OpenOffice.org XML File Format 1.0

Technical Reference Manual

Version 2

December 2002

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

December 2002

Copyrights and Trademarks

Copyright 2001 Sun Microsystems, Inc., 901 San Antonio Road, California 94303, U.S.A. All rights reserved.

This documentation is distributed under licenses restricting its use. You may make copies of and redistribute it, but you may not modify or make derivative works of this documentation without prior written authorization of Sun and its licensors, if any.

Sun, Sun Microsystems, the Sun logo, StarPortal, StarOffice,the StarOffice logo, Java, Java Beans JavaScript, and the Java Coffee Cup are trade marks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Contents

1 Introduction to OpenOffice.org XML 27
1.1 Namespaces 27
1.2 Structure of OpenOffice.org XML Documents 28
1.2.1 Document Root 28
1.2.2 Documents and Subdocuments 29
1.2.3 Primary Document Characteristics 29
1.3 Document Information 30
1.4 Styles 31
1.4.1 Location of Styles 32
1.4.2 Examples of Styles 32
1.5 Forms 33
1.6 Document Content 33
1.7 White-Space Processing and EOL Handling 33
1.8 Document Validation 34
1.8.1 Processing the Current Version 34

2 Common Document Content 35
2.1 Metadata 35
2.1.1 Generator 36
2.1.2 Title 36
2.1.3 Description 36
2.1.4 Subject 36
2.1.5 Keywords 37
2.1.6 Initial Creator 37
2.1.7 Creator 37
2.1.8 Printed By 37

2.1.9 Creation Date and Time 38
2.1.10 Modification Date and Time 38
2.1.11 Print Date and Time 38
2.1.12 Document Template 38
2.1.13 Automatic Reload 39
2.1.14 Hyperlink Behavior 40
2.1.15 Language 41
2.1.16 Editing Cycles 41
2.1.17 Editing Duration 42
2.1.18 User-defined Metadata 42
2.1.19 Document Statistics 42
2.1.20 Sample Metadata 44
2.2 Formatting Properties and Styles 44
2.2.1 Formatting Property Sets 45
2.2.2 Simple Formatting Properties 45
2.2.3 Complex Formatting Properties 45
2.2.4 Styles 46
2.2.5 Style Mappings 49
2.3 Page Styles and Layout 51
2.3.1 Page Master 52
2.3.2 Master Pages 58
2.3.3 Headers and Footers 59
2.3.4 Header and Footer Styles 60
2.3.5 Footnote Layout 61
2.3.6 Footnote Separator Line 61
2.4 Font Declarations 62
2.4.1 Font Declaration 63
2.5 Data Styles 64
2.5.1 Number Style 64
2.5.2 Currency Style 66
2.5.3 Percentage Style 68
2.5.4 Date Style 68
2.5.5 Time Style 72
2.5.6 Boolean Style 74
2.5.7 Text Style 75
2.5.8 Common Data Style Elements 76
2.5.9 Common Data Style Attributes 76
2.5.10 Common Number Style Attributes 81

2.6 Frames 84
2.6.1 Text Boxes 85
2.6.2 Images 86
2.6.3 Drawings 87
2.6.4 Objects 88
2.6.5 Applets 89
2.6.6 Plugins 91
2.6.7 Parameters 92
2.6.8 Floating Frames 92
2.6.9 Common Frame Elements 93
2.6.10 Common Frame Attributes 94
2.6.11 Frame Formatting Properties 97
2.6.12 Floating Frame Formatting Properties 103
2.6.13 Object Formatting Properties 104
2.6.14 Frame Events 104
2.7 Forms and Controls 105
2.8 Hyperlinks 105
2.8.1 Simple Hyperlinks 105
2.8.2 Client Side Image Maps 106
2.9 Number Format 109
2.9.1 Prefix and Suffix 109
2.9.2 Format Specification 110
2.9.3 Letter Synchronization in Number Formats 110
2.10 Scripts 110
2.11 Event Tables 111
2.11.1 Event 111
2.12 Change Tracking 112
2.12.1 Change Information 113
2.13 OpenOffice.org Application Settings 114
2.13.1 Base Settings 114
2.13.2 Sequence of Settings 114
2.13.3 Index Access of Sequences 115
2.13.4 Name Access of Sequences 115

3 Text Content 117
3.1 Headings and Paragraphs 117
3.1.1 Primary Heading and Paragraph Components 117
3.1.2 White-Space Characters 120

3.1.3 Tab Stops 120
3.1.4 Line Breaks 121
3.1.5 Text Styles 121
3.1.6 Text Formatting Properties 121
3.1.7 Hyperlinks 122
3.1.8 Bookmarks 124
3.1.9 Index Entries 125
3.1.10 References 125
3.1.11 Soft Hyphens, Hyphens, and Non-breaking Blanks 126
3.2 Sections 126
3.2.1 Section Source 128
3.2.2 DDE Source 129
3.3 Bulleted and Numbered Lists 130
3.3.1 List Blocks 131
3.3.2 List Header 132
3.3.3 List Item 132
3.3.4 List Styles 134
3.3.5 Number Level Style 135
3.3.6 Bullet Level Style 137
3.3.7 Image Level Style 138
3.4 Outline Numbering 140
3.4.1 Outline Style 141
3.4.2 Outline Level Style 141
3.5 Line Numbering 142
3.5.1 Line Numbering Configuration 142
3.5.2 Separator 145
3.5.3 Line Numbering Properties 145
3.6 Footnotes and Endnotes 146
3.6.1 Footnotes Configuration 146
3.6.2 Endnotes Configuration 149
3.6.3 Footnotes 149
3.6.4 Endnotes 151
3.7 Fields 152
3.7.1 Common Characteristics of Field Elements 152
3.7.2 Document Fields 153
3.7.3 Date Fields 153
3.7.4 Time Fields 154
3.7.5 Page Number Fields 155

3.7.6 Page Continuation Text 156
3.7.7 Sender Fields 157
3.7.8 Author Fields 160
3.7.9 Placeholders 161
3.7.10 Database Fields 162
3.7.11 Displaying Database Content 163
3.7.12 Selecting the Next Database Row 163
3.7.13 Selecting a Row Number 164
3.7.14 Displaying the Row Number 165
3.7.15 Display Current Database and Table 166
3.7.16 Metadata Fields 166
3.7.17 Conditional Text Fields 170
3.7.18 Hidden Text Field 172
3.7.19 Hidden Paragraph Fields 173
3.7.20 Chapter Fields 174
3.7.21 File Name Fields 175
3.7.22 Document Template Name Fields 176
3.7.23 Page Variable Fields 176
3.7.24 Macro Fields 178
3.7.25 DDE Connections 178
3.7.26 DDE Connection Fields 180
3.7.27 Reference Fields 180
3.7.28 Variable Fields 183
3.7.29 Declaring Simple Variables 183
3.7.30 Setting Simple Variables 184
3.7.31 Displaying Simple Variables 185
3.7.32 Simple Variable Input Fields 186
3.7.33 Declaring User Variables 187
3.7.34 Displaying User Variables 187
3.7.35 User Variable Input Fields 188
3.7.36 Declaring Sequence Variables 189
3.7.37 Using Sequence Fields 190
3.7.38 Expression Fields 191
3.7.39 Text Input Fields 192
3.7.40 Script Fields 192
3.7.41 Measure Fields 193
3.7.42 Table Formula Field 193
3.7.43 Common Field Attributes 194

3.8 Frames in Text Documents 199
3.8.1 Anchor Type 199
3.8.2 Anchor Position 199
3.8.3 Anchor Page Number 201
3.9 Ruby 201
3.9.1 Ruby Base 202
3.9.2 Ruby Text 203
3.10 Text Formatting Properties 203
3.10.1 Font Variant 203
3.10.2 Text Transformations 203
3.10.3 Color 203
3.10.4 Window Font Color 204
3.10.5 Text Outline 204
3.10.6 Crossing Out 204
3.10.7 Text Position 204
3.10.8 Font Name 205
3.10.9 Font Family 205
3.10.10 Font Family Generic 206
3.10.11 Font Style 206
3.10.12 Font Pitch 207
3.10.13 Font Character Set 207
3.10.14 Font Size 208
3.10.15 Relative Font Size 208
3.10.16 Letter Spacing 209
3.10.17 Language 209
3.10.18 Country 209
3.10.19 Font Style 210
3.10.20 Font Relief 210
3.10.21 Text Shadow 210
3.10.22 Underlining 211
3.10.23 Underline Color 211
3.10.24 Font Weight 211
3.10.25 Text Decoration Word Mode 212
3.10.26 Letter Kerning 212
3.10.27 Text Blinking 212
3.10.28 Text Background Color 213
3.10.29 Text Combine 213
3.10.30 Text Combine Start and End Characters 213

