 432.

Parents: “Appendix” on page 212, “Body (document body)” on page 222, “DBody (division body)” on page 246, “Else (other procedure path to follow)” on page 259, “LEDI (language element description item)” on page 304, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “ProcCmnd (procedure command)”, “Then (procedure action to take)” on page 430.

ProcCmnd (procedure command)

Purpose

The ProcCmnd element contains the command text for the procedure described in the procedure’s Desc element.

Use ProcCmnd to direct the user to take a specific action. More than one ProcCmnd element can be used on a ProcStep, but multiple ProcCmnd elements should be very closely related. If they are not closely related, they should be contained in separate ProcSteps.

Examples

```
<proccmnd>  
<desc>Check Johnny's diaper.</desc>  
</proccmnd>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “Desc (element description)” on page 251, “Proc (procedure)” on page 388, “ProcStep (procedure step)” on page 392.

Parents: “ProcStep (procedure step)” on page 392.

ProcEntry (procedure entry point)

Purpose

The ProcEntry element defines the entry criteria for a given procedure and describes the procedure itself. The PREREQPROCS and RELPROCS attributes reference prerequisite or related procedures.

Examples

```
<procentry>Six-month old baby Johnny was sleeping  
peacefully. Suddenly he began to cry.</procentry>
```

Attributes

PREREQPROCS

This attribute’s value references one or more prerequisite procedures. The order the procedure IDs are specified indicates the order the prerequisite procedures should be performed.

RELPROCS

The value of this attribute references a one or more related, but not prerequisite, procedures.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: text (#pcdata), “Attention (safety notice)” on page 214, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “StepRef (procedure step reference)” on page 416, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436.

Parents: “Proc (procedure)” on page 388, “ProcSummItem (procedure summary item)” on page 393.

ProcExit (procedure exit point)

Purpose

The ProcExit element describes the exit criteria for a procedure and optionally contains information about what to do next and how to recover if something went wrong.

Examples

```
<procexit>Johnny needed a bottle.</procexit>
```

Attributes

RECOVERYPROC

This attribute references a recovery procedure that describes what to do if the procedure is not completed successfully.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: text (#pcdata), “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “StepRef (procedure step reference)” on page 416, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436.

Parents: “Proc (procedure)” on page 388, “ProcStep (procedure step)” on page 392, “ProcSummItem (procedure summary item)” on page 393.

ProcIntro (procedure introduction)

Purpose

The ProcIntro element contains the introduction to a procedure.

Examples

```
<procintro>
<p>A father's quick guide to child care.</p>
</procintro>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “Annot (annotation)” on page 209, “AsmList (list of parts assemblies)” on page 214, “Attention (safety notice)” on page 214, “BibList (bibliography entry list)” on page 220, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “D (hierarchical division)” on page 242, “Danger (danger notice)” on page 243, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “FNList (footnote list)” on page 266, “GL (glossary list)” on page 272, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MarkList (marked note list)” on page 321, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Screen (display screen)” on page 411, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “UL (unordered list)” on page 436, “Xmp (example)” on page 440.

Parents: “Proc (procedure)” on page 388.

ProcStep (procedure step)

Purpose

The ProcStep element describes a single step in a procedure. A procedure is made up of one or more procedure steps. Each step contains a description of the step followed by an optional decision point specification indicating what further action to take. The default action is to proceed to the next step in the procedure. A step can also contain a StepNotes element containing notes about the step.

A procedure step description can itself contain a procedure, allowing you to nest procedures to any level desired (within the general element nesting limits imposed by IBMIDDoc).

Migration Note

Bookmaster only supports three levels of nesting. For migration purposes, nesting within procedure elements should be limited to three levels.

Examples

```
<procstep><proccmd>  
<desc>Change the diaper.</desc>  
</proccmd><procexit>Johnny was uncomfortable.</procexit>  
</procstep>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “DecisionPnt (decision point)” on page 247, “ProcCmnd (procedure command)” on page 389, “ProcExit (procedure exit point)” on page 391, “StepNotes (step notes)” on page 416, “TitleBlk (title information)” on page 432.

Parents: “Else (other procedure path to follow)” on page 259, “Proc (procedure)” on page 388, “ProcCmnd (procedure command)” on page 389, “Then (procedure action to take)” on page 430.

ProcSumm (procedure summary)

Purpose

The ProcSumm element contains procedure summary items.

Examples

```
<procsumm>  
<p>And that is how you care for a child.</p>  
</procsumm>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “ProcSummItem (procedure summary item)”.

Parents: “Proc (procedure)” on page 388.

ProcSummItem (procedure summary item)

Purpose

The ProcSummItem element specifies a procedure summary items.

Contexts

Children: “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391.

Parents: “ProcSumm (procedure summary)”.

ProdInfo (product information)

Purpose

The ProdInfo element contains the name and version number of the product that is associated with the document.

Examples

```
<PRODINFO>  
  <PRODNAME>Test Prod</PRODNAME>  
  <VERSION>2</VERSION>  
</PRODINFO>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Other prolog elements” on page 77.

Contexts

Children: “ProdName (product name)”, “Version (product version number)” on page 437.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

ProdName (product name)

Purpose

The ProdName element contains the name of the product with which the document is associated.

Examples

```
<IBMPRODINFO>  
  <PRODNAME>Test Prod</PRODNAME>  
  <VERSION>2</VERSION>  
  <REL>3</REL>  
  <MOD>1</MOD>  
</IBMPRODINFO>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Other prolog elements” on page 77.

Contexts

Children: text (#pcdata).

Parents: “IBMLibEntry (IBM document library definition)” on page 287, “IBMProdInfo (IBM product information)” on page 291, “LibEntry (document library definition)” on page 312, “ProdInfo (product information)” on page 393.

Prolog (document metainformation)

Purpose

The Prolog element contains metainformation about a document, which is information that describes the document, such as the document title, the author, and the document number. It also contains many different types of markup definitions used to define classes and properties. It contains *collectors* which contain information on reuse elsewhere in the document (using GLDefs and ObjLib).

Examples

```
<prolog>
<?xpp:lrs nopage>
<ibmbibentry><doctitle>
<library><titleblk>
<title>ID Workbench</title>
</titleblk></library>
<titleblk>
<title>IBMIDDoc User's Guide and Reference</ph></title>
</titleblk></doctitle>
<ibmdocnum>SH21-0783-09</ibmdocnum>
<externalfilename>iddugref</externalfilename>
</ibmbibentry><masterindexinfo><masterindexprefix>
IDDOC</masterindexprefix></masterindexinfo>
...
</prolog>
```

Attributes

See “Common Element Attributes (small set)” on page 205.

Usage

See “The preface” on page 87.

Contexts

Children: “Approvers (document approvers)” on page 212, “BibEntryDefs (contains bibliographic entries)” on page 219, “CopyRDefs (copyright definitions)” on page 237, “CritDates (set of critical dates)” on page 240, “GLDefs (glossary definitions)” on page 275, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMProdInfo (IBM product information)” on page 291, “IdxDefs (central index entries)” on page 292, “LDescs (link descriptions)” on page 302, “Maintainer (reader comment)” on page 320, “MasterIndexInfo (master index information)” on page 324, “ObjLib (object library)” on page 367, “Owners” on page 373, “ProdInfo (product information)” on page 393, “PropDefs (property definitions)” on page 396, “QualifDefs (qualification definitions)” on page 404, “RevDefs (revision tracking information)” on page 409.

Parents: “IBMIDDoc (IBM-specific product documentation)” on page 281.

PropDef (property set definition)

Purpose

The PropDef element defines values for properties that are common to a set of elements. Any element can refer to a PropDef element using the common PROPSRC attribute. When a set of properties applies only to a certain set of

element types, the ELETYPES attribute can be used to indicate which element types can refer to a given PropDef element.

You can also use PropDef to contain PROPS definitions for property attributes that are common to all elements, such Language and Proc. See “Defining Element Properties” on page 177 for guidance on using property definitions..

Examples

```
<propdef eletypes="xmp" style="BKM:(keep=10)">
<desc>Allow examples in BookMaster output to flow,
keep the 1st 10 lines together.</desc>
</propdef>
```

Attributes

PROPNAME=*name*

Defines the name to be referenced by the PROPSRC attribute.

ELETYPES=*element names*

Defines those element types (generic identifiers) to which this PropDef applies. Use ELETYPES when a PropDef is only meaningful for a specific set of element types, such as when the STYLE value is element-specific. When an ELETYPES value is specified, the PropDef values will apply only to elements of the specified type.

See “Common Element Attributes (small set)” on page 205.

Usage

See “Defining Element Properties” on page 177.

Contexts

Children: “Desc (element description)” on page 251.

Parents: “PropDefs (property definitions)”, “PropGroup (property group)” on page 397.

PropDefs (property definitions)

Purpose

The PropDefs element contains property definition elements. Properties apply to all elements contained within the division with which the property definitions are associated.

Examples

```
<propdefs>
<propdef eletypes="xmp" style="BKM:(keep=10)">
<desc>Allow examples in BookMaster output to flow,
keep the 1st 10 lines together.</desc>
</propdef>
...
</propdefs>
```

Attributes

See “Common Element Attributes (small set)” on page 205.

Usage

See “Chapter 19. Property and Class Definition” on page 177.

Contexts

Children: “ClassDef (element class definition)” on page 230, “LERSDef (LERS property definition)” on page 310, “MkDesc (mark description)” on page 329, “ModInfoDef (modular information property definition)” on page 340, “ModItemDef (item class definitions)” on page 341, “MsgItemDef (definition of message description items)” on page 350, “PropDef (property set definition)” on page 395, “PropDesc (property description)”, “PropGroup (property group)”.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

PropDesc (property description)

Purpose

The PropDesc element contains the definitions used in the values of the PROPS attribute. It also contains a default value that is used if no PROPS value is assigned by the author.

Attributes

PROPNAME

Contains the value that is being defined. This value can be used on the PROPS attribute.

Usage

See “Setting the properties to true or false” on page 172.

Contexts

Children: “Desc (element description)” on page 251, “Title” on page 431.

Parents: “PropDefs (property definitions)” on page 396, “PropGroup (property group)”.

Examples

```
<propdesc propname="ref" default="true">
  <desc>Include the Reference part</desc>
</propdesc>
```

PropGroup (property group)

Purpose

The PropGroup element enables you to organize elements within a PropDefs section. Because PropDef elements are used to define property values, the processing system cannot use those same properties to determine whether or not a given PropDef element should be processed. The PropGroup element provides a way to use properties to use or ignore PropDef elements.

The typical case is one where you want one set of properties for one output and a different set for another. You can use PropGroup elements with the PropDefs element to contain the different PropDef elements.

You can nest PropGroup elements in order to take advantage of property inheritance.

Examples

```
<PROLOG>
:
:
<PROPDEFS>
  <PROPDEF PROPNAME='xpert' SEC='IU0' PROPS='expert'>
    <DESC>This property definition applies to all elements in the document
  </PROPDEF>
  <PROPGROUP PROPS='display'>
    <DESC>The following property definition only applies
      when processing for online display.
    </DESC>
  </PROPGROUP>
  <PROPGROUP PROC='print'>
    <DESC>The following property definition only applies
      when processing for print.
    </DESC>
  <PROPDEF>
    :
  </PROPGROUP>
</PROPDEFS>
:
</PROLOG>
```

Attributes

See “Common Element Attributes (small set)” on page 205.

Contexts

Children: “ClassDef (element class definition)” on page 230, “Desc (element description)” on page 251, “LERSDef (LERS property definition)” on page 310, “MkDesc (mark description)” on page 329, “ModInfoDef (modular information property definition)” on page 340, “ModItemDef (item class definitions)” on page 341, “MsgItemDef (definition of message description items)” on page 350, “PropDef (property set definition)” on page 395, “PropDesc (property description)” on page 397, “PropGroup (property group)” on page 397.

Parents: “PropDefs (property definitions)” on page 396, “PropGroup (property group)” on page 397.

PrtLoc (country where printed)

Purpose

The PrtLoc element contains the name of the country where a document or library was printed.

Examples

```
<PRTLLOC>Boise, Idaho</PRTLLOC>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “An example of using BibEntry and BibEntryDefs” on page 122.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “BibEntry (bibliographic entry)” on page 218, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

PublicID (public identifier)

Purpose

The PublicID element contains the SGML public identifier for a document. It is intended that the public identifier of the document entity be used by presentation systems to locate the actual document, but specific presentation systems may define application-specific data to be specified as the system identifier of the document entity if they do not support the use of public identifiers. The public identifier can be included in the BibEntry itself as a way of keeping a document’s formal public identifier definition with the rest of its bibliographic information. This could allow, for example, the automatic generation of entity declarations for documents described by BibEntry elements.

See “Appendix B. Proposed IBM Standard for Formal Public Identifiers” on page 451 for a description of the formal public identifier standard defined by the IBM Corporation for its internal and external use.

Examples

```
<BIBENTRY ID="iddocref">
  ⋮
  <PUBLICID>+//ISBN 0-933186::IBM//DOCUMENT IBMIDDoc Reference//EN
  ⋮
</BIBENTRY>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “BibEntry (bibliographic entry)” on page 218, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

Publisher (document publisher)

Purpose

The Publisher element contains the name and address of the publisher of the document.

Examples

```
<PUBLISHER>IBM CORPORATION</PUBLISHER>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “About the prolog” on page 75.

Contexts

Children: “Address (address)” on page 208, “CorpName (corporation name)” on page 239.

Parents: “BibEntry (bibliographic entry)” on page 218, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMLibEntry (IBM document library definition)” on page 287, “LibEntry (document library definition)” on page 312.

PV (parameter variable)

Purpose

The PV element contains a parameter variable. The output style specification for a PV element provides a visual cue to the user, denoting that the content of the PV element has special meaning.

Examples

```
<P>This is a description of a  
<PV>parameter variable</PV>  
</P>
```

Attributes

OPTREQ=REQ | OPT | DEF

Indicates whether or not the variable is optional. REQ (required) is the default.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Highlighting” on page 35.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger

| (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "Desc
| (element description)" on page 251, "Entry (table entry)" on page 260, "Fn
| (footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311,
| "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on
| page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on
| page 331, "ModDesc (modular content description)" on page 343, "ModItem
| (module description item)" on page 344, "MsgText (message text)" on page 354,
| "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)"
| on page 382, "Q (quotation phrase)" on page 402, "SynNote (syntax note)" on
| page 418, "Term" on page 425, "Warning (warning notice)" on page 439, "Xmp
| (example)" on page 440, "XPh (example phrase)" on page 441.

Q (quotation phrase)

Purpose

Use the Q element to contain material excerpted from another source and that is used within the context of a paragraph or similar element. Use LQ for quotations that contain more than one paragraph. In the default presentation style, the quoted material is presented inline with the material around it. It is usually surrounded by typographical quotes as well.

One of the purposes of Q is to identify a quotation, it can refer to a BibEntry element using the BIBID attribute.

Examples

```
⋮
<P>New presidents often try to inspire the country to face new challenges
with words like:
  <Q BIBID="jfk0161">Ask not what your country can do for you,
    ask what you can do for your country</Q>
⋮
```

Attributes

BibId=*bibentry_id*

The ID of the BibEntry element that defines the source of the quotation. This is optional.

“Common Element Attributes (large set)” on page 204

Usage

See “Quotes and excerpts” on page 39.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “Formula (math formula)” on page 267, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “MD (marked deletion)” on page 326, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)”, “RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term” on page 425, “TM (Trademark)” on page 433, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “NoteBody (note body)” on page 362, “P

(paragraph)" on page 374, "Ph (Phrase)" on page 382, "Q (quotation phrase)" on page 402, "SynNote (syntax note)" on page 418, "Warning (warning notice)" on page 439.

Qualif (qualification)

Purpose

The Qualif element contains a qualification definition that is referenced by elements to which the qualification applies.

A qualification is a limitation or restriction on the application of information. For example, a qualification might be that information applies to 'OS/2 2.1 users only'. The QUALIF attribute is used to reference the Qualif element from the element that contains information of this type.

The Qualif element cannot be used to include or exclude information. It cannot be used for conditional processing. The QUALIF attribute and element have no effect on property-based retrieval.

For more information about the Qualif element, see "Other prolog elements" on page 77.

Examples

```
<qualifdefs>
  <qualif id="win99" ident="use">
    <title>Windows/99</title>
    <desc>Windows/99 information</desc>
  </qualif>
  <qualif id="os25" ident="use">
    <title>OS/2.5</title>
    <desc>OS/2.5 information</desc>
  </qualif>
</qualifdefs>
```

Attributes

ID=revision_ID

The ID of this qualification. The ID attribute is required. The ID is referred to with the QUALIF attribute from any element.

IDENT= USE | IGNORE

Indicates whether the qualification is a active.

"Common Element Attributes (small set)" on page 205

Usage

See "Qualifying information" on page 41.

Contexts

Children: "Desc (element description)" on page 251, "Title" on page 431.

Parents: "QualifDefs (qualification definitions)" on page 404.

QualifDefs (qualification definitions)

Purpose

The QualifDefs element contains a list of Qualif elements which are used in the document or division in which it is found.

Examples