3.10.31 Text Emphasis 214
3.10.32 Text Autospace 214
3.10.33 Text Scale 214
3.10.34 Text Rotation Angle 215
3.10.35 Text Rotation Scale 215
3.10.36 Punctuation Wrap 215
3.10.37 Line Break 215
3.11 Paragraph Formatting Properties..... 216
3.11.1 Fixed Line Height 216
3.11.2 Minimum Line Height 216
3.11.3 Line Distance 216
3.11.4 Text Align 217
3.11.5 Text Align of Last Line 217
3.11.6 Justify Single Word 217
3.11.7 Break Inside 218
3.11.8 Widows 218
3.11.9 Orphans 218
3.11.10 Tab Stops 218
3.11.11 Hyphenation 220
3.11.12 Hyphenation Keep 220
3.11.13 Hyphenation Remain Char Count 220
3.11.14 Hyphenation Push Char Count 221
3.11.15 Maximum Hyphens 221
3.11.16 Drop Caps 222
3.11.17 Register True 223
3.11.18 Numbering Style 223
3.11.19 Left and Right Margins 223
3.11.20 Text Indent 223
3.11.21 Automatic Text Indent 224
3.11.22 Top and Bottom Margins 224
3.11.23 Page Sequence Entry Point 224
3.11.24 Break Before and Break After 224
3.11.25 Paragraph Background Color 225
3.11.26 Paragraph Background Image 225
3.11.27 Border 226
3.11.28 Border Line Width 227
3.11.29 Padding 228
3.11.30 Shadow 228

3.11.31 Keep with Next 228
3.11.32 Line Numbering 229
3.11.33 Text Autospace, Punctuation Wrap, Line Break 229
3.11.34 Vertical Alignment 229
3.12 Section Formatting Properties 229
3.12.1 Section Background 230
3.12.2 Columns 230
3.12.3 Column Specification 230
3.12.4 Column Separator 232
3.12.5 Protect 233
3.13 Change Tracking in Text Documents 233
3.13.1 Tracked Changes 233
3.13.2 Changed Regions 234
3.13.3 Region Start and End 234
3.13.4 Insertion 235
3.13.5 Deletion 235
3.13.6 Format Change 236
3.14 Optional Information 236
3.14.1 Auto Mark File for Alphabetical Indices 237

4 Table Content 239
4.1 General Introduction to OpenOffice.org Tables 239
4.2 Document Protection 240
4.3 Calculation Settings 240
4.3.1 Null Date 242
4.3.2 Iteration 242
4.4 Change Tracking in Spreadsheets 243
4.4.1 Tracked Changes 244
4.4.2 Dependences 244
4.4.3 Dependence 244
4.4.4 Deletions 245
4.4.5 Cell Content Deletion 245
4.4.6 Change Deletion 245
4.4.7 Insertion 246
4.4.8 Deletion 247
4.4.9 Cut Offs 248
4.4.10 Insertion Cut Off 249
4.4.11 Movement Cut Off 249

4.4.12 Movement 250
4.4.13 Target Range Address, Source Range Address 250
4.4.14 Change Track Cell 252
4.4.15 Cell Content Change 253
4.4.16 Cell Address 253
4.4.17 Previous 253
4.4.18 Rejection 254
4.4.19 Common Change Tracking Attributes 254
4.5 Tables 255
4.5.1 Table 255
4.5.2 Table Source 257
4.5.3 Scenario Table 258
4.5.4 Shapes 260
4.6 Columns 261
4.6.1 Grouping 261
4.6.2 Column Groups 261
4.6.3 Column Description 262
4.7 Rows 264
4.7.1 Grouping 264
4.7.2 Row Groups 264
4.7.3 Row 265
4.8 Cells 267
4.8.1 Table Cell 267
4.8.2 Cell Range Source 273
4.8.3 Annotation 275
4.8.4 Detective 276
4.8.5 Highlighted Range 276
4.8.6 Operation 277
4.8.7 Cell Address Entity 278
4.8.8 Cell Range Address Entity 279
4.8.9 Cell Range Address List Entity 279
4.9 Table Cell Content Validations 280
4.9.1 Table Cell Content Validation 280
4.9.2 Help Message 282
4.9.3 Error Message 282
4.9.4 Error Macro 283
4.10 Subtables 284
4.11 Label Ranges 288

4.11.1 Label Range 288
4.12 Named Expressions 289
4.12.1 Named Range 290
4.12.2 Named Expression 291
4.13 Filters 292
4.13.1 Table Filter 292
4.13.2 Filter And 293
4.13.3 Filter Or 293
4.13.4 Filter Condition 294
4.14 Database Ranges 296
4.14.1 Database Range 296
4.14.2 Database Source SQL 298
4.14.3 Database Source Table 299
4.14.4 Database Source Query 300
4.14.5 Sort 300
4.14.6 Sort By 301
4.14.7 Subtotal Rules 302
4.14.8 Sort Groups 303
4.14.9 Subtotal Rule 304
4.14.10 Subtotal Field 304
4.15 Data Pilot Tables 305
4.15.1 Data Pilot Table 305
4.15.2 Source Service 308
4.15.3 Source Cell Range 309
4.15.4 Filter 309
4.15.5 Data Pilot Field 309
4.15.6 Data Pilot Level 311
4.15.7 Data Pilot Subtotals 311
4.15.8 Data Pilot Subtotal 312
4.15.9 Data Pilot Members 312
4.15.10 Data Pilot Member 312
4.16 Consolidation 313
4.17 DDE Links 315
4.17.1 DDE Link 315
4.17.2 DDE Source 315
4.18 Table Formatting Properties 316
4.18.1 Table Width 316
4.18.2 Table Alignment 316

4.18.3 Table Left and Right Margin 316
4.18.4 Table Top and Bottom Margin 317
4.18.5 Page Sequence Entry Point 317
4.18.6 Break Before and Break After 317
4.18.7 Table Background and Background Image 317
4.18.8 Table Shadow 317
4.18.9 Keep with Next 317
4.18.10 May Break Between Rows 317
4.18.11 Border Model Property 318
4.18.12 Page Style 318
4.18.13 Display 318
4.19 Column Formatting Properties 319
4.19.1 Column Width 319
4.20 Table Row Formatting Properties 319
4.20.1 Row Height 319
4.20.2 Break Before and Break After 320
4.21 Table Cell Formatting Properties 320
4.21.1 Vertical Alignment 320
4.21.2 Text Align 320
4.21.3 Text Align Source 321
4.21.4 Text Outline 321
4.21.5 Direction 321
4.21.6 Vertical Glyph Orientation 321
4.21.7 Text Shadow 321
4.21.8 Cell Shadow 321
4.21.9 Cell Background 322
4.21.10 Cell Borders and Border Line Width 322
4.21.11 Padding 322
4.21.12 Left Margin 322
4.21.13 Wrap Option 322
4.21.14 Rotation Angle 322
4.21.15 Rotation Align 323
4.21.16 Cell Protect 323
4.21.17 Print Content 323
4.21.18 Data Style 323

5 Graphic Content 325
5.1 Master Pages 326

5.1.1 Handout Master 327
5.1.2 Presentation Notes 328
5.2 Drawing Pages 328
5.2.1 Background Style Properties 330
5.2.2 Presentation Notes 330
5.3 Drawing Shapes 331
5.3.1 Rectangle 331
5.3.2 Line 332
5.3.3 Polyline 333
5.3.4 Polygon 334
5.3.5 Path 334
5.3.6 Circle 335
5.3.7 Ellipse 337
5.3.8 Connector 338
5.3.9 Caption 340
5.3.10 Measure 342
5.3.11 Control 343
5.3.12 Page Thumbnail 343
5.3.13 Grouping 344
5.3.14 Common Drawing Shape Attributes 344
5.4 Presentation Shapes 347
5.4.1 Common Presentation Shape Attributes 347
5.4.2 Title 348
5.4.3 Outline 348
5.4.4 Subtitle 348
5.4.5 Text 348
5.4.6 Graphic 349
5.4.7 Object 349
5.4.8 Chart 349
5.4.9 Table 349
5.4.10 Orgcharts 349
5.4.11 Pages 349
5.4.12 Notes 349
5.5 3D Shapes 349
5.5.1 Scene 349
5.5.2 Light 352
5.5.3 Cube 353
5.5.4 Sphere 354

5.5.5 Extrude 355
5.5.6 Rotate 355
5.6 Graphic Style Elements 356
5.6.1 Gradient 356
5.6.2 Hatch 358
5.6.3 Image 359
5.6.4 Transparency Gradient 360
5.6.5 Marker 361
5.6.6 Dash 362
5.7 Stroke Properties 363
5.7.1 Style 364
5.7.2 Dash 364
5.7.3 Width 364
5.7.4 Color 364
5.7.5 Start Marker 365
5.7.6 End Marker 365
5.7.7 Start Marker Width 365
5.7.8 End Marker Width 365
5.7.9 Start Marker Center 365
5.7.10 End Marker Center 366
5.7.11 Opacity 366
5.7.12 Joint 366
5.8 Fill Properties 366
5.8.1 Style 367
5.8.2 Color 367
5.8.3 Gradient 367
5.8.4 Gradient Step Count 367
5.8.5 Hatch 367
5.8.6 Solid Hatch 368
5.8.7 Bitmap 368
5.8.8 Transparency 369
5.9 Text Animation Properties 370
5.10 Text Properties 372
5.10.1 Auto Grow Width and Height 372
5.10.2 Fit To Size 372
5.10.3 Text Area Vertical Align 372
5.10.4 Text Area Horizontal Align 372
5.11 Graphic Properties 373

5.11.1 Color Mode 373
5.11.2 Color Inversion 373
5.11.3 Adjust Luminance 373
5.11.4 Adjust Contrast 373
5.11.5 Adjust Gamma 374
5.11.6 Adjust Red 374
5.11.7 Adjust Green 374
5.11.8 Adjust Blue 374
5.11.9 Adjust Transparency 374
5.12 Shadow Properties 375
5.13 Connector Properties 376
5.14 Measure Properties 376
5.15 Caption Properties 379
5.16 3D Geometry Properties 381
5.17 3D Lighting Properties 382
5.18 3D Texture Properties 383
5.19 3D Material Properties 384
5.20 3D Shadow Properties 385
5.21 Layer Sets 385
5.21.1 Layer 385
5.22 Glue Points 385
5.22.1 Glue Point 385
5.23 Presentation Page Layouts 386
5.23.1 Presentation Placeholder 386
5.24 Presentation Page Attributes 387
5.24.1 Transition Type 387
5.24.2 Transition Style 388
5.24.3 Transition Speed 388
5.24.4 Page Duration 388
5.24.5 Page Visibility 389
5.24.6 Sound 389
5.24.7 Background Size 389
5.24.8 Background Objects Visible 389
5.24.9 Background Visible 390
5.25 Presentation Settings 390
5.25.1 Presentation 390
5.25.2 Shows 394
5.26 Presentation Animations 395

5.26.1 Sound 395
5.26.2 Show Shape 396
5.26.3 Show Text 398
5.26.4 Hide Shape 398
5.26.5 Hide Text 399
5.26.6 Dim 399
5.26.7 Play 399

6 Form Content 401
6.1 Forms 401
6.1.1 Form 401
6.2 Controls 407
6.2.1 Text 408
6.2.2 Text Area 409
6.2.3 Password 410
6.2.4 File 410
6.2.5 Formatted Text 411
6.2.6 Fixed Text 412
6.2.7 Combo Box 412
6.2.8 Item 413
6.2.9 List Box 414
6.2.10 Option 414
6.2.11 Button 415
6.2.12 Image 415
6.2.13 Check Box 416
6.2.14 Radio Button 417
6.2.15 Frame 418
6.2.16 Image Frame 418
6.2.17 Hidden 419
6.2.18 Grid 419
6.2.19 Column 420
6.2.20 Generic Control 420
6.3 Common Form and Control Attributes 420
6.4 Common Control Attributes 421
6.4.1 Database Attributes 430
6.5 Events 432
6.5.1 Events with an Equivalent HTML Event Type 432
6.5.2 OpenOffice.org Event Types 433

6.6 Properties 434
6.6.1 Property Set 434
6.6.2 Property 435
6.6.3 Property Value 436

7 Indexing 437
7.1 Basic Components of OpenOffice.org XML Indexes 437
7.1.1 Index Source 437
7.1.2 Index Body 438
7.1.3 Index Title Template 438
7.1.4 Index Entry Templates 438
7.1.5 Common Index and Index Source Attributes 439
7.2 Index Entries 441
7.2.1 Chapter Number 441
7.2.2 Chapter Information 441
7.2.3 Entry Text 442
7.2.4 Page Number 442
7.2.5 Fixed String 442
7.2.6 Bibliography Information 442
7.2.7 Tab Stop 443
7.2.8 Hyperlink Start and End 444
7.2.9 Example of an Index Entry Configuration 445
7.3 Index Source Styles 446
7.3.1 Index Source Style 446
7.4 Index Marks 447
7.4.1 Table of Content Index Marks 447
7.4.2 User-Defined Index Marks 448
7.4.3 Alphabetical Index Mark 449
7.4.4 Bibliography Index Mark 450
7.5 Table of Contents 451
7.5.1 Table of Content Source 452
7.5.2 Table of Content Entry Template 453
7.6 Index of Illustrations 455
7.6.1 Index of Illustration Source 455
7.6.2 Illustration Index Entry Template 456
7.7 Index of Tables 457
7.7.1 Table Index Source 457
7.7.2 Table Index Entry Template 458

7.8 Index of Objects 458
7.8.1 Object Index Source 458
7.8.2 Object Index Entry Template 460
7.9 User-Defined Index 460
7.9.1 User-Defined Index Source 460
7.9.2 User-Defined Index Entry Template 462
7.10 Alphabetical Index 463
7.10.1 Alphabetical Index Source 463
7.10.2 Alphabetical Index Entry Template 466
7.11 Bibliography 467
7.11.1 Bibliography Index Source 467
7.11.2 Bibliography Entry Template 468

8 Chart Content 471
8.1 Introduction to Chart Documents 471
8.2 Chart 472
8.3 Title 473
8.4 Subtitle 473
8.5 Legend 474
8.6 Plot Area 474
8.7 Wall 476
8.8 Floor 477
8.9 Axis 477
8.9.1 Grid 481
8.10 Series 481
8.10.1 Domain 482
8.11 Categories 482
8.12 Data Point 483
8.13 Common Chart Properties 483
8.13.1 Fill Properties 483
8.13.2 Stroke Properties 483
8.13.3 Text Properties 484
8.13.4 Alignment Properties 484
8.13.5 Data Label Properties 484
8.13.6 Statistical Properties 485
8.13.7 Rotation of Three-Dimensional Diagrams 487

9 User Interface Content 489
9.1 Menubars 489

9.1.1 Menubar 489
9.1.2 Menu 490
9.1.3 Menu Popup 490
9.1.4 Menu Item 490
9.1.5 Menu Separator 491
9.1.6 Common Menubar Attributes 492
9.2 Accelerators 492
9.2.1 Accelerator List 492
9.2.2 Accelerator Item 493
9.3 Status Bars 495
9.3.1 Status Bar 496
9.3.2 Status Bar Item 496
9.4 Toolbars 499
9.4.1 Toolbar 499
9.4.2 Toolbar Item 500
9.4.3 Toolbar Layouts 503
9.4.4 Toolbar Layout 503
9.5 Events 506
9.5.1 Events 507
9.5.2 Event 507
9.6 Images 508
9.6.1 Images Container 509
9.6.2 Images 509
9.6.3 Entry 510
9.6.4 External Images 511
9.6.5 External Entry 511

10 Dialog Content 513
10.1 Dialog Containers 513
10.1.1 Bulletin Board 514
10.2 Styles 515
10.2.1 Style 515
10.3 Controls 518
10.3.1 Common Control Attributes 518
10.3.2 Button 519
10.3.3 Check Box 520
10.3.4 List Box and Combo Box 521
10.3.5 Radio Group and Titled Box 523

10.3.6 Label 525
10.3.7 Text Field 525
10.3.8 Image 527
10.3.9 File Dialog 527
10.3.10 Date Field 528
10.3.11 Time Field 529
10.3.12 Numeric Field 530
10.3.13 Currency Field 531
10.3.14 Pattern Field 532
10.3.15 Formatted Field 533
10.3.16 Fixed Line 535
10.3.17 Scrollbar 535
10.3.18 Progress Bar 536
10.4 Events 537
10.4.1 Events Specified Using Common Event Names 537
10.4.2 Explicit Events Specified by Listener 539

11 Package Format 541
11.1 Introduction 541
11.2 Zip File Structure 542
11.3 Encryption 542
11.4 Manifest File 543
11.4.1 Manifest Root Element 543
11.4.2 File Entry 543
11.4.3 Encryption Data 544
11.4.4 Algorithm 544
11.4.5 Key Derivation 545

Preface

About This Manual

This manual describes the OpenOffice.org XML file format. XML is the new native file format for the OpenOffice.org suite, replacing the old binary file format. Our goal is twofold: to have a complete specification encompassing all OpenOffice.org components, and to provide an open standard for office documents. In our opinion, XML is ideal as an open standard because of the free availability of XML specifications and document type declarations (DTDs), and the XML support for XSL, XSLT, XLink, SVG, MathML, and many other important and emerging standards. One single XML format applies to different types of documents, for example, the same definition applies to tables in text documents and tables in spreadsheets.