```
<qualifdefs>
  <qualif id="win99" ident="use">
    <title>Windows/99</title>
    <desc>Windows/99 information</desc>
  </qualif>
  <qualif id="os25" ident="use">
    <title>OS/2.5</title>
    <desc>OS/2.5 information</desc>
  </qualif>
</qualifdefs>
```

Attributes

“Common Element Attributes (small set)” on page 205

Usage

See “Other prolog elements” on page 77.

Contexts

Children: “Qualif (qualification)” on page 403.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

RCF (reader comment form)

Purpose

The RCF element contains the elements necessary to produce a reader comment form.

In order for the standard IBM RCF to be generated, the following IBMIDDoc elements must be specified in the document prolog:

- DocTitle
- Library
- IBMDocNum
- Version
- Release
- Maintainer

When creating an RCF for hardcopy, be sure to include the <MAINTAINER> section in your profile. This has the address information, fax number, etc. to be included in the RCF. Also, you need to specify the <RCF> element in the back matter.

Examples

For the RCF to be generated, you need to specify the MAINTAINER element information in the prolog of your document:

```
<maintainer>
<corp>
<corpname>IBM Corporation</corpname>
<address>ATTN: Dept 542
3605 HWY 52 N
Rochester, MN
<postalcode>55901-9986</postalcode>
<phone equip="fax">1-800-555-1212</phone></address>
</corp>
</maintainer>
```

In the back matter, you need to include the RCF element; for example:

```
<backm>
<rcf><gendtitle></rcf>
</backm>
```

Attributes

“Common Element Attributes (small set)” on page 205

Usage

See “Using reader’s comment form (RCF)” on page 90.

Contexts

Children: “GendTitle (default title specification)” on page 272, “RetKey (retrieval key)” on page 407, “TitleBlk (title information)” on page 432.

Parents: “BackM (back matter)” on page 217, “FrontM (front matter)” on page 270.

RefKey (reference key)

Purpose

The RefKey element establishes a visual link that identifies a specific part of a graphic.

Examples

See <refkey>1</refkey> for that part.

Attributes

“Common Element Attributes (large set)” on page 204

Usage

See “Highlighting” on page 35.

Contexts

Children: text (#pcdata).

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “CompCmt

(component comment)" on page 234, "Danger (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "Desc (element description)" on page 251, "Entry (table entry)" on page 260, "Fig (figure)" on page 262, "FigSeg (figure segment)" on page 264, "Fn (footnote)" on page 265, "L (explicit link)" on page 300, "LI (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "ModDesc (modular content description)" on page 343, "ModItem (module description item)" on page 344, "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)" on page 382, "Q (quotation phrase)" on page 402, "Screen (display screen)" on page 411, "SynNote (syntax note)" on page 418, "Term" on page 425, "Warning (warning notice)" on page 439, "Xmp (example)" on page 440.

Release (product release identifier)

Purpose

The Release element contains the product release number.

Examples

```
<VERSION>1</VERSION>
<RELEASE>1</RELEASE>
<MOD>1</MOD>
```

Attributes

"Common Element Attributes (large set)" on page 204

Usage

See "Using IBMProdInfo" on page 79.

Contexts

Children: text (#pcdata), "Ph (Phrase)" on page 382.

Parents: "IBMProdInfo (IBM product information)" on page 291.

RepSep (syntax repeat separator)

Purpose

The RepSep element defines a repeat separator within a syntax definition. Repeat separators must be defined at the beginning of the syntax definition.

Examples

```
<syntax>
<repsep id="rsep0003a"></repsep>
<group repid="rsep0003a">
<var>variable</var>
</group>
</syntax>
```

Attributes

OPTREQ=OPT | REQ

Indicates whether the contained group is required or optional.

CONVAR=CONSTANT|VAR

Indicates whether the content of the element is a constant or a variable in the context where is it used.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 14. Program Syntax Definitions” on page 125.

Contexts

Children: text (#pcdata).

Parents: “SynBlk (syntax block)” on page 417, “Syntax (syntax diagram)” on page 420.

RetKey (retrieval key)

Purpose

RetKey can be used to specify subject heading retrieval aid text and graphics for those output types which support such graphics. This can also contains information to be used by an information management system. Using this key, the system could conduct queries without resorting to a full-text search of all of the information. Use the RetKey element to contain a list of meaningful terms and abbreviations that might be used as keywords during a database search.

Examples

```
<D>
<DPROLOG>
<TITLEBLK><TITLE>Configuring Your New Whantoozler
</TITLE></TITLEBLK>
<RETKEY>Whantoozler 5.0 Setup</RETKEY>
</DPROLOG>
...
```

Attributes

OBJ=*graphic_entity*

specifies the the name of the graphic to be used as a retrieval aid.

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “Appendix” on page 212, “DProlog (division prolog)” on page 256, “FigList (list of figures)” on page 263, “IBMBibEntry (IBM bibliographic entry)” on page 279, “IBMSafety (IBM safety notices)” on page 291, “Index” on page 293, “LE (language element)” on page 302, “LERS (language element reference section)” on page 308, “Mod (information module)” on page 336, “ModInfo (modular information)” on page 338, “MsgList (list of message or code descriptions)” on page 352, “PartAsm (part assembly)” on page 379, “PNIndex (part number index)” on page 386, “Proc (procedure)” on page 388, “RCF (reader comment form)” on page 404, “Safety (safety notices)” on page 411, “SpecDProlog (special section division prolog)” on page 415, “TList (list of tables)” on page 432, “TOC (table of contents)” on page 435.

Rev (revision)

Purpose

The Rev element defines a single revision for a document or division. The REV attribute is used on revised elements to refer to the applicable Rev element. Use the Desc element within Rev to describe the purpose of the revision.

Examples

```
<prolog>
...
<revdefs>
<rev id="v4r5" ident="use">
<date>9/9/99</date>
<desc>First draft for v4r5</desc>
</rev>
</revdefs>
...
</prolog>
```

Attributes

ID=*revision_ID*

The ID of this revision. The ID attribute is required. The ID is referred to with the REV attribute from any element.

IDENT= **USE** | **IGNORE**

Indicates whether the revision is an active revision. Revisions that are active (USE) will be indicated by whatever mechanism is defined in the presentation style, typically by placing a vertical bar or other character in the margin.

Revisions for which IGNORE is specified are ignored when the document is processed.

CODE=*character*

Associates a character with the revision. When no code is specified, you get a vertical bar to the left of text containing the active REV attribute. This is what you want to appear in the final edition. For internal drafts, you can use some other character to indicate the revision level. For example, a plus sign, an asterisk, or a number. Use only a single character.

REASON=*reason_text*

Contains the text explaining the reason for the revision.

See “Common Element Attributes (small set)” on page 205.

Usage

See “Using Revisions” on page 91.

Contexts

Children: “Author” on page 215, “Date” on page 244, “Desc (element description)” on page 251.

Parents: “RevDefs (revision tracking information)” on page 409.

RevDefs (revision tracking information)

Purpose

The RevDefs element contains Rev and Mark elements which define the revision history of the document or division.

Examples

```
<prolog>
...
<revdefs>
<rev id="v4r5" ident="use">
<date>9/9/99</date>
<desc>First draft for v4r5</desc>
</rev>
</revdefs>
...
</prolog>
```

Attributes

See “Common Element Attributes (small set)” on page 205.

Usage

See “Using Revisions” on page 91.

Contexts

Children: “Mark (marked note definition)” on page 320, “Rev (revision)” on page 408.

Parents: “DProlog (division prolog)” on page 256, “Prolog (document metainformation)” on page 395, “SpecDProlog (special section division prolog)” on page 415.

Row (table row)

Purpose

The Row element contains the row information for a row in a TGroup.

Examples

```
<table pgwide="0" id="tablesample">
<cap>Sample table caption</cap>
<tgroup cols="1">
<colspec colname="col1">
<tbody>
<row>
<entry colname="col1">my little</entry>
</row>
<row>
<entry colname="col1">sample table</entry>
</row>
</tbody>
</tgroup>
</table>
```

Attributes

ROWSEP 0 | 1

This attribute's value specifies that a row separator rule should be:

- displayed below each Entry element ending in a Row (1)
- not displayed at all (0)

VALIGN=TOP | MIDDLE | BOTTOM

This attribute specifies the vertical alignment of the text contained in the Entry elements.

TOP

specifies alignment of the text at the top of the Entry elements.

MIDDLE

specifies alignment of the text at the middle of the Entry elements.

BOTTOM

specifies alignment of the text at the bottom of the Entry elements (the default).

See "Common Element Attributes (large set)" on page 204.

Usage

See "Chapter 7. Creating IBMIDDoc Tables" on page 57.

Contexts

Children: "Entry (table entry)" on page 260.

Parents: "TBody (table body)" on page 424, "TFoot (table footer)" on page 427, "THead (table heading)" on page 429.

Safety (safety notices)

Purpose

Use Safety to contain or refer to any safety-related information such as cautions, warnings, and FCC notices.

Examples

```
<safety><titleblk><title>Safety Notices</title></titleblk>
<caution>Watch out for splinters.</caution>
</safety>
```

Attributes

SPEC=AUTO | MAN

Specifies whether the content of the element is generated. If SPEC=AUTO is specified, the element's content is generated. This attribute is not supported at this time.

See "Common Element Attributes (large set)" on page 204.

Contexts

Children: "Attention (safety notice)" on page 214, "Caution (caution notice)" on page 225, "Danger (danger notice)" on page 243, "DL (definition list)" on page 253, "Fig (figure)" on page 262, "GendTitle (default title specification)" on page 272, "L (explicit link)" on page 300, "MMObj (multi-media object; artwork)" on page 333, "Note" on page 362, "NoteList (ordered note list)" on page 363, "OL (ordered list)" on page 370, "P (paragraph)" on page 374, "PBlk (paragraph block)" on page 380, "RetKey (retrieval key)" on page 407, "Table" on page 423, "TitleBlk (title information)" on page 432, "UL (unordered list)" on page 436, "Warning (warning notice)" on page 439.

Parents: "FrontM (front matter)" on page 270.

Screen (display screen)

Purpose

The Screen element contains or refers to a representation of a computer screen or user interface panel (window).

Examples

Attributes

OBJ=*data_entity_name*

Refers to an SGML data entity that contains the screen representation. The data entity can be in any supported notation and may in fact be a reference to a live version of the panel if that is supported by your online presentation system.

When the OBJ attribute is specified, it is an error to specify any screen data or the Screen end tag.

NOTATION=

Defines the notation of the contained data for inline screen data, as follows:

LINESPEC

Significant record ends are preserved in the output.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Screens” on page 49.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Litdata (literal data)” on page 316, “MMObj (multi-media object; artwork)” on page 333, “Ph (Phrase)” on page 382, “RefKey (reference key)” on page 405, “Term” on page 425.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

Sem (semantic meaning)

Purpose

The Sem element defines the meaning of a class. It should describe the type of information for which the class should be used.

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “DL (definition list)” on page 253, “Fig (figure)” on page 262, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “Note” on page 362, “NoteList (ordered note list)” on page 363, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436.

Parents: “ClassDef (element class definition)” on page 230.

Sep (syntactic separator)

Purpose

The Sep element contains a separator that is to separate keywords, variables, operators, or groups.

Examples

```
<syntax>
<group>
<kwd>FRED</kwd>
<sep>,</sep>
<kwd>BARNEY</kwd>
</group>
</syntax>
```

Attributes

OPTREQ = REQ | OPT

Indicates whether or not the separator is optional or required. REQ (required) is the default. Any separator that is not optional is, by definition, required.

CONVAR=CONSTANT | VAR

Indicates whether the content of the element is a constant or a variable in the context where is it used.

LINKEND=*id*

Contains an ID reference that enables a link to an arbitrary location in the document.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 14. Program Syntax Definitions” on page 125.

Contexts

Children: text (#pcdata).

Parents: “Group” on page 277, “SynPh (syntax phrase)” on page 419.

SOA (summary of amendments)

Purpose

The SOA element contains information summarizing any changes made to the information since prior versions.

Examples

```
<soa>
<specdprolog><titleblk><title>What's new and different
</title></titleblk></specdprolog>
<dbody>
<p>Changes since the last edition include...</p>
</dbody>
</soa>
```

Attributes

TOC=toc | notoc

Specifies whether the heading should appear in the table of contents. The default is for headings to appear in the table of contents; to a certain level (such as heading level 3). This replaces the older style=hidden attribute. Use this to prevent headings from appearing in the table of contents. It cannot be used to add a heading to the table of contents that would not normally appear.

+ The levels of headings that appear in the table of contents are determined by
+ the MAXTOC attribute of the IBMIDDoc tag.

+ See “Common Element Attributes (large set)” on page 204.

+ Usage

+ See “Summary of changes” on page 87.

+ Contexts

+ Children: “DBody (division body)” on page 246, “DIntro (division introduction)” on
+ page 252, “DSum (division summary)” on page 257, “SpecDProlog (special section
+ division prolog)” on page 415.

+ Parents: “BackM (back matter)” on page 217, “FrontM (front matter)” on page 270.

| SpanSpec (span specification)

| Purpose

| In tables, this specifies how cells are to be combined.

| Examples

```
| <table frame="all" pgwide="0" id="complx1">  
| <cap>Complex table example</cap>  
| <tgroup cols="3" colsep="1" rowsep="1">  
| <colspec colname="col1" colwidth="25*">  
| <colspec colname="col2" colwidth="32*">  
| <colspec colname="col3" colwidth="38*">  
| <spansec namest="col1" nameend="col2" spanname="1to2">  
| <tbody>  
| <row>  
| <entry spanname="1to2" valign="top">Row 1, Cell 1  
| </entry>  
| <entry colname="col3" morerows="1" valign="top">Row  
| 1, Cell 2</entry>  
| </row>  
| <row>  
| <entry valign="top">Row 1, Cell 3</entry>  
| <entry valign="top">Row 1, Cell 4</entry>  
| </row>  
| </tbody>  
| </tgroup>  
| </table>
```

| Attributes

| **namest**=*starting column*

| **nameend**=*ending column*

| **spanname**=*name*

| **align**= **center** | **char** | **justify** | **left** | **right**

| How to align the content of the cell.

| **charoff**=*distance*

| Offset from left edge of cell.

| **char**

| Character to align to

colsep= 1 | 0

Draw a line to the right of the cell (1); or not (0).

rowsep = 1 | 0

Draw a line under the bottom of the cell (1); or not (0).

Usage

See “Chapter 7. Creating IBMIDDoc Tables” on page 57.

Contexts

Children: empty.

Parents: “TGroup (table group)” on page 428.

SpecDProlog (special section division prolog)

Purpose

The SpecDProlog element contains prolog information specific to a division.

Examples

```
<SPECDPROLOG>
  <GENDTITLE>
  <AUTHORS>
    <AUTHOR>
      <PERSON>
        <NAME>
          Rick Dennis
        </NAME>
      </PERSON>
    </AUTHOR>
  </AUTHORS>
</SPECDPROLOG>
```

Attributes

See “Common Element Attributes (small set)” on page 205.

Contexts

Children: “Approvers (document approvers)” on page 212, “Authors” on page 216, “BibEntryDefs (contains bibliographic entries)” on page 219, “CopyRDefs (copyright definitions)” on page 237, “CritDates (set of critical dates)” on page 240, “GendTitle (default title specification)” on page 272, “GIDefs (glossary definitions)” on page 275, “IBMProdInfo (IBM product information)” on page 291, “IdxDefs (central index entries)” on page 292, “LDescs (link descriptions)” on page 302, “Maintainer (reader comment)” on page 320, “MasterIndexInfo (master index information)” on page 324, “ObjLib (object library)” on page 367, “Owners” on page 373, “ProdInfo (product information)” on page 393, “PropDefs (property definitions)” on page 396, “QualifDefs (qualification definitions)” on page 404, “RetKey (retrieval key)” on page 407, “RevDefs (revision tracking information)” on page 409, “TitleBlk (title information)” on page 432.

Parents: “Abbrev (abbreviations)” on page 207, “Abstract (abstract)” on page 207, “Bibliog (bibliography)” on page 219, “Glossary” on page 276, “Legend” on page 306, “MasterIndex (master index)” on page 323, “Preface” on page 387, “SOA (summary of amendments)” on page 413.

StepNotes (step notes)

Purpose

Use StepNotes to contain notes about a procedure step, such as special considerations about the step. Do not use StepNotes to convey decision-making information. Rather, use DecisionPnt to identify conditions and actions that may change the path through the procedure.

Examples

```
<procstep>
<stepnotes>
<li>Johnny can eat solid food.</li>
<li>Continue at <xref refid="frozstk"></li>
</stepnotes></procstep>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “LI (list item)” on page 311, “LIBlk (list item block)” on page 314.

Parents: “ProcStep (procedure step)” on page 392.

StepRef (procedure step reference)

Purpose

The StepRef element cross-references to a step in a procedure (PROC).

Contexts

Children: empty.

Parents: “Desc (element description)” on page 251, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391.

STitle (shortened title)

Purpose

The STitle element contains a shortened title for an element or document. The ShortTitle is used in running footers by some output formatter styles.

Examples

```
<!DOCTYPE IBMIDDOC PUBLIC "-//ISBN 0-933186::IBM//DTD IBMIDDoc//EN" "IBMIDDOC.DTD">
<IBMIDDOC>
<PROLOG>
<BIBENTRYDEFS>
<BIBENTRY>
<DOCTITLE>
```

```

<LIBRARY>
<TITLEBLK>
<TITLE>APPC Application Suite
<STITLE>User's Guide</STITLE>
<SUBTITLE>Approval Draft</SUBTITLE>
</TITLEBLK>
</DOCTITLE>
</BIBENTRY>
</BIBENTRYDEFS>
</PROLOG>
<BODY>
:
</BODY>
</IBMIDDOC>

```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “TitleBlk (title information)” on page 432.