This working draft manual contains the current specification of the OpenOffice.org XML file format. As the term "working draft" implies, the OpenOffice.org XML file format is work in progress. This fact has the following implications for this manual:

- The specification contained in this working draft is not complete. The XML specification for many of the OpenOffice.org features has not yet been decided or documented.
- This working draft may contain specifications that are not currently implemented in the OpenOffice.org XML import and export filters. This draft should also not omit specifications for any features that are already implemented in the OpenOffice.org XML filters but there may be exceptions to this.
- The specifications described in this working draft may change. This is especially true for specifications that are not currently implemented in the OpenOffice.org XML filters, but may also be the case for specifications that are already implemented. The reasons for changing the specifications include changes to related working drafts like XSL-FO or SVG, suggestions from reviewers of the manual, errors or inconsistencies that are found, or problems with new specifications that can only be resolved by changing existing specifications.
- This working draft may contain errors, missing information, or incomplete specifications.

Who Should Read This Manual

This manual is intended for software developers, both internal and external to Sun Microsystems®.

Structure of This Manual

This manual contains the following sections:

- Chapter 1, Introduction to OpenOffice.org XML
- Chapter 2, Common Document Content

- Chapter 3, Text Content
- Chapter 4, Table Content
- Chapter 5, Graphic Content
- Chapter 6, Form Content
- Chapter 7, Indexing
- Chapter 8, Chart Content
- Chapter 9, Package Format
- Glossary
- Index

Related Documentation

The following documents provide additional XML-related information:

- Extensible Markup Language (XML) 1.0, W3C Recommendation http://www.w3.org/TR/REC-xml.html
- Scalable Vector Graphics (SVG) 1.0 Specification, W3C Working Draft http://www.w3.org/TR/2000/03/WD-SVG-20000303/index.html
- Namespaces in XML, World Wide Web Consortium http://www.w3.org/TR/REC-xml-names
- XSL Transformations (XSLT) Version 1.0, W3C Recommendation http://www.w3.org/TR/xslt
- XML Path Language (XPath) Version 1.0, W3C Recommendation http://www.w3.org/TR/xpath
- XML Linking Language (XLink) Version 1.0, W3C Candidate Recommendation http://www.w3.org/TR/xlink
- Extensible Stylesheet Language (XSL) Version 1.0, W3C Working Draft http://www.w3.org/TR/xsl
- HTML 4.01 Specification, W3C Recommendation http://www.w3.org/TR/html401
- ISO 8601, http://www.iso.ch/markete/8601.pdf
- ISO 639, http://www.oasis-open.org/cover/iso639a.html
- ISO 3166, http://www.oasis-open.org/cover/country3166.html

At the time of writing this document, some of these related documents are working drafts. Please note that any information from these drafts that is used in this document may change.

Conventions

The following conventions are used in this manual:

Convention	Description
italic type	Italic type is used to indicate complete titles of manuals and to emphasize text.
boldface type	Boldface type indicates an item that is contained in the glossary.

Convention	Description
`courier font`	Courier font is used to indicate all XML elements and attributes and their values.

Terminology

The following terms are used frequently in this manual and have a specific meaning in the context of the manual:

Term	Meaning
Rules	This term is used in the tables that explain the OpenOffice.org XML elements and attributes. In this context, the term Rules means the ways in which you can use the element or attribute, what specific values are acceptable and not acceptable, and any other specific points that you need to know about using the element or attribute.

CHAPTER 1

Introduction to OpenOffice.org XML

This chapter introduces the structure and basic design features of the OpenOffice.org XML file format in OpenOffice.org documents. The chapter contains the following sections:

- Namespaces
- Structure of OpenOffice.org XML Documents
- Document Information
-
-
- Styles
- Forms
- Document Content
- White-Space Processing and EOL Handling
- Document Validation

1.1 Namespaces

lists the OpenOffice.org XML namespaces and their prefixes. For more information about XML namespaces, please refer to the Namespaces in XML specification, located at http://www.w3.org/TR/REC-xml-names

Table 1: OpenOffice.org XML Namespaces

Prefix	Description	Namespace
`office`	For all common pieces of information that are not contained in another, more specific namespace.	`http://openoffice.org/2000/office`
`style`	For elements and attributes that describe the style and inheritance model used by OpenOffice.org as well as some common formatting attributes.	`http://openoffice.org/2000/style`
`script`	For elements and attributes that represent scripts or events.	`http://openoffice.org/2000/script`
`api`	For elements and attributes that are related to the OpenOffice.org API.	`http://openoffice.org/2000/api`

Prefix	Description	Namespace
form	For elements and attributes that describe forms and controls.	http://openoffice.org/2000/form
text	For elements and attributes that may occur within text documents and text parts of other document types, such as the contents of a spreadsheet cell.	http://openoffice.org/2000/text
table	For elements and attributes that may occur within spreadsheets or within table definitions of a text document.	http://openoffice.org/2000/table
meta	For elements and attributes that describe meta information.	http://openoffice.org/2000/meta
number	For elements and attributes that describe data style information.	http://openoffice.org/2000/datastyle
draw	For elements and attributes that describe graphic content.	http://openoffice.org/2000/drawing
presen-tation	For elements and attributes that describe presentation content.	http://openoffice.org/2000/presentation
chart	For elements and attributes that describe chart content.	http://openoffice.org/2000/chart
xlink	The **XLink** namespace.	http://www.w3.org/1999/xlink
fo	The **XSL** formatting objects and properties namespace.	http://www.w3.org/1999/XSL/Format
svg	The **SVG** namespace.	http://www.w3.org/2000/svg
dialog	For elements and attributes that describe dialogs.	http://openoffice.org/2000/dialog

1.2 Structure of OpenOffice.org XML Documents

Each structural component in an OpenOffice.org XML document is represented by an **element**, with associated **attributes**. The structure of XML documents applies to all OpenOffice.org applications. There is no difference between a text document, a spreadsheet or a drawing, apart from the content. Also, all document types may contain different styles. You can exchange document content that is common to all document types from one type of document to another.

1.2.1 Document Root

The **document root element** is the primary element of an OpenOffice.org XML document. It contains the entire document. All types of OpenOffice.org XML documents, for example, text documents, spreadsheets, and drawing documents use the same type of document root element.

XML Code:	`<office:document>`
Rules:	All OpenOffice.org XML documents must have a document root element. All other sections of an OpenOffice.org XML document are represented by elements that are contained within the document root element.
DTD:	`<!ELEMENT office:document (office:meta?, office:settings?, office:script?, office:font-decls?, office:styles?, office:automatic-styles?, office:master-styles?, office:body)>`

1.2.2 Documents and Subdocuments

An OpenOffice.org XML document can be represented in the following two ways:

- As a single XML document that conforms to the OpenOffice.org document type definition.
- As a collection of several subdocuments. The subdocuments are valid XML document, but each subdocument has a different document root. Each subdocument stores a particular aspect of the XML document, for example, one subdocument can contain the style information and another subdocument can contain the content of the document. All types of documents, for example, text and spreadsheet documents, use the same document and subdocuments definitions.

When storing an OpenOffice.org XML document on disk, the OpenOffice.org software always uses the subdocument format, where each subdocument is stored as one entry in the overall package. This method is based on the popular ZIP file format.

The subdocuments that OpenOffice.org supports and the names that are used in the package are summarized in the following table:

Subdocument Name	Root Element	Sudocument Contents
`meta.xml`	`<office:document-meta>`	Document meta information, such as the author or the time of the last save action.
`styles.xml`	`<office:document-styles>`	Styles used in the document content and automatic styles used in the styles themselves.
`content.xml`	`<office:document-content>`	Document content and automatic styles used in the content.
`settings.xml`	`<office:document-settings>`	Application-specific settings, such as the window size or printer information.

The definitions of the root elements described in the table above are analogous to the definition of `<office:document>`, except that the child element specification is suitably restricted.

1.2.3 Primary Document Characteristics

You define primary document characteristics in the document root element using the following attributes:

- Class
- Version

Class

The `office:class` attribute identifies the document class of an OpenOffice.org XML document. Document classes that you can assign are as follows:

- Text
- Global document
- Spreadsheet
- Drawing
- Presentation

Although the document structure is the same for all document classes, most applications can only deal with a certain class of documents. For example, if you read a spreadsheet document using a word processor application there is always some loss of information. The class attribute enables an application to detect the document class without parsing the document. This is particularly useful in the following situations:

- When applications can deal with several document classes.
- When an **XSLT** transformation to another format should be applied, and there are several stylesheets available where each stylesheet is specific to a certain document class.

XML Code:	`office:class`
Rules:	You must specify a document class attribute for every OpenOffice.org XML document.
DTD:	`<!ATTLIST office:document office:class (text|text-global| spreadsheet|drawing|presentation|chart) #REQUIRED>`

Version

An OpenOffice.org XML file can contain the version number of the file format. The version number is in the format `revision.version`. If the file has a version and the OpenOffice.org application recognizes the DTD that belongs to this version, it may validate the document. Otherwise, the application does not need to validate the document, but the document must be **well formed**.

The `office:version` attribute provides the version number of the document.

XML Code:	`office:version`
Rules:	The `version` attribute is attached to the root element.
DTD:	`<!ATTLIST office:document office:version CDATA #IMPLIED>`
Notes:	The version number of the technical preview is 0.9.
	At some elements, you can attach custom attributes, which OpenOffice.org application keeps in case the document is exported. If you use this feature, then there must be no version number, because this could result in validating errors that prevent the document from loading.

1.3 Document Information

In this manual, information about an OpenOffice.org XML document is called **metadata**. Examples of metadata are:

- Document title

- Author
- Document creation date

You specify metadata within the `<office:meta>` element.

XML Code:	`<office:meta>`
Rules:	This element contains metadata.
DTD:	`<!ENTITY % meta "(meta:generator?,dc:title?, dc:description?,dc:subject?, meta:initial-creator?,meta:creation-date?, dc:creator?,dc:date?, meta:printed-by?,meta:print-date?, dc:keywords?,dc:language?, meta:editing-cycles?,meta:editing-duration?, meta:hyperlink-behaviour?,meta:auto-reload?, meta:template?,meta:user-defined*, meta: document-statistics?) <!ELEMENT office:meta %meta;>`
Note:	In the DTD, the element names correspond to the Dublin Core Element Set (http://purl.oclc.org/dc/). This is indicated by the `dc` namespace prefix.

1.4 Styles

An OpenOffice.org XML document contains the following types of **styles**:

- **Common styles**
  The XML representations of the styles that are available in the OpenOffice.org user interface are referred to as styles, or, where a differentiation from the other types of styles is required, they are referred to as common styles. The term *common* indicates that this is the type of style that an OpenOffice.org user, who is not interested in the OpenOffice.org XML file format, considers to be a style.
- **Automatic styles**
  An automatic style contains formatting properties that, in the user interface view of a document, are assigned to an object such as a paragraph. The term *automatic* indicates that the style is generated automatically at export time. In other words, formatting properties that are immediately assigned to a specific object are represented by an automatic style within an OpenOffice.org XML document. This way, a separation of content and layout is achieved.
- **Master styles**
  A master style is a common style that contains formatting information and additional content that is displayed with the document content when the style is applied. An example of a master style is an OpenOffice.org Draw master page. In this case, the additional content is any shapes that are displayed as the background of the draw page. Another example of master styles are page masters. In this case, the additional content is the headers and footers. Please note that the content that is contained within master styles is additional content that influences the representation of a document but does not change the content of a document.

As far as the OpenOffice.org user is concerned, all types of styles are part of the document. They represent the output device-independent layout and formatting information that the author of a document has used to create or edit the document. The assumption is that the author of the document wants this formatting and layout information to be preserved when the document is reloaded or displayed on a certain device, because this is common practice for documents created by word processors.

This type of style information differs from **CSS** or XSLT style sheets that are used to display a document. An additional style sheet for CSS, XSLT, and so on, is required to display an OpenOffice.org XML document on a certain device. This style sheet must take into account the styles in the document as well as the requirements and

capabilities of the output device. The ideal case is that this style sheet depends on the output device only.

1.4.1 Location of Styles

Common and automatic styles have the same OpenOffice.org XML representation, but they are contained within two distinct container elements, as follows:

- `<office:styles>` for common styles
- `<office:automatic-styles>` for automatic styles

Master styles are contained within a container element of its own:

- `<style:master-styles>`

1.4.2 Examples of Styles

The following examples illustrate the different types of OpenOffice.org **XML styles.**

Example: OpenOffice.org XML styles

```
<office:document ...>
  <office:styles>
    ...
  </office:styles>
  <office:automatic-styles>
    ...
  </office:automatic-styles>
</office:document>
```

Example: DTD

DTD:

```
<!ENTITY % styles "(style:default-style|style:style|
          text:list-style|number:number-style|
          number:currency-style|number:percentage-style|
          number:date-style|number:time-style|
          number:boolean-style|number:text-style|
          draw:gradient|draw:hatch|
          draw:fill-image|draw:marker|draw:stroke-dash|
          style:presentation-page-layout|draw:transparency)">
<!ELEMENT office:styles (%styles;|text:outline-style|
          text:footnotes-configuration|
          text:endnotes-configuration|
          text:bibliography-configuration|
          text:linenumbering-configuration)*>
<!ELEMENT office:automatic-styles (%styles;|style:page-master)
*)>
<!ELEMENT office:master-styles (draw:layer-set?,style:handout-
master?,style:master-page*)>
```

Note: Styles in a document are linked to the document by an XML element instead of a `<?xml-stylesheet?>` processing instruction.

1.5 Forms

The <office:forms> element contains all of the forms in an OpenOffice.org XML document.

XML Code:	<office:forms>
Rules:	Information to be supplied.
DTD:	<!ELEMENT office:forms (form:form)*>

1.6 Document Content

The <office:body> element contains the content of a document in one or more page sequences, as follows:

- The content distribution of text documents is specified in Section .
- A spreadsheet contains one page sequence for every table that is contained in the document.
- A drawing contains one page sequence for every page.

XML Code:	<office:body>
Rules:	Information to be supplied.
DTD:	<!ELEMENT office:body ANY>

1.7 White-Space Processing and EOL Handling

The **W3C** XML specification requires that white-space characters are ignored for elements that have element content, in other words that contain elements but not text. This condition applies to the following white-space and end-of-line (**EOL**) Unicode characters:

- HORIZONTAL TABULATION (0x0009)
- LINE FEED (0x000A)
- CARRIAGE RETURN (0x000D)
- SPACE (0x0020))

For any other element, white-spaces are preserved by default. Unless otherwise stated, there is no special processing for any of the four white-space characters. For some elements, different white-space processing may take place, for example the paragraph element.

The XML specification also requires that any of the four white-space characters that is contained in an attribute value is normalized to a SPACE character.

One of the following characters may be used to represent line ends:

- LINE FEED
- CARRIAGE RETURN
- The sequence of the characters CARRIAGE RETURN and LINE FEED

Conforming to the XML specification, all the possible line ends are normalized to a single LINE FEED character.

As a consequence of the white-space and EOL processing rules, any CARRIAGE RETURN characters that are contained either in the text content of an element or in an attribute value must be encoded by the character entity

`
`. The same applies to the HORIZONTAL TABULATION and LINE FEED characters if they are contained in an attribute value.

1.8 Document Validation

In general, an XML document may be validated or not. If it is validated, it must match the DTD exactly. In some situations, it is not appropriate to validate an XML document. For example:

- The document was created by a different version of the OpenOffice.org application.
- The document contains a custom extension.

These types of documents may contain attributes, attribute values, or elements that are unknown to the application that processes the file. The forward-compatible processing rules describe how an application should handle such elements and attributes to get the most from the contents of the document.