SubTitle (descriptive subtitle)

Purpose

The SubTitle element contains a descriptive subtitle that is often used on title pages to further describe the document’s subject matter.

Examples

```

<DOCTITLE>
<LIBRARY>
<TITLEBLK>
<TITLE>APPC Application Suite
<STITLE>User's Guide</STITLE>
<SUBTITLE>Approval Draft</SUBTITLE>
</TITLEBLK>
</DOCTITLE>

```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “TitleBlk (title information)” on page 432.

SynBlk (syntax block)

Purpose

The SynBlk element organizes syntax definitions into titled subdivisions. Use syntax blocks to organize the elements of a syntax definition into logical, optionally

titled groupings. For example, a single command may have several forms. Syntax blocks allow you to define all the forms in a single syntax definition.

In hardcopy, syntax blocks also automatically scale the diagram portion they contain to fit the column width.

Examples

```
<syntax>
<synblk>
<group>
<kwd>FORM</kwd>
<kwd>PROC</kwd>
</group>
<group>
<kwd>FILE</kwd>
<kwd>NAME</kwd>
</group>
</synblk>
</syntax>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 14. Program Syntax Definitions” on page 125.

Contexts

Children: “Fragment (syntax fragment)” on page 268, “FragRef (syntax fragment reference)” on page 269, “Group” on page 277, “RepSep (syntax repeat separator)” on page 406, “SynNote (syntax note)”, “Title” on page 431.

Parents: “Syntax (syntax diagram)” on page 420.

SynNote (syntax note)

Purpose

The SynNote element contains a note within a syntax definition group or fragment. Use SynNote to add notes to your syntax definition to explain aspects of the syntax that cannot be expressed in the syntax markup itself. In the default presentation, the syntax notes are associated with the syntax diagram using numeric callouts and the syntax notes are themselves collected at the end of the presented syntax diagram or syntax block.

Examples

```
<syntax>
<group>
<kwd>FRED</kwd>
<synnote>This is a rather common name.</synnote>
</group>
</syntax>
```

Attributes

refid=identifier

Refers to the ID of a corresponding SYNNOTE tag. This allows you to enter the note text once in the diagram, and reuse the note for another item in the diagram.

callout=character

This allows you to specify one character to use instead of a number for indicating the note.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Syntax Notes” on page 134.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Note” on page 362, “NoteList (ordered note list)” on page 363, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)”, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “Fragment (syntax fragment)” on page 268, “Group” on page 277, “SynBlk (syntax block)” on page 417, “Syntax (syntax diagram)” on page 420.

SynPh (syntax phrase)

Purpose

The SynPh element contains syntax definition elements and is used in the context of the information around it. Use SynPh to present syntax fragments outside the context of a complete syntax definition.

Examples

```
<synph><kwd optreq="def">Filename</kwd></synph>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Syntax Phrases” on page 135.

Contexts

Children: text (#pcdata), “Delim (syntax delimiter)” on page 250, “Kwd (syntax keyword)” on page 299, “Oper (syntax operator)” on page 371, “Sep (syntactic separator)” on page 412, “Var (syntax variable)” on page 437.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439, “Xmp (example)” on page 440.

Syntax (syntax diagram)

Purpose

The Syntax element contains the definition of the syntax of a statement in some computer language; or a command, function call, programming language statement, or other such construct. Use syntax definitions to define the rules for creating statements in some computer language, for example commands in a command language or programming language expressions. Syntax definitions can also be used to model more abstract constructs, such as the structural relationships between elements in a language.

The default presentation style is as “railroad tracks”, but their presentation is not limited to that form. A given definition can be presented in a variety of ways.

Examples

```
<syntax><title>Database Reference</title>
<repsep id="rsep0006"></repsep>
<group>
  <kwd>CREATE TABLE</kwd>
</group>
<group>
  <var>table_name</var>
</group>
<group repid="rsep0006">
  <group style="bkm:(composite)">
    <delim startend="START"></delim>
    <var>column_name</var>
  </group>
</group>
<fragref><title>Data Type</title></fragref>
```

```

<kwd optreq="OPT">NOT NULL</kwd>
<delim optreq="req" startend="END"></delim>
</group>
<fragment><title>Data Type</title>
<group choiceseq="CHOICE">
<kwd>INTEGER</kwd>
<group>
<group choiceseq="CHOICE">
<kwd>DECIMAL</kwd>
<kwd>DEC</kwd>
</group>
<group style="BKM:(COMPOSITE)">
<delim startend="START"></delim>
<var>length</var>
<sep>&ssbl;+&ssbl;</sep>
<var>colwidth</var>
<delim startend="END"></delim>
</group>
</group>
<group>
<group choiceseq="CHOICE">
<kwd>CHARACTER</kwd>
<kwd>CHAR</kwd>
</group>
<group optreq="OPT" style="BKM:(COMPOSITE)">
<delim startend="START"></delim>
<var>length</var>
<delim startend="END"></delim>
</group>
</group>
<group style="BKM:(COMPOSITE)">
<kwd>GRAPHIC</kwd>
<delim startend="START"></delim>
<var>length</var>
<delim startend="END"></delim>
</group>
</group>
</fragment>
</syntax>

```

Attributes

SYNSTYLE=SPACE | LBLBOX | BOX | RULE

Causes the figure to have a frame. The default is space — no frame.

LblBox

Causes a box to be placed around the diagram. The top line of the box has text label that is taken from the diagram's Title tag.

Box

Causes a box to be placed around the diagram.

Rule

Causes a line to be placed above and below the diagram; to visually separate it from the surrounding text.

Complang

Specifies the computer language. This is optional.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Chapter 14. Program Syntax Definitions" on page 125.

Contexts

Children: “Fragment (syntax fragment)” on page 268, “FragRef (syntax fragment reference)” on page 269, “Group” on page 277, “RepSep (syntax repeat separator)” on page 406, “SynBlk (syntax block)” on page 417, “SynNote (syntax note)” on page 418, “Title” on page 431.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “Warning (warning notice)” on page 439.

Table

Purpose

The Table element contains elements that make up a IBMIDDoc table.

Examples

```
<table frame="all" pgwide="0" id="complx1">
  <cap>Complex table example</cap>
  <tgroup cols="3" colsep="1" rowsep="1">
    <colspec colname="col1" colwidth="25*">
    <colspec colname="col2" colwidth="32*">
    <colspec colname="col3" colwidth="38*">
    <spanspec namest="col1" nameend="col2" spanname="1to2">
    <tbody>
      <row>
        <entry spanname="1to2" valign="top">Row 1, Cell 1
        </entry>
        <entry colname="col3" morerows="1" valign="top">Row
        1, Cell 2</entry>
      </row>
      <row>
        <entry valign="top">Row 1, Cell 3</entry>
        <entry valign="top">Row 1, Cell 4</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

Attributes

TOCENTRY= 0(NO) | 1(YES)

If YES, and if the Title element is included, the table title will be included in the generated TList for the document.

FRAME=TOP | BOTTOM | TOPBOT | ALL | SIDES | NONE

This attribute's value describes the frame around the table.

COLSEP=0(NO) | 1(YES)

This attribute's value specifies that the internal column rules should be:

- drawn to the right of each cell's content (1)
- not displayed at all (0)

ROWSEP 0(NO) | 1(YES)

This attribute's value specifies that the internal row rules should be:

- drawn below each Entry element that ends a row (1)
- not displayed at all (0)

ORIENT=PORT | LAND

This attribute specifies whether the orientation of the table presentation is portrait or landscape.

PGWIDE= 0 | 1 | 2

This attribute's value specifies that the table width should be:

- the full page width (1)
- column width (0)
- The width of the current textline (2). Use this to have a table inside a list item format to the indentation of that list item.

See "Common Element Attributes (large set)" on page 204.

Usage

See “Chapter 7. Creating IBMIDDoc Tables” on page 57.

Contexts

Children: “Cap (caption)” on page 225, “Desc (element description)” on page 251, “TGroup (table group)” on page 428.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BackCover (back cover)” on page 217, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “EdNotices (edition notices)” on page 259, “Fn (footnote)” on page 265, “FrontCover” on page 270, “LeDesc (language element description)” on page 304, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MsgItem (message description item)” on page 348, “NItem (notice item)” on page 359, “NoteBody (note body)” on page 362, “Notices (contains notices)” on page 364, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “Safety (safety notices)” on page 411, “Sem (semantic meaning)” on page 412, “SynNote (syntax note)” on page 418, “TextAlt (text alternative)” on page 427, “Warning (warning notice)” on page 439.

TBody (table body)

Purpose

The TBody element contains the body of a TGroup; the main part of a table..

Examples

```
<table frame="all" pgwide="0" id="complx">
  <cap>Complex table example</cap>
  <tgroup cols="3" colsep="1" rowsep="1">
    <colspec colname="col1" colwidth="25*">
    <colspec colname="col2" colwidth="32*">
    <colspec colname="col3" colwidth="38*">
    <spanspec namest="col1" nameend="col2" spanname="1to2">
    <tbody>
      <row>
        <entry spanname="1to2" valign="top">Row 1, Cell 1
        </entry>
        <entry colname="col3" morerows="1" valign="top">Row
        1, Cell 2</entry>
      </row>
      <row>
        <entry valign="top">Row 1, Cell 3</entry>
        <entry valign="top">Row 1, Cell 4</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

Attributes

VALIGN=TOP | MIDDLE | BOTTOM

This attribute specifies the vertical alignment of the text contained in the Entry elements.

TOP

specifies alignment of the text at the top of the Entry elements.

MIDDLE

specifies alignment of the text at the middle of the Entry elements.

BOTTOM

specifies alignment of the text at the bottom of the Entry elements (the default).

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: “Row (table row)” on page 409.

Parents: “TGroup (table group)” on page 428.

Term

Purpose

The Term element contains a term, usually within a glossary entry. When used outside the context of GLEntry, Term identifies a term that has been defined elsewhere.

Examples

```
<GLENTRY>
  <TERM>apple</TERM>
  <DEFN>The fruit of the apple tree.</DEFN>
</GLENTRY>
⋮
An <term>apple</term> a day keeps the doctor away.
```

Attributes

TERMDEF=defnid

Contains the ID of the correct definition for the term contained in the Term element. This definition is found in the glossary or definition list markup.

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Char (character data)” on page 227, “Dec (decimal number)” on page 247, “Formula (math formula)” on page 267, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “MMObj (multi-media object; artwork)” on page 333, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term”, “TM (Trademark)” on page 433, “XPh (example phrase)” on page 441.

Parents: “Address (address)” on page 208, “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Cap (caption)” on page 225, “Caution (caution notice)” on page 225, “CGraphic (character graphic)” on page 226, “CI (component item)” on page 228, “CLE (content list entry)” on page 231, “CompCmt (component comment)” on page 234, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “DefnHd (definition description heading)” on page 250, “Desc (element description)” on page 251, “DLEntry (definition list entry)” on page 255, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “GLEntry (glossary list entry)” on page 275, “L (explicit link)” on page 300, “LeDesc (language element description)” on page 304, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “ModName (modular information element name)” on page 346, “MsgText (message text)” on page 354, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Parm (parameter list entry)” on page 375, “Ph (Phrase)” on page 382, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “Q (quotation phrase)” on page 402, “Screen (display screen)” on page 411, “Sem (semantic meaning)” on page 412, “STitle (shortened title)” on page 416, “SubTitle (descriptive subtitle)” on page 417, “SynNote (syntax note)” on page 418, “Term” on page 425, “TermHd (term heading)”, “TextAlt (text alternative)” on page 427, “Title” on page 431, “TM (Trademark)” on page 433, “Warning (warning notice)” on page 439, “Xmp (example)” on page 440, “XPh (example phrase)” on page 441.

TermHd (term heading)

Purpose

The TermHd element contains the heading for the term portion of a definition or parameter list.

Examples

```
<DL>
  <TERMHD>Term</TERMHD>
  <DEFNHD>Definition</DEFNHD>
  <DLENTY>
    <TERM>Red Otter</TERM>
    <DEFN>The Red Otter lives in....
  </DEFN>
  </DLENTY>
  <DLENTY>
    <TERM>Blue Otter</TERM>
    <DEFN>Blue Otters inhabit the....
  </DEFN>
  </DLENTY>
</DL>

<PARML>
  <TERMHD>Parameter</TERMHD>
  <DEFNHD>Purpose</DEFNHD>
  <PARM>
    <TERM>D</TERM>
    <DEFN>The D element contains a hierarchical division.</DEFN>
  </PARM>
</PARM>
```

```
<TERM>P</TERM>
<DEFN>Contains a paragraph</DEFN>
</PARM>
</PARML>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Definition lists” on page 26.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “DL (definition list)” on page 253, “ParmL (parameter list)” on page 376.

TextAlt (text alternative)

Purpose

The TextAlt element contains a text description of a multimedia object for use in non-graphic environments.

Examples

```
<MMOBJ><OBJREF OBJ="TESTGRAF">
<TEXTALT><P>This is a description of the object referred to.
</TEXTALT>
</MMOBJ>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Including artwork in documents” on page 45.

Contexts

Children: text (#pcdata), “DL (definition list)” on page 253, “L (explicit link)” on page 300, “Note” on page 362, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “Ph (Phrase)” on page 382, “Table” on page 423, “Term” on page 425, “UL (unordered list)” on page 436.

Parents: “AreaDef (defines graphic hot spot area)” on page 213, “MMObj (multi-media object; artwork)” on page 333.

TFoot (table footer)

Purpose

The TFoot element contains the footer rows that occur after the TBody element. TFoot cannot be created within the current version of the graphical table editor. You can add the TFoot element to your table from the tag view, but the table cannot be edited with the graphical table editor without removing the TFoot

element. For this reason, using TFoot is not recommended. Use the last row of the table body to contain the table footing; typically to contain a list of table notes.

If you want to use the TFoot element, do not add it to your table markup until the rest of the table markup and content is complete.

Attributes

VALIGN=TOP | MIDDLE | BOTTOM

This attribute specifies the vertical alignment of the text contained in the Entry element.

TOP

specifies alignment of the text at the top of the Entry.

MIDDLE

specifies alignment of the text at the middle of the Entry.

BOTTOM

specifies alignment of the text at the bottom of the Entry (the default).

Usage

See “Adding footnotes to a table” on page 68.

Contexts

Children: “Row (table row)” on page 409.

Parents: “TGroup (table group)”.

TGroup (table group)

Purpose

The TGroup element contains elements that make up a section of a table.

Examples

```
<TABLE FRAME="ALL">
  <TGROUP COLS="5" COLSEP="1" ROWSEP="1" ORIENT="PORT" PGWIDE="0">
    :
  </TGROUP>
</TABLE>
```

Attributes

COLS=number_of_cols

This value indicates the number of columns defined for the TGroup.

COLSEP=0(NO) | 1(YES)

This attribute’s value specifies that the internal column rules should be:

- drawn to the right of each cell’s content (1)
- not displayed at all (0)

ROWSEP 0(NO) | 1(YES)

This attribute’s value specifies that the internal row rules should be:

- drawn below each Entry element that ends a row (1)
- not displayed at all (0)

ALIGN=LEFT | RIGHT | CENTER | JUSTIFY | CHAR

This attribute specifies the horizontal positioning of the text:

LEFT

specifies left alignment (the default)

RIGHT

specifies right alignment

CENTER

specifies center alignment

JUSTIFY

specifies justification of the text

CHAR

specifies alignment on a particular character

PGWIDE= 0 | 1 | 2

This attribute's value specifies that the TGroup width should be:

- the full page width (1)
- column width (0)
- text-line width (2)

ColSpec

Contains the column specification for a column.

SpanSpec

Contains the specification for a table span.

THead

Contains the table header.

TFoot

Contains the table footer.

Tbody

Contains the body of a TGroup in a Table.

Usage

See "Chapter 7. Creating IBMIDDoc Tables" on page 57.

Contexts

Children: "ColSpec (column specification)" on page 232, "SpanSpec (span specification)" on page 414, "TBody (table body)" on page 424, "TFoot (table footer)" on page 427, "THead (table heading)".

Parents: "Table" on page 423.

THead (table heading)

Purpose

The THead element contains the heading rows of a TGroup element.

Examples

```
<table frame="all" pgwide="0">
<cap>Another sample table</cap>
<tgroup cols="4" colsep="1" rowsep="1">
<colspec colname="col1" colwidth="1*">
```

```

<colspec colname="col2" colwidth="2*">
<colspec colname="col3" colwidth="3*">
<colspec colname="col4" colwidth="1*">
<thead>
<row>
<entry valign="top" rowsep="1">Col #1</entry>
<entry valign="top" rowsep="1">Col #2</entry>
<entry valign="top" rowsep="1">Col #3</entry>
<entry valign="top" rowsep="1">Col #4</entry>
</row>
</thead>
<tbody>
<row>
<entry valign="top">Row 1, Cell 1</entry>
<entry valign="top"><ol>
<li>Row 1</li>
<li>Cell 2</li>
</ol></entry>
<entry valign="top">Row 1, Cell 3; here's a little
more text than the other cells have</entry>
<entry valign="top">Row 1, Cell 4</entry>
</row>
<row>
<entry valign="top">Row 2, Cell 1</entry>
<entry valign="top">Row 2, Cell 2</entry>
<entry valign="top"><ph style="italic">Row 2, Cell
3</ph></entry>
<entry valign="top">Row 2, Cell 4</entry>
</row>
</tbody>
</tgroup>
</table>

```

Attributes

VALIGN=TOP | MIDDLE | BOTTOM

This attribute specifies the vertical alignment of the text contained in the Entry element.