1.8.1 Processing the Current Version

Validation and forward-compatible processing is controlled by the `office:version` attribute. Every application has a **current file format** version, which stores all the information contained in the document without losing any information when the document is read again. An application may also be able to process documents created using other versions. For simplicity, it is assumed that these versions are also covered by the concept of a current version. For every version, there is a specific DTD that may be used to validate documents.

shows the relationships between a current version of a document, consisting of a major version and a minor version, and the way it is processed by an application:

Table 1: Processing Relationships For Current Document Versions

If the major version of the document is...	and/or	Forward-compatible processing is...	Validation Status
...the same as the current major version...	...and the minor version of the document is less or the same as the current minor version...	Disabled	The document may be validated, but it does not need to be.
...the same as the current major version...	...and the minor version of the document is greater than the current minor version...	Enabled	The document must not be validated. The only type of information that may be lost is information about features that are supported by the more recent version of OpenOffice.org.
...different from the current major version...	...or if there is no version contained in the document at all...	Enabled	The document must not be validated.

CHAPTER 2

Common Document Content

This chapter describes the OpenOffice.org XML file format for content that is common to all document types. It contains the following sections:

- Metadata
- Formatting Properties and Styles
- Page Styles and Layout
- Data Styles
- Frames
- Forms and Controls
- Hyperlinks
- Number Format
- Scripts
- Event Tables
- Change Tracking
-
- OpenOffice.org Application Settings

2.1 Metadata

Metadata is general information about a document. In an OpenOffice.org XML document, all of the metadata elements are contained in an `<office:meta>` element, usually located at start of the document.

XML Code:	`<office:meta>`
Rules:	This element contains metadata elements.
DTD:	`<!ENTITY % meta "(meta:generator?,dc:title?, dc:description?,dc:subject?, meta:initial-creator?,meta:creation-date?, dc:creator?,dc:date?, meta:printed-by?,meta:print-date?, dc:keywords?,dc:language?, meta:editing-cycles?,meta:editing-duration?, meta:hyperlink-bahviour?,meta:auto-reload?, meta:template?,meta:user-defined*, meta:document-statistic?) <!ELEMENT office:meta %meta;>`

2.1.1 Generator

The `<meta:generator>` element contains a string that identifies the application or tool that was used to create or last modify the XML document.

XML Code:	`<meta:generator>`
Rules:	If the application that created the document could not provide an identifier string, the application does not export this element. If another application modifies the document and it cannot provide a unique identifier, it is not allowed to export the original identifier belonging to the application that created the document.
DTD:	`<!ELEMENT meta:generator (#PCDATA)>`

2.1.2 Title

The `<dc:title>` element specifies the title of the document.

XML Code:	`<dc:title>`
Rules:	
DTD:	`<!ELEMENT dc:title (#PCDATA)>`

2.1.3 Description

The `<dc:description>` element contains a brief description of the document.

XML Code:	`<dc:description>`
Rules:	
DTD:	`<!ELEMENT dc:description (#PCDATA)>`

2.1.4 Subject

The `<dc:subject>` element specifies the subject of the document.

XML Code:	`<dc:subject>`
Rules:	
DTD:	`<!ELEMENT dc:subject (#PCDATA)>`

2.1.5 Keywords

The `<meta:keywords>` element contains keywords for the document. The metadata can contain any number of `<meta:keyword>` elements, each element specifying one keyword.

XML Code:	`<meta:keywords>`
Rules:	This element can contain one keyword.
DTD:	`<!ELEMENT meta:keywords (meta:keyword)*>` `<!ELEMENT meta:keyword (#PCDATA)>`

2.1.6 Initial Creator

The `<meta:initial-creator>` element specifies the name of the person who created the document initially.

XML Code:	`<meta:initial-creator>`
Rules:	
DTD:	`<!ELEMENT meta:initial-creator (#PCDATA)>`

2.1.7 Creator

The `<dc:creator>` element specifies the name of the person who last modified the document

XML Code:	`<dc:creator>`
Rules:	
DTD:	`<!ELEMENT dc:creator (#PCDATA)>`
Notes:	The name of this element was chosen for compatibility with the Dublin Core.

2.1.8 Printed By

The `<meta:printed-by>` element specifies the name of the last person who printed the document.

XML Code:	`<meta:printed-by>`
Rules:	
DTD:	`<!ELEMENT meta:printed-by (#PCDATA)>`

2.1.9 Creation Date and Time

The <meta:creation-date> element specifies the date and time when the document was created initially.

XML Code:	<meta:creation-date>
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 24 for a pointer to ISO 8601.
DTD:	<!ENTITY % c-date-time "(#PCDATA)"> <!ELEMENT meta:creation-date %c-date-time;>

2.1.10 Modification Date and Time

The <dc:date> element specifies the date and time when the document was last modified.

XML Code:	<dc:date>
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 24 for a pointer to ISO 8601.
DTD:	<!ELEMENT dc:date %c-date-time;>
Notes:	The name of this element was chosen for compatibility with the Dublin Core.

2.1.11 Print Date and Time

The <meta:print-date> element specifies the date and time when the document was last printed.

XML Code:	<meta:print-date>
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 24 for a pointer to ISO 8601.
DTD:	<!ELEMENT meta:print-date %c-date-time;>

2.1.12 Document Template

The <meta:template> element contains a URL for the document template that was used to create the document. The URL is specified as an XLink.

XML Code:	<meta:template>
Rules:	This element conforms to the XLink Specification. See page 24 for a pointer to this specification.
DTD:	<!ELEMENT meta:template EMPTY> <!ATTLIST meta:template xlink:type (simple) #FIXED "simple"> <!ATTLIST meta:template xlink:role CDATA #IMPLIED> <!ATTLIST meta:template xlink:actuate (onRequest) #IMPLIED>

The attributes that you can associate with the <meta:template> element are:

- Template location
- Template title
- Template modification date and time

Template Location

An `xlink:href` attribute specifies the location of the document template.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST meta:template xlink:href %url; #REQUIRED>`

Template Title

The `xlink:title` attribute specifies the name of the document template.

XML Code:	`xlink:title`
Rules:	
DTD:	`<!ATTLIST meta:template xlink:title CDATA #IMPLIED>`

Template Modification Date and Time

The `meta:date` attribute specifies the date and time when the template was last modified, prior to being used to create the current document.

XML Code:	`meta:date`
Rules:	To conform with ISO 8601, the date and time format is YYYY-MM-DDThh:mm:ss. See page 24 for a pointer to ISO 8601.
DTD:	`<!ENTITY %date-time "CDATA">` `<!ATTLIST meta:template meta:date %date-time; #IMPLIED>`

2.1.13 Automatic Reload

The `<meta:auto-reload>` element specifies whether a document is reloaded or replaced by another document after a certain period of time has elapsed.

XML Code:	`<meta:auto-reload>`
Rules:	
DTD:	`<!ELEMENT meta:auto-reload EMPTY>`

The attributes that you can associate with the `<meta:auto-reload>` element are:

- Reload URL
- Reload delay

Reload URL

If a loaded document should be replaced by another document after a certain period of time, the `<meta:auto-reload>` element is presented as an XLink. An `xlink:href` attribute identifies the URL of the replacement document.

XML Code:	`xlink:href`
Rules:	

DTD:	`<!ATTLIST meta:auto-reload xlink:type (simple) #IMPLIED>`
	`<!ATTLIST meta:auto-reload xlink:show (replace) #IMPLIED>`
	`<!ATTLIST meta:auto-reload xlink:actuate (onLoad) #IMPLIED>`
	`<!ATTLIST meta:auto-reload xlink:href %url; #IMPLIED>`

Reload Delay

The `meta:delay` attribute specifies the reload delay.

XML Code:	`meta:delay`
Rules:	To conform with ISO 8601, the format of the value of this attribute is `PnYnMnDTnHnMnS`. See Section 5.5.3.2 of ISO 8601 for more detailed information on this time format. See page 24 for a pointer to ISO 8601.
DTD:	`<!ENTITY % duration "CDATA">` `<!ATTLIST meta:auto-reload meta:delay %duration; "P0S">`

2.1.14 Hyperlink Behavior

The `<meta:hyperlink-behaviour>` element specifies the default behavior for hyperlinks in the document.

XML Code:	`<meta:hyperlink-behaviour>`
Rules:	
DTD:	*To be supplied*

The attribute that you can associate with the `<meta:hyperlink-behaviour>` element is:

- Target frame

Target Frame

The `meta:target-frame-name` attribute specifies the name of the default target frame in which to display a document referenced by a hyperlink.

XML Code:	`meta:target-frame-name`
Rules:	This attribute can have one of the following values: • `_self` : The referenced document replaces the content of the current frame. • `_blank` : The referenced document is displayed in a new frame. • `_parent` : The referenced document is displayed in the parent frame of the current frame. • `_top` : The referenced document is displayed in the topmost frame, that is the frame that contains the current frame as a child or descendent but is not contained within another frame. • A frame name : The referenced document is displayed in the named frame. If the named frame does not exist, a new frame with that name is created. To conform with the XLink Specification, an additional `xlink:show` attribute is attached to the `<meta:hyperlink-behavour>` element. See page 24 for a pointer to the XLink Specification. If the value of the `meta:target-frame-name` attribute is `_blank`, the `xlink:show` attribute value is `new`. If the value of the `meta:target-frame-name` attribute is any of the other value options, the value of the `xlink:show` attribute is `replace`.
DTD:	`<!ATTLIST meta:hyperlink-behaviour meta:target-frame-name CDATA #IMPLIED>` `<!ATTLIST meta:hyperlink-behaviour xlink:show (new|replace) #IMPLIED>`

2.1.15 Language

The `<dc:language>` element specifies the default language of the document.

XML Code:	`<dc:language>`
Rules:	The manner in which the language is represented is similar to the language tag described in RFC 1766 , located at http://info.internet.isi.edu/in-notes/rfc/files/rfc1666.txt. It consists of a two-letter Language Code taken from the ISO 639 standard optionally followed by a hyphen (-) and a two-letter Country Code taken from the ISO 3166 standard. See page 24 for pointers to ISO 639 and ISO 3166.
DTD:	`<!ENTITY % c-language "(#PCDATA)">` `<!ELEMENT dc:language %c-language;>`

2.1.16 Editing Cycles

The `<meta:editing-cycles>` element specifies the number of editing cycles the document has been through.

XML Code:	`<meta:editing-cycles>`
Rules:	The value of this element is incremented every time the document is saved. The element contains the number of editing cycles as text.
DTD:	`<!ENTITY % c-number "(#PCDATA)">` `<!ELEMENT meta:editing-cycles %c-number;>`

2.1.17 Editing Duration

The <meta:editing-duration> element specifies the total time spent editing the document.

XML Code:	<meta:editing-duration>
Rules:	The duration is represented in the manner described in Section 5.5.3.2 of ISO 8601. See page 24 for a pointer to ISO 8601.
DTD:	<!ENTITY % c-duration "(#PCDATA)"> <!ELEMENT meta:editing-duration %c-duration;>

2.1.18 User-defined Metadata

The <meta:user-defined> element specifies any additional user-defined metadata for the document.

XML Code:	<meta:user-defined>
Rules:	Each instance of this element can contain one piece of user-defined metadata. The element contains: • A meta:name attribute, which identifies the name of the metadata element. • The value of the element, which is the metadata.
DTD:	<!ELEMENT meta:user-defined (#PCDATA)> <!ATTLIST meta:user-defined meta:name CDATA #REQUIRED>

2.1.19 Document Statistics

The <meta:document-statistic> element specifies the statistics of the document, for example, the page count, word count, and so on. The statistics are specified as attributes of the <meta:document-statistic> element and the statistics that are exported with the document depend on the document type and the application used to create the document.

Document Type	Document Statistics Attributes
Text	meta:page-count meta:table-count meta:draw-count meta:ole-object-count meta:paragraph-count meta:word-count meta:character-count meta:row-count
Spreadsheet	meta:page-count meta:table-count meta:cell-count meta:object-count
Graphic	meta:page-count meta:object-count

XML Code:	<meta:document-statistic>
Rules:	

DTD:

```
<!ELEMENT meta:document-statistic EMPTY>
<!ATTLIST meta:document-statistic meta:page-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:table-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:draw-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:ole-object-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:paragraph-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:word-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:character-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:row-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:cell-count %
positiveInteger; #IMPLIED>
<!ATTLIST meta:document-statistic meta:object-count %
positiveInteger; #IMPLIED>
```

2.1.20 Sample Metadata

Example: Sample metadata in an OpenOffice.org XML document

```
<office:meta>
  <dc:title>Title of the document</dc:title>
  <dc:description>Description/Comment for the document</dc:description>
  <meta:initial-creator>User Name</meta:initial-creator>
  <meta:creation-date>1999-10-18T12:34:56</meta:creation-date>
  <dc:creator>User Name</dc:creator>
  <dc:date>1999-10-19T15:16:17</dc:date>
  <meta:printed-by>User Name</meta:printed-by>
  <meta:print-date>1999-10-20T16:17:18</meta:print-date>
  <dc:subject>Description of the document</dc:subject>
  <meta:duration-time>PT5H10M10S</meta:editing-duration>
  <meta:keywords>
    <meta:keyword>First keyword</meta:keyword>
    <meta:keyword>Second keyword</meta:keyword>
    <meta:keyword>Third keyword</meta:keyword>
  </meta:keywords>
  <meta:template xlink:type="simple"
    xlink:href="file:///c|/office52/share/template/german/finance/budget.
vor"
    xlink:title="Template name"
    meta:date="1999-10-15T10:11:12" />
  <meta:auto-reload
    xlink:type="simple"
    xlink:href="file:///..."
    meta:delay="P60S" />
  <dc:language>de-DE</dc:language>
  <meta:user-defined meta:name="Field 1">Value 1</meta:user-defined>
  <meta:user-defined meta:name="Field 2">Value 2</meta:user-defined>
</office:meta>
```

2.2 Formatting Properties and Styles

Many objects in an OpenOffice.org document have formatting properties. A formatting property influences the visual representation of an object but it does not contribute to the content or structure of the document. Examples of formatting properties are:

- Font family
- Font size
- Font color
- Page margins

In an OpenOffice.org XML document, formatting properties are only stored within styles. This differs to the OpenOffice.org user interface, where you can assign formatting properties to an object directly or you can apply a style to the object. Assigning formatting properties to an object directly has the same effect as assigning an unnamed style with the same properties to that object. Therefore, user interface styles remain unchanged conceptually in the OpenOffice.org XML file format, while formatting properties assigned directly to an object are assumed to be unnamed styles. In order to use unnamed styles, they are assigned a name and therefore become automatic styles.

There are two main reasons for using styles to store formatting properties:

1. You can keep the format and layout of the document separate from the document content.

2. If two or more objects have the same formatting properties and styles assigned, the formatting properties that are assigned to the objects directly can be represented by a single automatic style for all objects. This saves disk space and allows styles to integrate seamlessly into the overall document style.

2.2.1 Formatting Property Sets

A document can contain several style elements. To acquire a common set of formatting properties, you use a `<style:properties>` element which is included as a child element of any style element. This container element offers two important advantages, as follows:

- Formatting properties can be addressed by CSS or XSL stylesheets regardless of the style type.
- Styles contain additional information that is not a formatting property, for example, the style name and parent style. It is good practice to separate this type of information.

XML Code:	`<style:properties>`
Rules:	You must use this container element to store a common set of formatting properties.
DTD:	`<!ELEMENT style:properties ANY>`

2.2.2 Simple Formatting Properties

Most formatting properties are simple and can be represented as attributes of the `<style:properties>` element. Where possible, XSL attributes are used to represent formatting properties. In this specification, the namespace prefix `fo` is used for XSL properties, that is properties that are part of the XSL-FO namespace. In general, formatting properties that cannot be represented by XSL properties are part of the `style` namespace.

In OpenOffice.org, there are some formatting properties that you cannot specify without specifying one or more additional formatting properties. If the required properties are missing, a default value is assumed. This specification highlights the properties where this limitation applies.

Example: Simple style properties

This example shows a formatting property container that specifies an upper margin of 1 cm as well as a lower margin of 0.5 cm:

```
<style:properties fo:margin-left="1cm" fo:margin-bottom=".5cm"/>
```

2.2.3 Complex Formatting Properties

If a formatting property is too complex to be represented by XML attributes, it is represented by an XML element. Each such property is represented by an element type of its own.