TOP

specifies alignment of the text at the top of the Entry.

MIDDLE

specifies alignment of the text at the middle of the Entry.

BOTTOM

specifies alignment of the text at the bottom of the Entry (the default).

Usage

See “A Simple Table with a Table Header and IBMIDDoc Elements” on page 66.

Contexts

Children: “Row (table row)” on page 409.

Parents: “TGroup (table group)” on page 428.

Then (procedure action to take)

Purpose

The Then element contains a description of the action to take as the result of a condition that occurs at a decision point of a procedure.

Examples

```
<decisionpnt>
<cond>Is the diaper wet?</cond>
<then><procstep><proccmd>
<desc>Change the diaper.</desc>
</proccmd><procexit>Johnny was uncomfortable.</procexit>
</procstep>
</then>
<else>
<desc>Continue at <xref refid="hungry">.</desc>
</else>
</decisionpnt>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 20. Creating maintenance analysis procedures” on page 183.

Contexts

Children: “Desc (element description)” on page 251, “Proc (procedure)” on page 388, “ProcStep (procedure step)” on page 392.

Parents: “DecisionPnt (decision point)” on page 247.

Title

Purpose

The Title element contains a title for elements that can have titles. Note that the meaning of the Title element is dependent upon the context in which it is used. For example, when used within a Division, Title contains the chapter or topic heading. When used in a PBlk element, it contains the title for the block of paragraphs.

Examples

```
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
</dbody></d>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating divisions (D element)” on page 18.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Ph (Phrase)” on page 382, “Term” on page 425, “TM (Trademark)” on page 433.

Parents: “Annot (annotation)” on page 209, “Author” on page 215, “Bridge (bridge between concepts)” on page 223, “ClassDef (element class definition)” on page 230,

“DLBlk (definition list block)” on page 254, “EdNotices (edition notices)” on page 259, “Fragment (syntax fragment)” on page 268, “FragRef (syntax fragment reference)” on page 269, “GLBlk (glossary list block)” on page 274, “Group” on page 277, “LIBlk (list item block)” on page 314, “ModInfo (modular information)” on page 338, “ModItemDef (item class definitions)” on page 341, “Msg (message or code description)” on page 348, “MsgItemDef (definition of message description items)” on page 350, “Note” on page 362, “NoteList (ordered note list)” on page 363, “ParmBlk (parameter list block)” on page 376, “PartAsm (part assembly)” on page 379, “PBlk (paragraph block)” on page 380, “PropDesc (property description)” on page 397, “Qualif (qualification)” on page 403, “SynBlk (syntax block)” on page 417, “Syntax (syntax diagram)” on page 420, “TitleBlk (title information)”.

TitleBlk (title information)

Purpose

The TitleBlk element contains title information.

Examples

```
<d>
<dprolog><titleblk>
<title>My Little Chapter</title>
</titleblk></dprolog>
<dbody>
<p>Here's the beginning of my chapter.</p>
</dbody></d>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Creating divisions (D element)” on page 18.

Contexts

Children: “STitle (shortened title)” on page 416, “SubTitle (descriptive subtitle)” on page 417, “Title” on page 431.

Parents: “DIntro (division introduction)” on page 252, “DocTitle (document title)” on page 256, “DProlog (division prolog)” on page 256, “DSum (division summary)” on page 257, “FigList (list of figures)” on page 263, “IBMSafety (IBM safety notices)” on page 291, “Index” on page 293, “Library” on page 314, “PNIndex (part number index)” on page 386, “Proc (procedure)” on page 388, “ProcStep (procedure step)” on page 392, “RCF (reader comment form)” on page 404, “Safety (safety notices)” on page 411, “SpecDProlog (special section division prolog)” on page 415, “TList (list of tables)”, “TOC (table of contents)” on page 435.

TList (list of tables)

Purpose

The TList element either causes a table list to be generated, or contains an explicit list of tables to be presented.

Examples

```
<tlist><gendtitle></tlist>
```

Attributes

SPEC=**AUTO** | **MAN**

Specifies whether the content of the element is generated. If SPEC=AUTO is specified, the list is automatically generated.

LAYOUT=**Default-Layout** | **OneCol** | **TwoCol** | **ThreeCol**

Specifies the column-style for this portion of the book.

OneCol

The entries format across the entire page.

TwoCol

The entries format in two columns.

ThreeCol

The entries format in three columns.

See “Common Element Attributes (large set)” on page 204.

Usage

See “List of tables” on page 87.

Contexts

Children: “CLE (content list entry)” on page 231, “GendTitle (default title specification)” on page 272, “RetKey (retrieval key)” on page 407, “TitleBlk (title information)” on page 432.

Parents: “FrontM (front matter)” on page 270.

TM (Trademark)

Purpose

The TM tag identifies the trademark terms in the document.

The TM tag identifies the trademark terms in your source by surrounding the trademark term or phrase. This tag has attributes that are not translated; they contain no “MRI”. The TM attributes contain information for the author and the processing formatters. The TMType attribute creates the appropriate trademarking character after the term or phrase. The files IDDIRTM.LST or IDTMSCAN.LST list the trademarks and the attributes needed for the TM tag. Use these files to insert the TM tag with proper attributes for the trademark term or phrase.

Examples

```
<tm trademark="OS/2" tmowner="IBM Corporation" tmtype="reg"
tmclass="ibm">OS/2</tm> requires a <TM trademark="Pentium"
tmowner="Intel Corporation" tmtype="reg" tmclass="special">Pentium</tm>
166MHz processor.
```

Attributes

trademark

This is the trademark term or phrase repeated in the tag. This attribute is required.

tmowner

This identifies the trademark owner. For example, the trademark owner could be "IBM Corporation". This attribute is optional.

tmtype

This identifies the trademark type. One of the following must be chosen:

TM

TrademarkTM.

REG

Registered trademark[®].

SERVICE

Service marksSM.

tmclass

This identifies the trademark classification. One of the following must be chosen:

IBM

IBM Corporation.

IBMSUB

IBM subsidiary (such as TivoliTM or Lotus).

SPECIAL

Requires special notation. These are for companies who have a special agreement with IBM. There is a legal obligation for IBM to mark these trademarks in the document as well as in the Notices section.

OTHER

All other trademarks. These are not marked in the output.

Usage

See the *ID Workbench Getting Started and User's Guide* for the tools to insert trademarks into your document.

Contexts

Children: text (#pcdata), "Ph (Phrase)" on page 382, "Term" on page 425.

Parents: "Address (address)" on page 208, "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on page 214, "Bridge (bridge between concepts)" on page 223, "Cap (caption)" on page 225, "Caution (caution notice)" on page 225, "CI (component item)" on page 228, "CLE (content list entry)" on page 231, "CompCmt (component comment)" on page 234, "Cond (procedure result)" on page 235, "CopyR (copyrights)" on page 237, "Danger (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "DefnHd (definition description heading)" on page 250, "Desc (element description)" on page 251, "Entry (table entry)" on page 260, "Fn (footnote)" on page 265, "L (explicit link)" on page 300, "LeDesc (language element description)" on page 304, "LEN (language element name)" on page 307, "LI (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326, "MkNote (marked note)" on page 331, "ModDesc (modular

content description)" on page 343, "ModItem (module description item)" on page 344, "ModName (modular information element name)" on page 346, "MsgText (message text)" on page 354, "NoteBody (note body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)" on page 382, "ProcEntry (procedure entry point)" on page 390, "ProcExit (procedure exit point)" on page 391, "Q (quotation phrase)" on page 402, "Sem (semantic meaning)" on page 412, "STitle (shortened title)" on page 416, "SubTitle (descriptive subtitle)" on page 417, "SynNote (syntax note)" on page 418, "Term" on page 425, "TermHd (term heading)" on page 426, "Title" on page 431, "Warning (warning notice)" on page 439.

TOC (table of contents)

Purpose

The TOC element either causes a table of contents to be generated, or contains an explicit list of CLE elements to be presented.

Examples

```
<toc><gendtitle></toc>
```

Attributes

SPEC=**AUTO** | **MAN**

Specifies whether the content of the element is generated. If SPEC=AUTO is specified, then the element's content is generated.

LAYOUT=**Default-Layout** | **OneCol** | **TwoCol** | **ThreeCol**

Specifies the column-style for this portion of the book.

OneCol

The entries format across the entire page.

TwoCol

The entries format in two columns.

ThreeCol

The entries format in three columns.

See "Common Element Attributes (large set)" on page 204.

Usage

See "Table of contents" on page 86.

Contexts

Children: "CLE (content list entry)" on page 231, "GendTitle (default title specification)" on page 272, "RetKey (retrieval key)" on page 407, "TitleBlk (title information)" on page 432.

Parents: "DIntro (division introduction)" on page 252, "FrontM (front matter)" on page 270.

UL (unordered list)

Purpose

The UL element contains a list of items whose order of appearance is not important.

Migration Note

The simple list element from BookMaster has been included into the unordered list element; the only difference is the type of bullet used for the two lists.

Examples

```
<ul>
<li>This is an item in an unordered list. To separate
it from other items in the list, the formatter puts
a bullet beside it.</li>
<li>The paragraph that is contained in the LI element
is part of the list item which contains it. <p>This
is the contained paragraph.</p></li>
<li>This is a separate list item in our unordered
list.</li>
</ul>
```

Attributes

ULTYPE= checkoff | normal

Specifies the type of the list. Checkoff lists have an underscore before the bullet.

style= simple

Specifies a simple list; no bullet is produced.

LINESPACE=SPACE | COMPACT

Specifies whether the items in the list should be compacted or have space between the items. Nested lists automatically inherit the setting of the outer list, but can override that default with their own setting.

Usage

See “Unordered lists” on page 23.

Contexts

Children: “Bridge (bridge between concepts)” on page 223, “LI (list item)” on page 311, “LIBlk (list item block)” on page 314.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BackCover (back cover)” on page 217, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Cond (procedure result)” on page 235, “CopyR (copyrights)” on page 237, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “EdNotices (edition notices)” on page 259, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “FrontCover” on page 270, “LeDesc (language element description)” on page 304,

“LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NItem (notice item)” on page 359, “NoteBody (note body)” on page 362, “Notices (contains notices)” on page 364, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcEntry (procedure entry point)” on page 390, “ProcExit (procedure exit point)” on page 391, “ProcIntro (procedure introduction)” on page 391, “Safety (safety notices)” on page 411, “Sem (semantic meaning)” on page 412, “SynNote (syntax note)” on page 418, “TextAlt (text alternative)” on page 427, “Warning (warning notice)” on page 439.

Var (syntax variable)

Purpose

Use Var to define variables within a syntax definition.

Examples

```
<syntax>
<group>
<kwd>LANGUAGE</kwd>
<oper>=</oper>
<var>language_name</var>
</group>
</syntax>
```

Attributes

OPTREQ=OPT | REQ | DEF

Indicates whether or not the variable is optional.

LINKEND=id

Contains an ID reference that enables a link to an arbitrary location in the document.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 14. Program Syntax Definitions” on page 125.

Contexts

Children: text (#pcdata).

Parents: “Group” on page 277, “SynPh (syntax phrase)” on page 419.

Version (product version number)

Purpose

The Version element contains the version number of the product that the document describes.

Examples

```
<ibmprodinfo>  
<prodname>ID Workbench</prodname>  
<version>Version 37</version>  
<release>Release 29</release>  
</ibmprodinfo>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “Other prolog elements” on page 77.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “IBMProdInfo (IBM product information)” on page 291, “ProdInfo (product information)” on page 393.

VNet (IBM VNet mail address)

Purpose

The VNet element contains an IBM VNet email address.

Examples

```
<maintainer>  
<corp>  
<corpname>IBM Corporation</corpname>  
<address>ATTN: Dept 542  
3605 HWY 52 N  
Rochester, MN  
<postalcode>55901-9986</postalcode><phone equip="fax">  
1-800-555-1212</phone><vnet>http://w3.rchland.ibm.com/projects/IDWB  
</vnet></address>  
</corp>  
</maintainer>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata).

Parents: “Address (address)” on page 208.

VolId (volume identifier)

Purpose

Contains the identifier for one portion of a multivolume document.

Examples

```
<doctitle>
<library><titleblk>
<title>ID Workbench</title>
</titleblk></library>
<titleblk>
<title>IBMIDDoc User's Guide <ph props="ref">and
Reference</ph></title>
</titleblk></doctitle><valid>Volume 1</valid><ibmdocnum>
SH21-0783-09</ibmdocnum><externalfilename>iddugref
</externalfilename>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata), “Ph (Phrase)” on page 382.

Parents: “IBMBibEntry (IBM bibliographic entry)” on page 279.

Warning (warning notice)

Purpose

Use Warning to contain a mandatory safety notice, consisting of one or more paragraphs or other paragraph-level elements. For non-mandatory notices, use the Attention element. See “Attention (safety notice)” on page 214 for information about the Attention element.

Examples

```
<WARNING>Do not use this blow dryer while taking a shower.
</WARNING>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See “The perils of processing: Attention, caution, and danger” on page 40.

Contexts

Children: text (#pcdata), “Address (address)” on page 208, “Annot (annotation)” on page 209, “APL (APL data)” on page 211, “Bin (binary data)” on page 221, “Bridge (bridge between concepts)” on page 223, “CGraphic (character graphic)” on page 226, “Char (character data)” on page 227, “Cit (document citation)” on page 229, “Date” on page 244, “Dec (decimal number)” on page 247, “DL (definition list)” on page 253, “Fig (figure)” on page 262, “Formula (math formula)” on page 267, “GL (glossary list)” on page 272, “Hex (hexadecimal)” on page 277, “L (explicit link)” on page 300, “Lines (text with line boundaries)” on page 315, “Litdata (literal data)” on page 316, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “MMObj (multi-media object; artwork)” on page 333, “ModInfo (modular information)” on page 338, “MV (message variable)” on page 355, “Num (number)” on page 366, “Oct (octal number)” on page 369, “OL (ordered list)” on page 370, “P (paragraph)” on page 374, “ParmL (parameter list)” on page 376, “PBlk (paragraph block)” on page 380

page 380, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Q (quotation phrase)” on page 402, “RefKey (reference key)” on page 405, “Screen (display screen)” on page 411, “SynPh (syntax phrase)” on page 419, “Syntax (syntax diagram)” on page 420, “Table” on page 423, “Term” on page 425, “TM (Trademark)” on page 433, “UL (unordered list)” on page 436, “Xmp (example)”, “XPh (example phrase)” on page 441, “XRef (cross reference)” on page 442.

Parents: “Safety (safety notices)” on page 411.

WebPage

Purpose

Use WebPage to contain a web-page address. Use this to provide a location in your book to allow others to “read more about it”. Webpage is used as part of an address. It currently does not cause any automated linking from a PDF file. When used in the document, it is printed as an address line. When used in the Maintainer section, it is printed on non-US reader comment forms.

Examples

```
<maintainer>
<corp>
<corpname>IBM Corporation</corpname>
<address>ATTN: Dept 542
3605 HWY 52 N
Rochester, MN
<postalcode>55901-9986</postalcode><phone equip="fax">
1-800-555-1212</phone>
<webpage>http://w3.rchland.ibm.com/projects/IDWB</webpage>
</address>
</corp>
</maintainer>
```

Attributes

See “Common Element Attributes (large set)” on page 204.

Contexts

Children: text (#pcdata).

Parents: “Address (address)” on page 208.

Xmp (example)

Purpose

Use Xmp to contain examples of computer input or output, such as code samples or listings. In the default style, Xmp data is presented in a monospaced typeface. Within Xmp, significant record ends are preserved and presented.

Examples

```
<XMP STYLE='BKM:(KEEP="10")'>
10 LET A = B
20 IF A GT C THEN GO 40
30 LET A = C
40 PRINT A, C
</XMP>
```

Attributes

OBJ=object-reference

Allows you to include a declared program entity.

NOTATION=LINESPEC

LINESPEC is the default value for the NOTATION attribute on Xmp.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Examples of computer output” on page 44.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “Litdata (literal data)” on page 316, “MV (message variable)” on page 355, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “RefKey (reference key)” on page 405, “SynPh (syntax phrase)” on page 419, “Term” on page 425.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “BackCover (back cover)” on page 217, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “Danger (danger notice)” on page 243, “DBody (division body)” on page 246, “Defn (definition of a term)” on page 249, “DIntro (division introduction)” on page 252, “DSum (division summary)” on page 257, “Entry (table entry)” on page 260, “Fig (figure)” on page 262, “FigSeg (figure segment)” on page 264, “Fn (footnote)” on page 265, “FrontCover” on page 270, “LEDI (language element description item)” on page 304, “LI (list item)” on page 311, “LQ (stand-alone quotation)” on page 318, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “MsgItem (message description item)” on page 348, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “PBlk (paragraph block)” on page 380, “ProcIntro (procedure introduction)” on page 391, “SynNote (syntax note)” on page 418, “Warning (warning notice)” on page 439.

XPh (example phrase)

Purpose

Use the XPh element for computer input or output phrase that occurs within text. In the default style, XPh is presented in the same typeface as is used for Xmp elements.

Examples

The system will respond with a <XPH>READY</XPH> message.

Attributes

See “Common Element Attributes (large set)” on page 204.

Usage

See Table 1 on page 36.

Contexts

Children: text (#pcdata), “L (explicit link)” on page 300, “MV (message variable)” on page 355, “Ph (Phrase)” on page 382, “PK (programming keyword)” on page 385, “PV (parameter variable)” on page 400, “Term” on page 425.

Parents: “AnnotBody (annotation body)” on page 210, “Attention (safety notice)” on page 214, “Bridge (bridge between concepts)” on page 223, “Caution (caution notice)” on page 225, “CompCmt (component comment)” on page 234, “Danger (danger notice)” on page 243, “Defn (definition of a term)” on page 249, “Desc (element description)” on page 251, “Entry (table entry)” on page 260, “Fn (footnote)” on page 265, “L (explicit link)” on page 300, “LI (list item)” on page 311, “Lines (text with line boundaries)” on page 315, “LQ (stand-alone quotation)” on page 318, “MD (marked deletion)” on page 326, “MkNote (marked note)” on page 331, “ModDesc (modular content description)” on page 343, “ModItem (module description item)” on page 344, “NoteBody (note body)” on page 362, “P (paragraph)” on page 374, “Ph (Phrase)” on page 382, “Q (quotation phrase)” on page 402, “SynNote (syntax note)” on page 418, “Term” on page 425, “Warning (warning notice)” on page 439.

XRef (cross reference)

Purpose

The XRef element defines a reference to another element and generates the reference text (or other link indicator) automatically. The element referred to can be in the same document or in another document. XRef can point to another element indirectly by referring to a NameLoc element.

XRef defines a link to another element and automatically generates the link indicator. The link indicator is normally the title of the element linked to along with some locator, such as a page or panel reference, or, when the element does not have a title, just a locator. When the element linked to has an XRefText attribute specified, the XRefText value is used as the link indicator. For example, a cross reference to a paragraph that does not have XRefText coded would generate just a page number.

The style of the generated cross reference is determined by the active document style.

You can create a cross reference indirectly by using a NameLoc element to define the target of the cross reference. For example, you may have already defined a NameLoc to something for use by the L element because it will be linked to many times. You can use this same NameLoc for cross references to the element.

When XRef references a NameLoc element which references another document using the DOCNAME attribute, a cross-document link is generated.

The name on the NameLoc's DOCNAME attribute must match the name specified on the DOCNAME attribute on an IBMBibEntry or BibEntry element. DOCNAME values must be unique for each IBMBibEntry or BibEntry element.

Note: XREFTEXT is the only IBMIDDoc attribute that can take DBCS data.

Examples

Referring directly to another element:

```
<P>See <XREF REFID="AboutXRef"> for more information.  
...  
<D ID="AboutXref">XRef Explained
```

Here is an example referring to a title using Form=Full:

```
<title id=heading>Basic use of Elements</title>  
...  
<p>The best place for more information is in  
<xref refid=heading form=full>.</p>
```

The example would read:

The best place for more information is in
"Chapter 3. Basic use of Elements" on page 20.

Here is an example referring to an ordered list using Form=Full:

```
<p>Here is a list of important items:  
  <ol>  
    <li>This is the first important item</li>  
    <li>This is the second important item</li>  
    <li id=third>This is the third important item</li>  
  </ol></p>  
<p>This information refers back to the important list item  
<xref refid=third form=full>
```

Which generates this output:

This information refers back to the important list item 3 on
page 5.

Here is an example referring to a title using Form=Text:

```
<title id=Heading>Basic Use of Elements</title>  
...  
<p>The best place for more information is in  
<xref: refid=heading form=text>.
```

The example shows:

The best place for more information is in Chapter 3. Basic Use of
Elements.

Attributes

REFID=*element_id*

The ID value of the element being linked to. The linked element may be any element that takes an ID attribute.

OBJTYPE=*target_type*

Specifies the target type of the object being referenced.

FORM=*formtype*

There are three types of cross references that can be used with the Form attribute: text, number, or location. Text elements are usually linked to titles.

The location is a page number for printed output. The number is for enumerated elements only (for example, ordered list items). Be sure to check out the examples section to see how some of the attributes are used. If you are processing the document through a transform that generates a hard copy version, you can control the form of the reference. If you are processing the document through a transform that generates an online version, the xref form attribute is currently ignored. Here are the specific choices listed under Form:

NORMAL

The active presentation style defines the output.

FULL

A full reference includes all variable information about the target element. For example, if you were referring to a chapter heading, the xref would list the chapter number, name of the chapter, and the page number where the chapter can be found.

TEXT

This lists the text of the target element (typically the title). This will list the chapter number and the name of the chapter. The page number is not listed.

NUMBER

This lists only the number of the target. If this was used in reference to a chapter title, only the chapter number would be listed. If this was used in reference to an ordered list, the xref would only list the number of the step--not the page where the list number is located.

LOCATION

The location of an element, such as its page number or panel ID. In some presentations, there may not be a meaningful locator, in which case the presentation style may do the best it can, such as presenting the title of the nearest containing element.

NUMLOC

The number of an enumerated element followed by its location.

NUMTEXT

The number of an enumerated element followed by its text.

LOCTEXT

The location of an element followed by its text.

TEXTLOC

The text of an element followed by its location.

TEXTNUM

The text of an element followed by its number.

LOCNUM

The location of an element followed by its number.

See “Common Element Attributes (large set)” on page 204.

Usage

See “Chapter 6. Cross-referencing” on page 51.

Contexts

Children: empty.

| Parents: "AnnotBody (annotation body)" on page 210, "Attention (safety notice)" on
| page 214 , "Bridge (bridge between concepts)" on page 223, "Caution (caution
| notice)" on page 225, "CompCmt (component comment)" on page 234, "Danger
| (danger notice)" on page 243, "Defn (definition of a term)" on page 249, "Desc
| (element description)" on page 251, "Entry (table entry)" on page 260, "Fn
| (footnote)" on page 265, "Group" on page 277, "L (explicit link)" on page 300, "LI
| (list item)" on page 311, "Lines (text with line boundaries)" on page 315, "LQ
| (stand-alone quotation)" on page 318, "MD (marked deletion)" on page 326,
| "MkNote (marked note)" on page 331, "ModDesc (modular content description)"
| on page 343, "ModItem (module description item)" on page 344, "NoteBody (note
| body)" on page 362, "P (paragraph)" on page 374, "Ph (Phrase)" on page 382, "Q
| (quotation phrase)" on page 402, "SynNote (syntax note)" on page 418, "Warning
| (warning notice)" on page 439.

Part 4. Appendixes

Appendix A. IBMIDDoc Supported Notations

The following table lists the notations and their identifiers.

Table 19. Notation table

Notation name	Notation identifier
SGMLDOC	PUBLIC "ISO 8879:1986//NOTATION STANDARD GENERALIZED MARKUP LANGUAGE (SGML)//EN"
SMFF	PUBLIC "+//ISBN 0-933186::IBM//NOTATION SCRIPT MATHEMATICAL FORMULA FORMATTER//EN"
PSEG	PUBLIC "+//ISBN 0-933186::IBM//NOTATION AFP PAGE SEGMENT//EN"
PSEG3820	PUBLIC "+//ISBN 0-933186::IBM//NOTATION AFP PAGE SEGMENT::3820//EN"
PSEG38PP	PUBLIC "+//ISBN 0-933186::IBM//NOTATION AFP PAGE SEGMENT::38PP//EN"
PSEG4250	PUBLIC "+//ISBN 0-933186::IBM//NOTATION AFP PAGE SEGMENT::4250//EN"
ANIMATION	PUBLIC "+//ISBN 0-933186::IBM//NOTATION GENERIC ANIMATION META-NOTATION//EN"
VIDEO	PUBLIC "+//ISBN 0-933186::IBM//NOTATION GENERIC VIDEO META-NOTATION//EN"
AUDIO	PUBLIC "+//ISBN 0-933186::IBM//NOTATION GENERIC AUDIO META-NOTATION//EN"
GRAPHICS	PUBLIC "+//ISBN 0-933186::IBM//NOTATION GENERIC GRAPHICS META-NOTATION//EN"
VECTOR	PUBLIC "+//ISBN 0-933186::IBM//NOTATION GENERIC VECTOR META-NOTATION//EN"
IMAGE	PUBLIC "+//ISBN 0-933186::IBM//NOTATION GENERIC IMAGE META-NOTATION//EN"
EPS	PUBLIC "-//ADOBE//NOTATION ENCAPULATED POSTSCRIPT//EN"
APL	PUBLIC "ISO 8485:1989//NOTATION PROGRAMMING LANGUAGES - APL//EN"
ASM	PUBLIC "+//ISBN 0-933186::IBM//NOTATION 80X86 ASSEMBLER PROGRAMMING LANGUAGE//EN"
ASSEMBLE	PUBLIC "+//ISBN 0-933186::IBM//NOTATION 370 ASSEMBLER PROGRAMMING LANGUAGE//EN"
BAT	PUBLIC "+//ISBN 0-933186::IBM//NOTATION DOS BAT PROGRAMMING LANGUAGE//EN"
C	PUBLIC "ISO/IEC 9899:1990//NOTATION PROGRAMMING LANGUAGES - C//EN"
COBOL	PUBLIC "ISO 1989:1985//NOTATION PROGRAMMING LANGUAGES - COBOL//EN"
CPP	PUBLIC "+//ISBN 0-933186::IBM//NOTATION C++ PROGRAMMING LANGUAGE//EN"
EXEC	PUBLIC "+//ISBN 0-933186::IBM//NOTATION EXEC PROGRAMMING LANGUAGE//EN"
FORTRAN	PUBLIC "ISO/IEC 1539:1991//NOTATION INFORMATION TECHNOLOGY - PROGRAMMING LANGUAGES - FORTRAN//EN"
PLI	PUBLIC "ISO 6160:1979//NOTATION PROGRAMMING LANGUAGES - PL/1//EN"

Table 19. Notation table (continued)

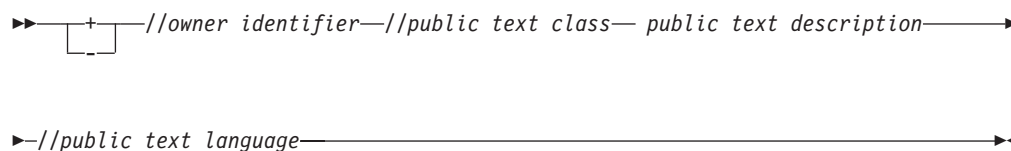
Notation name	Notation identifier
REXX	PUBLIC "+//ISBN 0-933186::IBM//NOTATION REXX PROGRAMMING LANGUAGE//EN"
CMNDLINE	PUBLIC "+//ISBN 0-933186::IBM//NOTATION COMMAND LINE ENTRY//EN"
HYQ	PUBLIC "ISO/IEC 10744:1992//NOTATION HYTIME QUERY NOTATION//EN"
HYLEX	PUBLIC "ISO/IEC 10744:1992//NOTATION HYTIME LEXICAL MODEL NOTATION//EN"
SCRIPT	PUBLIC "+//ISBN 0-933186::IBM//NOTATION DOCUMENT COMPOSITION FACILITY MARKUP//EN"
SCREEN	PUBLIC "+//ISBN 0-933186::IBM//NOTATION CHARACTER SCREEN REPRESENTATION//EN"
LINESPEC	PUBLIC "+//ISBN 0-933186::IBM//NOTATION LINE SPECIFIC CONTENT//EN"
PROGRAM	PUBLIC "+//ISBN 0-933186::IBM//NOTATION GENERIC PROGRAM LAUNCH META-NOTATION//EN"
BOOKMANAGERBOOK	PUBLIC "+//ISBN 0-933186::IBM//NOTATION BOOKMANAGER COMPILED BOOK NOTATION//EN"
IPFINF	PUBLIC "+//ISBN 0-933186::IBM//NOTATION INFORMATION PRESENTATION FACILITY BOOK NOTATION//EN"
URL	PUBLIC "-//IETF//NOTATION UNIVERSAL RESOURCE LOCATOR NOTATION:: IETF RFC 1738//EN"

Appendix B. Proposed IBM Standard for Formal Public Identifiers

One of the SGML mechanisms on which IBMIDDoc depends is Formal Public Identifiers (FPIs). This is a system-independent naming syntax that is defined in the standards ISO 8879 and ISO 9070. There are several advantages to using them, as opposed to using system identifiers directly. Resolution of references to these entities are supported using a catalog lookup feature found in most SGML products. This paper describes the conventions to be used within IBM when using these formal public identifiers.

Here is the basic format of a formal public identifier:

FPI Format



The following sections will describe each of these fields. In addition, these sections describe a format to be used within IBM information development when creating formal public identifiers. Identifiers which comply with this standard will be unique and consistent across the corporation. This may become a the standard for other parts of the corporation. Our customers may wish to use a definition like this in their own work.

Notice that when processed, FPIs are transformed or normalized using the same normalization rules as those used for tokenizing attribute value literals:

1. Ignore RS
2. Replace RE and SEPCHAR with SPACE
3. Replace a sequence of SPACE characters with a single SPACE and
4. Ignore leading and trailing SPACE characters.

This means that record ends or multiple spaces may be freely inserted anywhere a space is allowed.

Owner Identifier

The first part described here is a structure called an owner identifier. There are two forms of interest: registered (preceded with a "+") and unregistered (preceded with a "-"). The ISO 9070 standard describes a registration process and an authority. It also defines a syntax for using a registered identifier based on an ISBN number. Either the 9070 registration or the ISBN registration may be used in registered owner identifiers. Since the registration authority has not yet been named, the ISBN number syntax must be used.

IBM has an ISBN publisher prefix, 0-933186. This will be used for all formal public identifiers for IBM-owned objects. This yields a registered owner identifier of +//ISBN 0-933186::IBM.

Public Text Class and Public Text Description

Following the owner identifier, there are two fields called the public text class and public text description. The public text class defines the type or class of entity being named. The public text description gives more information about the entity described.

Here is an example of a formal public identifier which conforms to this standard.

```
+//ISBN 0-933186::IBM//DOCUMENT PUB SC31-1234-00//EN
```

The "+//ISBN ..." is the registered owner identifier, in this case indicating that the owner is IBM using the ISBN registration. Notice that **0-933186** represents the ISBN registration identifier that has been assigned.

The keyword **DOCUMENT** defines the public text class for the entity. The list of possible values for this field are defined in ISO 8879. It is also reiterated in the list below. The keyword **DOCUMENT** indicates that the entity identified is a complete SGML document.

The keyword **PUB** is the start of the public text description. The keyword **PUB** in this context indicates an IBM publication and is followed by the standard IBM publication number. The "EN" preceded by the "/" is the public text language identifier, in this case, English.

The following list defines the public text description field format. Notice that its value is keyed on the public text class. Unless otherwise noted, the data in the field identifies the entity itself; in some cases it may refer to either a containing context or another entity to which the defined information is related. As used in these definitions, the term "context" means the document library or collection to which the entity applies or which *owns* the entity.

Public Text Class

Public Text Description

CAPACITY

The following field identifies the DTD to which this capacity set entity applies.

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n]

A corporate standard will be written for all IBM internal DTDs which are distributed throughout the company. The CAPACITY file may be used to maintain the CAPACITY used by these DTDs and is referenced from SGML declaration files which apply to the DTD.

Local Definition

LCL owner descriptor [::descriptor]

CHARSET

[CP nnnnn[-n]]

where

nnnnn[-n]

indicates a standard IBM codepage, if specified. If not specified, the character set is assumed to not be an IBM-defined codepage.

Notice that if another standard is used for character encoding, such as an ANSI or ISO standard, the FPI for their standards, as they define them, should be used.

DOCUMENT

The following field applies to the document being described, not an owning context.

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n]

IBM publication

PUB zznn-nnnn[-nn]

IBM collection

LIB zznn-nnnn[-nn]

ISBN notation

ISBN n-nn-nnnnnnn-n

Local Definition

LCL owner descriptor [::descriptor]

DTD**Corporate Standard**

STD ppp n-nnnn-nnn [nnnn-n]

A corporate standard will be written for all IBM internal DTDs which are distributed throughout the company.

Local Definition

LCL owner descriptor [::descriptor]

Notice that if a standard DTD is used from another institution, like ANSI or ISO, the FPI for their standards, as they define them, should be used.

ELEMENTS

If the declared elements are intended for general use throughout IBM, the entity would be identified by the corporate standard identifier of the DTD to which the declarations are related.

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n][::descriptor]

If elements are defined for a particular document, library or other collection, the identifier identifies the owning context.

IBM publication

PUB zznn-nnnn[-nn][::descriptor]

IBM collection

LIB zznn-nnnn[-nn][::descriptor]

ISBN notation

ISBN n-nn-nnnnnnn-n[::descriptor]

Local Definition

LCL owner descriptor [::descriptor]

ENTITIES

When an entity set is distributed throughout the company, a corporate standard will be written for it and it will be identified using this field:

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n]

When an entity set is used to hold the entity declarations for a particular document, library or collection, the identifier for that context should be used to identify the entity:

IBM publication

PUB zznn-nnnn[-nn][::descriptor]

IBM collection

LIB zznn-nnnn[-nn][::descriptor]

ISBN notation

ISBN n-nn-nnnnnn-n[::descriptor]

Local Definition

LCL owner descriptor [::descriptor]

Notice that if a standard entity set is used from another institution, like American Mathematical Society or ISO, the FPI for their standards, as they define them should be used.

LPD

When a link process declaration set is distributed throughout the company, a corporate standard will be written for it and it should be identified using this field:

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n]

When an LPD is used to hold the declarations for a particular document, library or collection, the identifier for that context should be used to identify the entity:

IBM publication

PUB zznn-nnnn[-nn][::descriptor]

IBM collection

LIB zznn-nnnn[-nn][::descriptor]

ISBN notation

ISBN n-nn-nnnnnn-n[::descriptor]

Local Definition

LCL owner descriptor [::descriptor]

Note: The LINK features are not currently used within IBM but this specification is included for completeness and in anticipation of when these features will be used.

NONSGML

The following fields identify the owning context for the non-SGML entity.

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n][::descriptor]

IBM publication
 PUB zznn-nnnn[-nn][::descriptor]

IBM collection
 LIB zznn-nnnn[-nn][::descriptor]

ISBN notation
 ISBN n-nn-nnnnnn-n[::descriptor]

Local Definition
 LCL owner descriptor [::descriptor]

NOTATION

The following field identifies the document which defines the notation referred to by this notation identifier.

Corporate Standard
 STD ppp n-nnnn-nnn [nnnn-n]

Local Definition
 LCL owner descriptor [::descriptor]

Notice that if a notation is defined by a standards body, like ANSI or ISO, the FPI for their standards, as they define them should be used.

SHORTREF

When a short reference declaration set is distributed throughout the company, a corporate standard will be written for it and it should be identified using this field:

Corporate Standard
 STD ppp n-nnnn-nnn [nnnn-n]

When a short reference set is used to hold the declarations for a particular document, library or collection, the identifier for that context should be used to identify the entity:

IBM publication
 PUB zznn-nnnn[-nn]

IBM collection
 LIB zznn-nnnn[-nn]

ISBN notation
 ISBN n-nn-nnnnnn-n

Local Definition
 LCL owner descriptor [::descriptor]

Note: The short reference capabilities of SGML are not currently used by IBMIDDoc. This section is included for completeness. There are no plans to use short references with IBMIDDoc within IBM.

SUBDOC

The following field applies to the document being described, not an owning context.

Corporate Standard
 STD ppp n-nnnn-nnn [nnnn-n]

IBM publication
 PUB zznn-nnnn[-nn]

ISBN notation

ISBN n-nn-nnnnnn-n

Local Definition

LCL owner descriptor [::descriptor]

SYNTAX

The following field identifies the DTD to which this syntax definition entity applies.

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n]

A corporate standard will be written for all IBM internal DTDs which are distributed throughout the company. The SYNTAX file may be used to maintain the SYNTAX used by these DTDs and is referenced from SGML declaration files which apply to the DTD.

When an syntax definition only applies to a particular document, library or collection, the identifier for that context should be used to identify the entity:

IBM publication

PUB zznn-nnnn[-nn]

IBM collection

LIB zznn-nnnn[-nn]

ISBN notation

ISBN n-nn-nnnnnn-n

Local Definition

LCL owner descriptor [::descriptor]

TEXT

The following fields identify the owning context for the SGML text entity.

Corporate Standard

STD ppp n-nnnn-nnn [nnnn-n][::text descriptor]

IBM publication

PUB zznn-nnnn[-nn][::text descriptor]

IBM collection

LIB zznn-nnnn[-nn][::text descriptor]

ISBN notation

ISBN n-nn-nnnnnn-n[::text descriptor]

Local Definition

LCL owner descriptor [::descriptor]

Where

author name

This is a name which uniquely defines the author. It may be an RSCS address, an Internet address, an employee serial number. The key is that it must be unique across the corporation.

descriptor

is a unique identifier within the scope of the owning context that identifies that content.

owner descriptor

identifies the owner of the data. This need not be unique; it could be the name of the author, the owning department or site.

nnnnn[-n]

This is the identifier for the IBM codepage being specified.

ISBN n-nn-nnnnnn-n

This is an ISBN number.

STD ppp n-nnnn-nnn [nnnn-n]

This is the identifier defined for the specific IBM corporate standard in question.

text descriptor

is a unique identifier within the scope of the owning context. It has the following format:

descriptor[/author name[/owner descriptor]]

PUB or LIB zznn-nnnn[-nn]

This is the identifier of a document or library.

Public Text Language

The public text language field indicates the language used within the entity identified by the formal public identifier. The language codes which are valid for this field are defined in ISO 639.

Appendix C. Notices

legal stuff yet to come

Part Number Index

	Part Number	Asm- Index	Page
+	1230987	1-1	192
+	1234939	1-3	192
+	1238475	1-2	192
+	33-5234	2-1	195
+	4563423	1-1	192
+	56-2345	2-	195
+	56-3476	2-5	195
+	56-3489	2-3	195
+	56-4352	2-2	195
+	56-6534	2-4	195
+	56-8393	2-6	195
+	56-9845	2-7	195
+	56-9874	2-8	195

Index

Special Characters

_blank, linking 300
_self, linking 300
_top, linking 300

Numerics

1st-level headings 20
2nd-level headings 20

A

abbrev 88
Abbrev (abbreviations) 207
abbreviations 88
abbreviations, Abbrev 207
about this book 87
abstract 88
abstract, Abstract 207
Abstract (abstract) 207
Acrobat PDF bookmarks 75
action definition, MkAction marked
 note 327
address 77, 208
address, IBMMail, email 289
address, Internet, e-mail 294
address, Person person's name and 381
address, VNet IBM VNet mail 438
Adobe Acrobat PDF bookmarks 75
affecting how a table appears 61
alignment, table row 410
alternative, TextAlt text 427
alternatives, retrieval 174
amendments, SOA summary of 413
analysis procedures, creating
 maintenance 183
and address, Person person's name 381
Annot (annotation) 209
annot element 41
annotation, Annot 209
annotation body, AnnotBody 210
annotations 41
AnnotBody (annotation body) 210
another document, linking to 110
APL 36, 211
appears, affecting how a table 61
Appendix 212
appendix heading prefix 282
appendix titles, controlling generated 74
appendixes 88
Approvers (document approvers) 212
AreaDef (defines graphic hot spot
 area) 213
articles, HTML 282
artwork, MMObj multi-media object 333
artwork, multimedia, examples, 43
artwork, object reference 368
artwork in documents, including 45
AsmList (list of parts assemblies) 214
assemblies, list of parts 214
assembly, PartAsm part 379

assembly lists 195
assembly segment, PartAsmSeg part 380
Attention (safety notice) 214
attention notices 40
attribute, props 171
Attribute Descriptions, Element and 201
attributes 8
 ID
 on INDEX tag 105
 language 74, 284
 qualif 41
 REFID
 on IREF tag 100
 retkey 273, 308, 352
 rev 92
 toc 207, 208, 220, 242, 276, 293, 303,
 306, 323, 379, 386, 387, 413
Attributes, Common Element (large
 set) 204
Attributes, Common Element (small
 set) 205
Attributes, Table Tag 61
attributes in the CONLOC reference,
 reusing 168
author 77
Author 215
author's note 41
Authors 216

B

back cover, adding 78
back cover, BackCover 217
back cover page 85
back matter 88
back matter, BackM 217
BackCover (back cover) 217
BackM (back matter) 217
basic indexing 98
BibEntry (bibliographic entry) 218
BibEntryDefs (contains bibliographic
 entries) 219
bibliographies 89
bibliog 88
Bibliog (bibliography) 219
bibliographic entries 219
bibliographic entry, BibEntry 218
bibliographic entry, IBMIBibEntry,
 IBM 279
Bibliographies and citations 119
bibliography 88
bibliography, Bibliog 219
bibliography definitions 84
bibliography entry list, BibList 220
BibList (bibliography entry list) 220
bill of forms number, BOFNum 222
bill of forms number, IBMBOFNum 280
bin 36
Bin (binary data) 221
binary 36
binary data, Bin 221
blanks
 keeping blanks inside of 286

blanks (*continued*)
 removing blanks inside of 286
block, GLBlk, glossary list 274
block, LIBlk, list item 314
block, ParmBlk parameter list 376
block, PBlk paragraph 380
block, SynBlk syntax 417
blocking list items 32
Body (document body) 222
body, NoteBody note 362
body, ObjLibBody object library 367
body, TBody table 424
body element, using 17
bodyhd1 282
BOFNum (bill of forms number) 222
bold 35
bold-italic 35
book, multiple-language safety 287
BookManager
 DWIDTH for tables 60
Bookmarks for PDF tables of
 contents 75
books, identifying 119
books, Improving the searching of
 PDF 77
boolean properties 173
box frames 47
boxed tables 61
boxes, Labeled 40
branding 287
Bridge (bridge between concepts) 223
bridge between concepts, Bridge 223
Bridge element, using 33
bridging lists 33
broken lines 43
bulleted lists 23

C

Cap (caption) 225
caption, Cap 225
captions, table 59
catalog lists, creating parts 191
Caution (caution notice) 225
caution notice, Caution 225
caution notices 40
cells, table, combining 414
central index entries, IdxDefs 292
central indexing 102
CGraphic (character graphic) 226
cgraphics 48
change bars 91
change bars, defining 408
changes, SOA summary of 413
chapter heading prefix 282
chapter titles, controlling generated 74
char 36
Char (character data) 227
character 36
character data, Char 227
character entities 5, 155
character for a revision level 92

- character graphic, CGraphic 226
 - character graphics 48
 - characters, revision bars 408
 - characters, special 156
 - checkoff lists 25
 - choosing the proper element 19
 - CI (component item) 228
 - Cit (document citation) 229
 - CIT, using 120
 - citation link to another document 111
 - citations, Bibliographies and 119
 - citations, simple title 37
 - citations, title 120
 - class definition 177
 - Class Definition 8
 - class definition, MkClass marked note 328
 - class definitions, ModItemDef item 341
 - ClassDef (element class definition) 230
 - classes, LEDI 144
 - classification, document 282
 - classification, security 73
 - CLE (content list entry) 231
 - Code (message code number) 232
 - code, PostalCode postal or zip 387
 - code description, Msg message 348
 - code descriptions, MsgList (list of 352
 - code lists 29
 - code number, Code 232
 - collections of marked notes 93
 - ColSpec (column specification) 232
 - column separators in a table, turning off 62
 - column specification, ColSpec 232
 - column-wide figures 47
 - column-wide tables 60
 - column widths table 58
 - columns, spanning 67
 - combining table cells 414
 - command, ProcCmnd procedure 389
 - comment, Maintainer, reader 320
 - comment form, RCF reader 404
 - comment form, reader 90
 - Common Element Attributes (large set) 204
 - Common Element Attributes (small set) 205
 - compact lists 32
 - CompCmt (component comment) 234
 - CompL (component list) 234
 - component assembly lists 195
 - component comment, CompCmt 234
 - component item, CI 228
 - component list, CompL 234
 - component lists 191
 - cross-referencing 194
 - computer output, examples 44
 - concat=no for tables 65
 - concepts, IBMIDDoc 4
 - concepts, table 57
 - Cond (procedure result) 235
 - conditional processing 81
 - Conditional Processing 171
 - conditional text, translation considerations for 172
 - confidential, setting 73
 - conloc, object library 367
 - CONLOC, reusing information 166
 - CONLOC and cross-referencing 169
 - CONLOC reference, Reusing attributes in the 168
 - Considerations and Rules, Markup 9
 - considerations for conditional text, translation 172
 - ContainedDocs (documents in IBMLibEntry and LibEntry) 236
 - containment 5, 19
 - contains bibliographic entries, BibEntryDefs 219
 - Content and Style, Separation of 9
 - content description, ModDesc modular 343
 - content list entry, CLE 231
 - contents, Bookmarks for PDF tables of 75
 - contents, controlling the heading levels in the table of 86
 - contents, controlling which headings appear in the table of 87
 - contents, table of
 - controlling which headings appear 87
 - hiding headings from 87
 - contents, Table of 86
 - contents, TOC table of 435
 - contents maximum heading level, table of 285
 - continuing lists 25
 - controlling generated chapter, part, and appendix titles 74
 - controlling SGML delimiter recognition 176
 - controlling the form of cross references 55
 - controlling the heading levels in the table of contents 86
 - controlling which headings appear in the table of contents 87
 - CopyR (copyrights) 237
 - CopyRDefs (copyright definitions) 237
 - copyright 283
 - copyright, IBM 284
 - copyright, other company 78
 - copyright, setting 72
 - copyright definition 78
 - copyright definitions, CopyRDefs 237
 - copyrights, CopyR 237
 - Corp (enterprise name and address) 238
 - CorpName (corporation name) 239
 - corporation name, CorpName 239
 - country where printed, PrtLoc 398
 - cover, front 270
 - cover art or text 78
 - cover definition, CoverDef 239
 - cover page 85
 - cover spine 85
 - coverdef 78
 - CoverDef (cover definition) 239
 - Creating
 - graphic links 46
 - creating a document 17
 - creating a master index document 105
 - creating links 109
 - creating maintenance analysis procedures 183
 - creating paragraphs 18
 - creating parts catalog lists 191
 - creating simple documents 17
 - creating simple tables 58
 - CritDate (critical date for a document) 240
 - CritDates (set of critical dates) 240
 - critical date for a document, CritDate 240
 - cross-indexing 102
 - cross reference, XRef 442
 - cross references
 - controlling the form of 55
 - cross-referencing 51
 - anything 54
 - component lists 194
 - figures 52
 - list items 53
 - tables 53
 - cross-referencing, PDF 262
 - Cross-referencing items that use CONLOC 169
 - customer checkoff lists 25
 - customer setup lists 25
- ## D
- D (division) elements, using 18
 - D (hierarchical division) 242
 - Danger (danger notice) 243
 - danger notice, Danger 243
 - danger notices 40
 - data, Litdata, literal 316
 - data, literal text 45
 - date 77
 - Date 244
 - date for a document, CritDate 240
 - dates, copyright 72
 - dates, CritDates 240
 - DBCS Languages, Line Justification for 75
 - DBlk (Division block) 245
 - DBody (division body) 246
 - dec 36
 - Dec (decimal number) 247
 - decimal 36
 - decimal number, Dec 247
 - decision point, DecisionPnt 247
 - DecisionPnt (decision point) 247
 - default title specification, GendTitle 272
 - defines graphic hot spot area, AreaDef 213
 - defining glossary terms 116
 - defining rows and entrys 65
 - defining the syntax diagram 125
 - definition, MkClass marked note class 328
 - definition, IBMLibEntry, IBM document library 287
 - definition, LERSDef, LERS property 310
 - definition, LibEntry, document library 312
 - definition, Mark 320
 - definition, MkAction marked note action 327

- definition, ModInfoDef modular information property 340
- definition, PropDef property set 395
- definition, property and class 177
- definition description heading, DefnHd 250
- definition list
 - headings 27
- definition list, DL 253
- definition list block, DLBlk 254
- definition list entry, DLEntry 255
- definition lists 26
 - term-width 27
- definition of a term, Defn 249
- definition of message description items, MsgItemDef 350
- Definitions
 - Property and Class 8
- definitions, GIDefs, glossary 275
- definitions, ModItemDef item class 341
- definitions, program syntax 125
- definitions, PropDefs property 396
- definitions, QualifDefs qualification 404
- Defn (definition of a term) 249
- DefnHd (definition description heading) 250
- deletion, marking text 93
- deletion, MD marked 326
- Delim (syntax delimiter) 250
- DELIM element 131
- delimiter, Delim, syntax 250
- delimiter recognition, controlling SGML 176
- delimiter syntax element 131
- Desc (element description) 251
- description, Desc 251
- description, LeDesc, language element 304
- description, MkDesc mark 329
- description, ModDesc modular content 343
- description, Msg message or code 348
- description, PropDesc property 397
- description heading, DefnHd 250
- description item, LEDI, language element 304
- description item, ModItem module 344
- description item, MsgItem (message) 348
- description items, MsgItemDef definition of message 350
- Descriptions, Element and Attribute 201
- descriptions, LDescs, link 302
- descriptions, MsgList 352
- descriptions, tables 59
- descriptive subtitle, SubTitle 417
- diagram, defining the syntax 125
- diagram, Syntax syntax 420
- dictionary-like retrieval 273, 308, 352, 407
- dintro (division introduction) 21
- DIntro (division introduction) 252
- display screen, Screen 411
- display width for tables 60
- dividing lists 25
- division, D 242
- division (D) elements, using 18
- Division block, DBlk 245

- division body, DBody 246
- division introduction 21
- division introduction, DIntro 252
- division prolog, DProlog 256
- division prolog, SpecDProlog special section 415
- division summary, DSum 257
- divisions
 - division introduction 21
 - nesting 20
 - organizing with parts 22
 - prologs 21
- DL (definition list) 253
- DL element, using 26
- DLBlk (definition list block) 254
- DLEntry (definition list entry) 255
- DocTitle (document title) 256
- document, linking to another 110
- document approvers, Approvers 212
- document body, Body 222
- document citation, Cit 229
- document classification 282
- document ISBN number, ISBN 295
- document language 74, 284
- document library definition, IBMLibEntry 287
- document library definition, LibEntry 312
- document metainformation, Prolog 395
- document number 76
- document number, IBMDocNum 280
- document number, OrigIBMDocNum, original IBM 372
- document part, Part major 378
- document prolog 75
- document publisher, Publisher 400
- document structure 17, 71
 - prolog 102
- document style 71
- document styles 71, 283
- document title 76
- document title, DocTitle 256
- documentation, IBMIDDoc 281
- documents
 - creating 17
- documents, including artwork in 45
- documents, simple, creating 17
- documents in IBMLibEntry and LibEntry, ContainedDocs 236
- dprolog (division prolog) 21
- DProlog (division prolog) 256
- draft title page 85
- DSum (division summary) 257
- DVCF (document version control facility) 171
- DWIDTH (display width), BookManager, for tables 60

E

- e-mail address, Internet 294
- edition notices 86
- edition notices, EdNotices 259
- EdNotices (edition notices) 259
- element 102
- element, LE, language 302
- Element and Attribute Descriptions 201

- Element Attributes, Common (large set) 204
- Element Attributes, Common (small set) 205
- element class definition, ClassDef 230
- element description, Desc 251
- element description, LeDesc, language 304
- element description item, LEDI, language 304
- element name, ModName modular information 346
- element name, LEN, language 307
- element reference section, LERS, language 308
- elements
 - body, using 17
 - DELIM 131
 - division (D), using 18
 - group 128
 - IBMIDDoc 71
 - KWD 130
 - OPER 131
 - RepSep 132
 - SEP 131
 - syntax 127
 - VAR 130
 - Xref 51
- Elements, Omitted Tags and Implied 10
- elements and tags 4
- elements from an object library, reusing 166
- Else (other procedure path to follow) 259
- email address, IBMMail 289
- enabling revisions 92
- enterprise name and address, Corp 238
- entities 5
 - file 155
 - text 155
- entities, file, text, and character 155
- entities, NMList, named list of 360
- entity reference 5
- entries, IdxDefs, central index 292
- entries, index
 - basic 99
- entry, GLEntry, glossary list 275
- entry, I1, primary index 296
- entry, I2, secondary index 297
- entry, I3, tertiary index 298
- entry, IBMBibEntry, IBM bibliographic 279
- entry, Parm, parameter list 375
- Entry (table entry) 260
- entry point, ProcEntry procedure 390
- entry reference, IRef, index 294
- entrys, defining 65
- EPS graphics 45
- eServer branding 287
- example, Xmp 440
- example phrase 36
- example phrase, XPh 441
- example screens 49
- example width 44
- examples, artwork, multimedia 43
- examples of computer output 44
- excerpts, long quotes 39

exit point, ProcExit(procedure 391
Explanation, Reference 201
explicit link, L 300
external entity 5
ExternalFileName 262

F

false, setting the properties to true
or 172
FBC (folio-by-chapter) page
numbering 73
feature number, IBMFeatNum 281
Fig (figure) 262
FigList (list of figures) 263
FigSeg (figure segment) 264
figure
referencing 52
figure, Fig 262
figure list 87
figure segment, FigSeg 264
figures 47
frames 47
width 47
figures, FigList, (list of 263
file entities 5, 155
file name, external 262
file number, FileNum 265
FileNum (file number) 265
first-level headings 20
Fn (footnote) 265
FNList (footnote list) 266
folio-by-chapter page numbering 73
fonts 35
footer, TFoot table 427
footnote, Fn 265
footnote list, FNList 266
footnotes 38
footnotes in a table 68
form, RCF reader comment 404
form of cross references, controlling 55
forms number, bill of 222
forms number, IBMBOFNum, bill of 280
Formula (math formula) 267
formulas, math 50
fragment, Fragment, syntax 268
Fragment (syntax fragment) 268
fragment reference, FragRef, syntax 269
fragments, syntax 133
FragRef (syntax fragment reference) 269
frame-based articles 282
front cover, adding 78
front matter 85
front matter, FrontM 270
FrontCover 270
FrontM (front matter) 270
full window, linking 300

G

GendTitle (default title specification) 272
generated titles, controlling 74
GIF graphics 45
GL (glossary list) 272
GLBlk (glossary list block) 274
GIDefs (glossary definitions) 275
GLEntry (glossary list entry) 275

glossaries 89, 115
glossary
separation letters 116
Glossary 276
glossary, defining terms 116
glossary definitions 83
glossary definitions, GIDefs 275
glossary list, GL 272
glossary list block, GLBlk 274
glossary list entry, GLEntry 275
graphic entities 5
graphic hot spot area, defines 213
graphic links, Creating 46
graphics, character 48
green-screens 49
Group 277
group, PropGroup property 397
group, TGroup table 428
group element 128
grouping list items 32

H

H1, H2 headings 20
heading, D 242
heading, DefnHd 250
heading, TermHd term 426
heading, THead table 429
heading hierarchy 20
heading in a PDF document, linking
to 112
heading level, table of contents
maximum 285
heading levels in the table of contents,
controlling 86
heading prefix, appendix 282
heading prefix, chapter 282
heading prefix, part 285
headings, definition lists 27
headings, nesting 20
headings, using 18
headings appear in the table of contents,
controlling which 87
headings from the table of contents,
hiding 87
hex 36
Hex (hexadecimal) 277
hexadecimal 36
hexadecimal, Hex 277
hiding headings from the table of
contents 87
hierarchical division, D 242
hierarchy, heading 20
highlighting 35
highlighting, citing, noting, and
quoting 35
hot spot area, defines graphic 213
how a table appears, affecting 61
HTML articles 282
hypertext linking 109

I

I1 (primary index entry) 296
I2 (secondary index entry) 297
I3 (tertiary index entry) 298

IBM bibliographic entry,
IBMBibEntry 279
IBM copyright 284
IBM copyright, setting 72
IBM document library definition,
IBMLibEntry 287
IBM document number,
IBMDocNum 280
IBM document number,
OrigIBMDocNum original 372
IBM feature number, IBMFeatNum 281
IBM part number, IBMPartNum 290
IBM product information 79
IBM product information,
IBMProdInfo 291
IBM program number,
IBMPgmNum 290
IBM registered logo 85
IBM safety 88
IBM safety notices, IBMSafety 291
IBM-specific product documentation,
IBMIDDoc 281
IBM VNet mail address, VNet 438
IBMBibEntry (IBM bibliographic
entry) 279
IBMBOFNum (bill of forms
number) 280
IBMDocNum (IBM document
number) 280
IBMFeatNum (assigned IBM feature
number) 281
IBMIDDoc
creating documents 17
document structure 17
introduction 3
Markup Considerations and Rules 9
Markup Rules 11
terms 4
IBMIDDoc (IBM-specific product
documentation) 281
IBMIDDoc element 71
IBMLibEntry, ContainedDocs 236
IBMLibEntry (IBM document library
definition) 287
IBMMail (IBMMail email address) 289
IBMMail email address, IBMMail 289
IBMPartNum (IBM part number) 290
IBMPgmNum (IBM program
number) 290
IBMProdInfo (IBM product
information) 291
IBMSafety (IBM safety notices) 291
ID attribute
on INDEX tag 105
identifier, MsgNum message 353
identifier, PublicID, public 399
identifier, Release product release 406
identifier, VolId volume 438
Identifying attributes 8
identifying books 119
IDs or entities, NMList, named list
of 360
IdxDefs (central index entries) 292
IdxTerm (index term) 293
imbedding examples 44
Implied Elements, Omitted Tags and 10

- Improving the searching of PDF books 77
- including artwork in documents 45
- index 90
 - master, creating 105
- Index 293
- index, MasterIndex, master 323
- index, part number 89
- index, PNIndex Part number 386
- index entries
 - positioning 99
- index entries, IdxDefs 292
- index entry
 - placement 101
- index entry, I1, primary 296
- index entry, I2, secondary 297
- index entry, I3, tertiary 298
- index entry reference, IRef 294
- index information, MasterIndexInfo, master 324
- index object, MasterIndexObj master 324
- index prefix, MasterIndexPrefix master 325
- INDEX tag 105
- index term, IdxTerm 293
- indexes
 - part number 196
- indexing 97
 - basic entries 99
 - central 102
 - cross-indexing 102
 - placement in back matter 105
 - primary entries 99
 - secondary entries 99
 - see and see-also 103
 - tertiary entries 99
- indexing tags
 - placing 99
- information, conditioning 171
- information, IBMProdInfo, IBM product 291
- information, MasterIndexInfo, master index 324
- information, ModInfo modular 338
- information, modular 151
- information, ProdInfo(product 393
- information, qualifying 41
- information, RevDefs revision tracking 409
- information, TitleBlk title 432
- Information centers 282
- information element name, ModName modular 346
- information module, Mod 336
- information property definition, ModInfoDef modular 340
- internal entity 5
- Internet (internet e-mail address) 294
- internet e-mail address, Internet 294
- introduction, DIntro, (division 252
- introduction, IBMIDDoc 3
- introduction, ProcIntro procedure 391
- IPF document linking 112
- IRef (index entry reference) 294
- ISBN (document ISBN number) 295
- iSeries branding 287
- italic 35

- item , ModItem module description 344
- item, LL, list 311
- item, MsgItem (message description 348
- item, NItem notice 359
- item, ProcSummItem procedure summary 393
- item block, LIBlk, list 314
- item class definitions, ModItemDef 341
- items, MsgItemDef definition of message description 350
- items that use CONLOC, cross-referencing 169

J

- Justification for DBCS Languages, Line 75

K

- keepblanks 286
- keeping blanks in phrases 286
- keeping list items together 33
- key, RefKey reference 405
- key, RetKey retrieval 407
- keyword, Kwd, syntax 299
- keyword, PK programming 385
- keyword syntax elements 130
- Kwd (syntax keyword) 299
- KWD element 130

L

- L (explicit link) 300
- labeled boxes 40
- language 74, 284
- language element, describing 144
- language element, LE 302
- language element description, LeDesc 304
- language element description item, LEDI 304
- language element name, LEN 307
- language element reference section, LERS 308
- language reference materials 141
- languages 74, 284
- LDescs (link descriptions) 302
- LE (language element) 302
- LeDesc (language element description) 304
- LEDI (language element description item) 304
- LEDI classes 144
- legend 88
- Legend 306
- LEN (language element name) 307
- LEN, suppressing new pages 144
- LERS, compacting 144
- LERS (language element reference) 141
- LERS (language element reference section) 308
- LERS property definition, LERSDef 310
- LERSDef (LERS property definition) 310
- letter groupings, glossary 116
- level, ModLvl modification 346

- level, table of contents maximum heading 285
- levels, heading, in the table of contents, controlling 86
- LI (list item) 311
- LibEntry, ContainedDocs 236
- LibEntry (document library definition) 312
- LIBlk (list item block) 314
- LiBlk element, using 32
- libraries
 - object 7
- Library 314
- library, ObjLib object 367
- library, reusing elements from an object 166
- library body, ObjLibBody object 367
- library definition, IBMLibEntry 287
- library definition, LibEntry, document 312
- library entries 121
- Licensed material 282
- line boundaries, Lines 315
- Line Justification for DBCS Languages 75
- line-spacing, lists 32
- line-wide tables 60
- lines 43
- Lines (text with line boundaries) 315
- link, L, explicit 300
- link, MMObjLink multi-media object 335
- Link attributes 8
- link descriptions 82
- link descriptions, LDescs 302
- linking 109
 - IPF document 112
 - PDF document 111
 - headings 112
 - web document 111
- linking, cross-referencing 51
- linking to another document 110
- linking to new windows 300
- linking to web pages 82
- links
 - creating 109
- links, Creating graphic 46
- list, FNList, footnote 266
- list, GL, glossary 272
- list, MarkList, marked note 321
- list block, DLBlk, definition 254
- list block, GLBlk, glossary 274
- list block, ParmBlk parameter 376
- list entry, DLEntry, definition 255
- list entry, GLEntry, glossary 275
- list entry, Parm, parameter 375
- list item, LI 311
- list item block 32
- list item block, LIBlk 314
- list items
 - referencing 53
- list of figures, FigList 263
- list of IDs or entities, NMList named 360
- list of message or code descriptions, MsgList 352
- list of parts assemblies, AsmList 214

- list of tables, TList 432
- list subheadings, overriding the message 31
- listings, computer 44
- listings, showing 440
- lists 23
 - bridging items 33
 - checkoff 25
 - code 29
 - compacting 32
 - continuing 25
 - customer setup 25
 - definition 26
 - DL, definition 253
 - grouping items 32
 - message 29
 - NoteList, ordered note 363
 - OL ordered 370
 - ordered 24
 - parameter 28
 - ParmL parameter 376
 - separating items 33
 - simple 24
 - UL unordered 436
 - unordered 23
- lists, creating parts catalog 191
- lists and paragraphs 19
- Litdata (literal data) 316
- literal data, Litdata 316
- literal text data 45
- location, NameLoc named 357
- location, Notloc notation-specific 365
- logo, IBM registered 85
- long quotes 39
- looping, syntax diagrams 406
- LQ (stand-alone quotation) 318

M

- mail address, VNet IBM VNet 438
- Maintainer (reader comment) 320
- maintenance analysis procedures, creating 183
- major document part, Part 378
- MAPS 183
- Mark (marked note definition) 320
- mark description, MkDesc 329
- marked deletion 36
- marked deletion, MD 326
- marked note, MkNote 331
- marked note action definition, MkAction 327
- marked note class definition, MkClass 328
- marked note definition, Mark 320
- marked note list, MarkList 321
- marked notes 91
- marked notes, collections of 93
- marked sections 175
- marking text for deletion 93
- MarkList (marked note list) 321
- Markup Considerations and Rules 9
- markup declaration 5
- Markup Rules 11
- master index, MasterIndex 323
- master index document, creating a 105
- master index information, MasterIndexInfo 324
- master index object, MasterIndexObj 324
- master index prefix, MasterIndexPrefix 325
- MasterIndex (master index) 323
- MasterIndexInfo (master index information) 324
- MasterIndexObj (master index object) 324
- MasterIndexPrefix (master index prefix) 325
- material, Licensed 282
- material, Restricted 282
- materials, language reference 141
- math formula, Formula 267
- math formulas 50
- matter, FrontM, front 270
- maximum heading level, table of contents 285
- md 36
- MD (marked deletion) 326
- meaning, Sem semantic 412
- message code number, Code 232
- message description item, MsgItem 348
- message description items, MsgItemDef definition of 350
- message descriptions, MsgList (list of) 352
- message identifier, MsgNum 353
- message list subheadings, overriding the 31
- message lists 29
- message or code description, Msg 348
- message text, MsgText 354
- message variable 36
- message variable, MV 355
- metainformation, Prolog document 395
- MkAction (marked note action definition) 327
- MkClass (marked note class definition) 328
- MkDesc (mark description) 329
- MkNote (marked note) 331
- MMObj (multi-media object; artwork) 333
- MMObjLink (multi-media object link) 335
- mocha conditions 171
- Mod (information module) 336
- ModDesc (modular content description) 343
- modification level, ModLvl 346
- ModInfo (modular information) 338
- modinfo element 151
- ModInfoDef (modular information property definition) 340
- ModItem (module description item) 344
- ModItemDef (item class definitions) 341
- ModLvl (modification level) 346
- ModName (modular information element name) 346
- modular content description, ModDesc 343
- modular information 151
- modular information, ModInfo 338
- modular information element name, ModName 346

- modular information property definition, ModInfoDef 340
- module, Mod information 336
- module description item, ModItem 344
- monospaced 35
- MOREROWS, spanning of rows 67
- Msg (message or code description) 348
- MsgItem (message description item) 348
- MsgItemDef (definition of message description items) 350
- Msgl element, using 29
- MsgList (list of message or code descriptions) 352
- MsgNum (message identifier) 353
- MsgText (message text) 354
- multi-media object; artwork, MMObj 333
- multi-media object link, MMObjLink 335
- multimedia, examples, artwork 43
- multiple-language safety book 287
- multiple volume books 73
- multiple volumes 285
- mv 36
- MV (message variable) 355

N

- name, LEN, language element 307
- name, ModName modular information element 346
- name, Name person's 357
- Name (person's name) 357
- name, ProdName product 394
- name and address, Person person's 381
- named list of IDs or entities, NMList 360
- named location, NameLoc 357
- NameLoc (named location) 357
- NAMEST and NAMEEND, spanning columns 67
- nesting divisions 20
- new pages, LEN, suppressing 144
- new window, linking 300
- NItem (notice item) 359
- NMList (named list of IDs or entities) 360
- no-recycle logo 85
- non-boxed tables 61
- nopage, LERS 144
- notation-specific location, Notloc 365
- Note 362
- note, MkNote marked 331
- note, SynNote syntax 418
- note action definition, MkAction marked 327
- note body, NoteBody 362
- note class definition, MkClass marked 328
- note definition, Mark 320
- note list, MarkList, marked 321
- note list, NoteList, ordered 363
- NoteBody (note body) 362
- NoteList (ordered note list) 363
- notes
 - footnote 38
 - lists 38
 - single 37

- notes (*continued*)
 - syntax 134
- notes, marked 91
- notes, StepNotes step 416
- notes in a table 68
- notice, Warning warning 439
- notice item, NItem 359
- notices 40, 86
- Notices (contains notices) 364
- notices, edition 86
- notices, EdNotices, edition 259
- notices, IBMSafety, safety 291
- notices, Safety safety 411
- Notloc (notation-specific location) 365
- num 36
- Num (number) 366
- number, Dec 247
- number, document 76
- number, FileNum, file 265
- number, IBMBOFNum, bill of forms 280
- number, IBMDOcNum, IBM document 280
- number, IBMFeatNum, IBM feature 281
- number, IBMPartNum, part 290
- number, IBMPgmNum, IBM program 290
- number, ISBN, document 295
- number, Num 366
- number, Oct octal 369
- number, OrderNum order 372
- number, OrigIBMDocNum, original IBM document 372
- number, Phone telephone 384
- number, Version product version 437
- number index, PNIndex Part 386
- number with specified base 36
- numbered lists 24

O

- object; artwork, MMOBj
 - multi-media 333
- object, MasterIndexObj master index 324
- object libraries 7
- object library, ObjLib 367
- object library, reusing elements from
 - an 166
- object library body, ObjLibBody 367
- object link, MMOBjLink
 - multi-media 335
- object reference, ObjRef 368
- ObjLib (object library) 367
- ObjLibBody (object library body) 367
- ObjRef (object reference) 368
- oct 36
- Oct (octal number) 369
- octal 36
- octal number, Oct 369
- of message description items,
 - MsgItemDef definition 350
- OL (ordered list) 370
- OL element, using 24
- oltype=checkoff 25
- oltype=step 25
- Omitted Tags and Implied Elements 10
- Oper (syntax operator) 371
- OPER element 131
- operator, Oper syntax 371

- operator syntax element 131
- or zip code, PostalCode postal 387
- order number, OrderNum 372
- ordered list, OL 370
- ordered lists 24
- ordered note list, NoteList 363
- OrderNum (order number) 372
- OrigIBMDocNum (original IBM document number) 372
- original IBM document number,
 - OrigIBMDocNum 372
- output, examples of computer 44
- overriding the message list
 - subheadings 31
- Owners 373

P

- P (paragraph) 374
- P element, creating paragraphs 18
- page, web 440
- page numbering 73
- page prefix, MasterIndexPrefix master index 325
- page-wide figures 47
- page-wide tables 60
- paragraph, P 374
- paragraph block, PBlk 380
- paragraph-like elements 12
- paragraphs, creating 18
- paragraphs and lists 19
- parameter list, ParmL 376
- parameter list block, ParmBlk 376
- parameter list entry, Parm 375
- parameter lists 28
 - term-width 29
- parameter variable, PV 400
- Parm (parameter list entry) 375
- ParmBlk (parameter list block) 376
- ParmL (parameter list) 376
- ParmL element, using 28
- Part (major document part) 378
- part, Part major document 378
- part assembly, PartAsm 379
- part assembly segment, PartAsmSeg 380
- part heading prefix 285
- part number, IBMPartNum 290
- part number index 89
- Part number index, PNIndex 386
- part number indexes 196
- part titles, controlling generated 74
- PartAsm (part assembly) 379
- PartAsmSeg (part assembly segment) 380
- parts, organizing divisions 22
- parts assemblies, list of 214
- parts catalog lists, creating 191
- PBlk (paragraph block) 380
- PDF bookmarks 286
- PDF books, Improving the searching of 77
- PDF document, linking to 111
- PDF external file name 262
- PDF heading, linking to 112
- PDF tables of contents, Bookmarks for 75
- perils of processing 40
- Person (person's name and address) 381
- person's name, Name 357
- person's name and address, Person 381
- Ph (Phrase) 382
- Phone (telephone number) 384
- Phrase, Ph 382
- phrase, Q quotation 402
- phrase, SynPh syntax 419
- phrase, XPh example 441
- phrase-like elements 12
- phrases
 - keeping blanks inside of 286
 - removing blanks inside of 286
 - syntax 135
- phrases, conditioning 172
- pictures, including in documents 45
- PK (programming keyword) 385
- placing index tags 99
- PNIndex (Part number index) 386
- point, ProcEntry procedure entry 390
- point, ProcExit procedure exit 391
- positioning index entries 99
- postal or zip code, PostalCode 387
- PostalCode (postal or zip code) 387
- preface 87
- Preface 387
- prefix, appendix heading 282
- prefix, chapter heading 282
- prefix, MasterIndexPrefix master index 325
- prefix, part heading 285
- preformatted listings 44
- preformatted text 43
- primary index entry, I1 296
- printed, PrtLoc, country where 398
- Proc (procedure) 388
- ProcCmnd (procedure command) 389
- procedure, Proc 388
- procedure action to take, Then 430
- procedure command, ProcCmnd 389
- procedure entry point, ProcEntry 390
- procedure exit point, ProcExit 391
- procedure introduction, ProcIntro 391
- procedure result, Cond 235
- procedure step, ProcStep 392
- procedure step reference, StepRef 416
- procedure summary, ProcSumm 393
- procedure summary item,
 - ProcSummItem 393
- procedures, creating maintenance
 - analysis 183
- ProcEntry (procedure entry point) 390
- Processing, Conditional 171
- processing, perils 40
- ProcExit (procedure exit point) 391
- ProcIntro (procedure introduction) 391
- ProcStep (procedure step) 392
- ProcSumm (procedure summary) 393
- ProcSummItem (procedure summary item) 393
- ProdInfo (product information) 393
- ProdName (product name) 394
- product branding 287
- product documentation, IBMIDDoc 281
- product information, IBM 79
- product information, IBMProdInfo 291
- product information, ProdInfo 393
- product name, ProdName 394

- product release identifier, Release 406
- product version number, Version 437
- program number, IBMPgmNum 290
- program syntax definitions 125
- programming keyword, PK 385
- prolog, document 75
- Prolog (document metainformation) 395
- prolog, DProlog, division 256
- prolog, SpecDProlog special section division 415
- prologs, division 21
- PropDef (property set definition) 395
- PropDefs (property definitions) 396
- PropDesc (property description) 397
- properties, conditional text 171
- properties boolean 173
- properties to true or false, setting the 172
- Property attributes 8
- property-based retrieval 171
- property definition 177
- Property Definition 8
- property definition, LERSDef, LERS 310
- property definition, ModInfoDef modular information 340
- property definitions 79
- property definitions, PropDefs 396
- property description, PropDesc 397
- property group, PropGroup 397
- property set definition, PropDef 395
- PropGroup (property group) 397
- props, conditioning phrases 172
- props attribute 171
- PrtLoc (country where printed) 398
- pSeries branding 287
- public identifier, PublicID 399
- PublicID (public identifier) 399
- Publisher (document publisher) 400
- publisher, Publisher document 400
- PV (parameter variable) 400

Q

- Q (quotation phrase) 402
- Qualif (qualification) 403
- qualif attribute 41
- QualifDefs (qualification definitions) 404
- qualification, Qualif 403
- qualification definitions, QualifDefs 404
- qualifying information 41
- quotation, LQ, stand-alone 318
- quotation phrase, Q 402
- quotes 39

R

- RCF (reader comment form) 404
- reader comment, Maintainer 320
- reader comment form 90
- reader comment form, RCF 404
- recognition, controlling SGML delimiter 176
- recycle logo, omitting 85
- reference, FragRef, syntax fragment 269
- reference, IRef, index entry 294
- reference, ObjRef object 368
- reference, StepRef procedure step 416
- reference, XRef cross 442
- Reference Explanation 201
- reference key 36
- reference key, RefKey 405
- reference materials, language 141
- reference section, LERS, language element 308
- references
 - controlling the form of 55
- referencing
 - anything 54
 - component lists 194
 - figures 52
 - list items 53
 - reusing attributes in the CONLOC 168
 - tables 53
- referencing, cross 51
- REFID attribute
 - cross-indexing primaries description 100
 - on IREF tag 100
- refkey 36
- RefKey (reference key) 405
- registered IBM logo 85
- Release (product release identifier) 406
- release identifier, Release product 406
- removeblanks 286
- removing blanks in phrases 286
- repeat separator, RepSep syntax 406
- repeat separator syntax element 132
- RepSep (syntax repeat separator) 406
- RepSep element 132
- Restricted material 282
- RetKey (retrieval key) 407
- retkey attribute 273, 308, 352
- retrieval, dictionary-like 273, 308, 352, 407
- retrieval, property-based 171
- retrieval alternatives 174
- retrieval key, RetKey 407
- reusing attributes in the CONLOC reference 168
- reusing elements from an object library 166
- Rev (revision) 408
- REV attribute 92
- RevDefs (revision tracking information) 409
- reverse key, RefKey 405
- revision
 - character 92
- revision, Rev 408
- revision characters 408
- revision elements 91
- revision tracking information, RevDefs 409
- revisions 91
 - defining 91
 - enabling 92
 - indicating 92
- row, Row(table 409
- Row (table row) 409
- row alignment, table 410
- row separators in a table, turning off 62
- rows, defining 65
- rows, spanning 67

- Rules, Markup 11
- Rules, Markup Considerations 9
- rules frames 47
- running foot short title 18
- running head 407

S

- safety, IBM 88
- Safety (safety notices) 411
- safety book, multiple-language 287
- safety notice, Attention 214
- safety notices, IBMSafety 291
- safety notices, Safety 411
- sample code, displaying 440
- scaling examples 44
- Screen (display screen) 411
- screen, Screen display 411
- screen examples 49
- searching of PDF books, Improving the 77
- second-level headings 20
- secondary index entry, I2 297
- section division prolog, SpecDProlog special 415
- sections, marked 175
- security 284
- security classification 73
- see-also index entries 103
- segment, FigSeg, figure 264
- segment, PartAsmSeg part assembly 380
- Sem (semantic meaning) 412
- semantic meaning, Sem 412
- Sep (syntactic separator) 412
- SEP element 131
- separating list items 33
- Separation of Content and Style 9
- separator, RepSep syntax repeat 406
- separator, Sep syntactic 412
- separators, table rows 410
- seperator syntax element 131
- sequential page numbering 73
- set definition, PropDef property 395
- set of critical dates, CritDates 240
- setting the properties to true or false 172
- setup lists, customer 25
- SGML delimiter recognition, controlling 176
- short title 18
- shortened title, STitle 416
- simple lists 24
- simple title citations 37
- small caps 35
- SOA (summary of amendments) 413
- SOA (summary of changes) 87
- span specification, SpanSpec 414
- spanning rows and columns 67
- SpanSpec (span specification) 414
- SpecDProlog (special section division prolog) 415
- special characters 156
- special section division prolog, SpecDProlog 415
- specification, SpanSpec span 414
- specifying table column widths 58
- spine 85
- splitting tables between pages 61

- spot area, defines graphic hot 213
- stand-alone quotation, LQ 318
- step, ProcStep procedure 392
- step lists 25
- step notes, StepNotes 416
- step reference, StepRef procedure 416
- StepNotes (step notes) 416
- StepRef (procedure step reference) 416
- STitle 18
- STitle (shortened title) 416
- structure, document 17, 71
- style, document 71
- Style and Content, Separation of 9
- Style attributes 8
- style overrides, IBMIDDoc tag 286
- style=simple 24
- styles
 - syntax 128
- styles, document 71, 283
- subheadings, overriding the message
 - list 31
- subscripts 35
- SubTitle (descriptive subtitle) 417
- subtitle, SubTitle descriptive 417
- summary, DSum, division 257
- summary, ProcSumm procedure 393
- summary item, ProcSummItem
 - procedure 393
- summary of amendments 87
- summary of amendments) SOA 413
- summary of changes xi, 87
- superscript 35
- symbols 156
- SynBlk (syntax block) 417
- SynNote (syntax note) 418
- SynPh (syntax phrase) 419
- syntactic separator, Sep 412
- syntax
 - examples 136
 - phrases 135
- Syntax (syntax diagram) 420
- syntax block, SynBlk 417
- syntax definitions, program 125
- syntax delimiter, Delim 250
- syntax diagram, defining the 125
- syntax diagram, Syntax 420
- syntax element 127
- syntax fragment, Fragment 268
- syntax fragment reference, FragRef 269
- syntax fragments 133
- syntax keyword, Kwd 299
- syntax lists 28
- syntax note, SynNote 418
- syntax notes 134
- syntax operator, Oper 371
- syntax phrase, SynPh 419
- syntax repeat separator, RepSep 406
- syntax styles 128
- syntax variable, Var 437

T

- table
 - column specifications 64
 - referencing 53
- Table 423
- table body, TBody 424
- table cell
 - alignment 64

- table cell (*continued*)
 - column separators 64
 - row separators 65
 - width 64
- table cells, combining 414
- table entry, Entry 260
- table footer, TFoot 427
- table group, TGroup 428
- table heading, THead 429
- table list 87
- table of contents
 - controlling the heading levels in 86
 - controlling which headings appear in the 87
 - hiding headings from 87
- Table of contents 86
- table of contents, TOC 435
- table of contents maximum heading
 - level 285
- table row, Row 409
- table row alignment 410
- table row separators 410
- Table Tag Attributes 61
- tables 57
 - appearance 61
 - captions 59
 - column widths 58
 - complex example 67
 - complex header 68
 - concepts 57
 - controlling inside lines 62
 - controlling the frame 61
 - controlling width 60
 - defining rows and entries 65
 - DWIDTH (display width),
 - BookManager 60
 - examples 65
 - notes 68
 - sideways 62
 - simple, creating 58
 - simple example 65
 - simple headers 66
 - spanning rows and columns 67
 - splitting 61
 - unformatted text 65
 - width, column, page, or textline 60
- tables, TList list of 432
- tables of contents, Bookmarks for
 - PDF 75
- Tag Attributes, Table 61
- tags
 - INDEX 105
- tags and elements 4
- Tags and Implied Elements, Omitted 10
- target="_blank" 300
- target="_self" 300
- target="_top" 300
- TBody (table body) 424
- telephone number, Phone 384
- term 36
- Term 425
- term, Defn 249
- term, IdxTerm, index 293
- term heading, TermHd 426
- term lists 26
- term-width
 - definition lists 27

- term-width (*continued*)
 - parameter lists 29
- TermHd (term heading) 426
- terms, IBMIDDoc 4
- tertiary index entry, I3 298
- text, changed 91
- text, MsgText message 354
- text, translation considerations for
 - conditional 172
- text, turning on or off 171
- text alternative, TextAlt 427
- text data, literal 45
- text entities 155
- text for deletion, marking 93
- text with line boundaries, Lines 315
- TextAlt (text alternative) 427
- textline width tables 60
- TFoot (table footer) 427
- TGroup (table group) 428
- THead (table heading) 429
- Then (procedure action to take) 430
- Title 431
- title, DocTitle, document 256
- title, document 76
- title, STitle shortened 416
- title citations 120
- title citations, simple 37
- title information, TitleBlk 432
- title specification, GendTitle, default 272
- TitleBlk (title information) 432
- titles, controlling generated chapter, part,
 - and appendix 74
- TList (list of tables) 432
- TM (Trademark) 433
- TOC (table of contents) 435
- toc attribute 207, 208, 220, 242, 276, 293,
 - 303, 306, 323, 379, 386, 387, 413
- tracking information, RevDefs
 - revision 409
- Trademark, TM 433
- trademarks 42
- translation considerations for conditional
 - text 172
- true or false, setting the properties
 - to 172
- turning change bars on and off 408

U

- UL (unordered list) 436
- UL element, using 23
- underlined 35
- unformatted tables 65
- unordered list, UL 436
- unordered lists 23
- using division (D) elements 18
- using parts to organize divisions 22
- using the body element 17

V

- values, property 171
- vanilla conditions 171
- Var (syntax variable) 437
- VAR element 130
- variable, MV message 355
- variable, PV parameter 400

- variable, Var syntax 437
- variable syntax element 130
- Version (product version number) 437
- version number, Version product 437
- VNet (IBM VNet mail address) 438
- VNet mail address, VNet 438
- VolId (volume identifier) 438
- volume identifier, VolId 438
- volumes, multiple 73, 285

W

- Warning (warning notice) 439
- warning notice, Warning 439
- warning notices 40
- web page, linking to 111
- web pages, linking to 82
- WebPage 440
- what's new xi
- where printed, PrtLoc, country 398
- where to place index entries 99
- where to put index entries 101
- widths
 - table column 58

X

- Xmp (example) 440
- xph 36
- XPh (example phrase) 441
- xpp:(justify=yes) 75
- XRef (cross reference) 442
- Xref element 51
- xSeries branding 287
- Xyvision override
 - keeping list items together 33
 - LEN nopage 144
- Xyvision overrides
 - Line Justification for DBCS
 - Languages 75
 - PDF bookmarks 286

Z

- zip code, PostalCode postal or 387
- zSeries branding 287

Readers' Comments — We'd Like to Hear from You

ID Workbench
IBMIDDoc User's Guide and Reference
Release 3.4

Publication No. SH21-0783-09

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



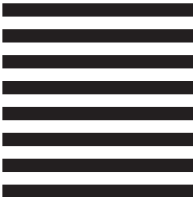
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
ATTN: Dept 542
3605 HWY 52 N
Rochester, MN
55901-9986



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SH21-0783-09