Example: Complex formatting properties

This is an example of a formatting property container that specifies upper and lower margins as well as tab stop position at 2 and 4 cm.

```
<style:properties>
  <style:tab-stops>
     <style:tab-stop style:position="2cm"/>
     <style:tab-stop style:position="4cm"/>
  </style:tab-stops>
</style:properties>
```

2.2.4 Styles

Some style families are very similar in structure and can be represented by the same element. For example, the `<style:style>` element can represent paragraph, text, and frame styles.

XML Code:	`<style:style>`
Rules:	
DTD:	`<!ELEMENT style:style (style:properties?,style:map*)>`
Note:	The same elements can represent common and automatic styles but the difference is that they are contained in different container elements. An exception to this is automatically generated styles for elements contained in the `<office:styles>` elements, which are marked with the `style:automatic` attribute.

The attributes that you can associate with the `<style:style>` element are:

- Style name
- Style family
- Automatic
- Parent style
- Next style
- List style
- Master page name
- Automatically update
- Formatting properties
- Outline level numbering

Style Name

The `style:name` attribute identifies the name of the style.

XML Code:	`style:name`
Rules:	
DTD:	`<!ENTITY % style-name "CDATA">` `<!ATTLIST style:style style:name %style-name; #IMPLIED>`
Notes:	This attribute, combined with the `style family` attribute, uniquely identifies a style. You cannot have two styles with the same family and the same name. For automatic styles, a name is generated during document export. If the document is exported several times, you cannot assume that the same name is generated each time. In an XML document, the name of each style is a unique name that is independent of the language selected for the OpenOffice.org user interface. These style names are the same as the names used by the OpenOffice.org API and are usually the names used for the English version of the user interface.

Style Family

The `style:family` attribute identifies the family of the style, for example, paragraph, text, or frame.

XML Code:	`style:family`
Rules:	
DTD:	`<!ATTLIST style:style style:family (paragraph|text|frame) #REQUIRED>`

Automatic

The `style:automatic` attribute specifies whether or not the style is an automatic style.

XML Code:	`style:automatic`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST style:style style:automatic %boolean; #IMPLIED>`

Parent Style

The `style:parent-style-name` attribute specifies the name of the parent style.

XML Code:	`style:parent-style-name`
Rules:	If a parent style is not specified, a default parent style defined by the application is used. The parent style cannot be an automatic style and if you specify a parent style that is not defined, an error occurs.
DTD:	`<!ATTLIST style:style style:parent-style-name %style-name; #IMPLIED>`

Next Style

The `style:next-style-name` attribute specifies the style to used for the next paragraph if a paragraph break is inserted in the user interface.

XML Code:	`style:next-style-name`
Rules:	If you do not include this attribute, by default the current style is used as the next style.
DTD:	`<!ATTLIST style:style style:next-style-name %style-name; #IMPLIED>`
Note:	This concept is only supported by the paragraph style family.

List Style

A paragraph style can have an associated list style. This applies to automatic and common styles.

XML Code:	`style:list-style-name`
Rules:	If you specify a list style, it is *only* applied to paragraphs that are contained in a list, where the list does not specify a list style itself, and the list has no list style specification for any of its parents.
DTD:	`<!ATTLIST style:style style:list-style-name %style-name; #IMPLIED>`

Master Page Name

A paragraph or table style can have an associated `style:master-page-name` attribute. This applies to automatic and common styles. If this attribute is associated with a style, a page break is inserted when the style is applied and the specified master page is applied to the preceding page.

XML Code:	`style:master-page-name`
Rules:	This attribute is ignored if it is associated with a paragraph style that is applied to a paragraph within a table.
DTD:	`<!ATTLIST style:style style:master-page-name %style-name; #IMPLIED>`

Automatically Update

The `style:auto-update` attribute determines whether or not styles are automatically updated when the formatting properties of an object that has the style assigned to it are changed. For example, if you have a paragraph style that contains a formatting property specifying that paragraph text is centered, and this paragraph style is applied to a paragraph. If you manually change the formatting of the paragraph text to be right-aligned and the value of the `style:auto-update` is `true`, the paragraph style is automatically updated to reflect the new paragraph formatting and every paragraph that uses the paragraph style is also modified to right-align the paragraph text.

XML Code:	`style:auto-update`
Rules:	This attribute can have a value of `true` or `false`.
DTD:	`<!ATTLIST style:style style:auto-update %boolean; "false">`
Implementation limitation:	This feature is only supported in OpenOffice.org Writer documents.

Formatting Properties

If a style has formatting attributes assigned, the style element contains a formatting property container element

called <style:properties>. See Section 2.2.1 for detailed information about this element.

Outline Numbering Level

See Section 3.4.2 for information on the outline numbering level for a style.

Sample Style

Example: OpenOffice.org XML representation of the "Text body" paragraph style

```
<style:style style:name="Text body" style:family="paragraph"
             style:parent-style-name="Standard"
             style:pool-id="2049">
  <style:properties fo:margin-top="0cm" fo:margin-bottom=".21cm"/>
</style:style>
```

2.2.5 Style Mappings

Note: The elements and attributes described in this section only apply to conditional styles.

The <style:map> element specifies the mapping to another style, if certain conditions exist.

XML Code:	<style:map>
Rules:	This element is contained in the style element of a conditional style. There is one element for every condition that the style uses.
DTD:	<!ELEMENT style:map EMPTY>
Implementation limitation:	Conditional styles are only supported by OpenOffice.org Writer paragraph styles.

The attributes that you can associate with the <style:map> element are:

- Condition
- Applied style
- Base cell address

Condition

The style:condition attribute specifies the condition in which a style map should be applied.

XML Code:	`style:condition`
Rules:	The value of this attribute is a Boolean expression. The syntax of the expression is similar to the XPath syntax. The following conditions are valid for paragraph styles: • `list-level()=`*n*, where *n* is a number between 1 and 10 • `outline-level()=`*n*, where *n* is a number between 1 and 10 • `table()` and `table-header()` • `section()` • `header()` and `footer()` • `footnote()` and `endnote()` • `Condition ::= TrueFunction | TrueCondition` ➢ `TrueFunction ::= is-true-formula(Formula) | cell-content-is-between(Value, Value) | cell-content-is-not-between(Value, Value)` ➢ `TrueCondition ::= Expression` • `Expression ::= GetFunction Operator Value` ➢ `GetFunction ::= cell-content()` ➢ `Operator ::= '<' | '>' | '<=' | '>=' | '=' | '!='` ➢ `Value ::= NumberValue | String | Formula` **Note:** A `NumberValue` is a whole or decimal number. A `String` comprises one or more characters surrounded by quotation marks. A `Formula` is a formula (see 4.8.1) without the equals (=) sign at the beginning. You must include an `Operator`. The number in a `NumberValue` or `Formula` cannot contain comma separators for numbers of 1000 or greater. The conditions that apply for different types of styles may differ.
DTD:	`<!ATTLIST style:map style:condition CDATA #REQUIRED>`
Notes:	If an application detects a condition that it does not recognize, it must ignore the entire `<style:map>` element.

Applied Style

The `style:apply-style-name` attribute specifies the style to apply when the condition specified by the `style:condition` attribute is `true`.

XML Code:	`style:apply-style-name`
Rules:	If the referenced style is undefined or is an automatic style, an error occurs.
DTD:	`<!ATTLIST style:map style:apply-style-name %style-name #REQUIRED>`

Base Cell Address

The `style:base-cell-address` attribute specifies the base cell for relative addresses in formulas.

XML Code:	`style:base-cell-address`
Rules:	This attribute only applies to cell styles where the condition contains a formula. The value of this attribute must be an absolute cell address with a table name.
DTD:	`<!ATTLIST style:map style:base-cell-address %cell-address; #IMPLIED>`

Sample Style Mapping

Example: Style mapping

```
<style:style style:name="Text body" style:family="paragraph"
             style:parent-style-name="Standard"
             style:next-style-name="Text body">
  <style:properties fo:margin-top="0cm" fo:margin-bottom=".21cm"/>
  <style:map style:condition="footnote" style:apply-style-name="footnote"/>
  <style:map style:condition="heading(1)"
             style:apply-style-name="Heading 1"/>
  <style:map style:condition="heading(2)"
             style:apply-style-name="Heading 2"/>
</style:style>
```

2.3 Page Styles and Layout

The style and layout of the pages in a document is determined by:

- Page Masters
- Master Pages

A **page master** describes the physical properties or geometry of a page, for example, page size, margins, header height, and footer height.

A **master page** is a template for pages in a document. It contains a reference to a page master which specifies the physical properties of the page and can also contain static content that is displayed on all pages in the document that use the master page. Examples of static content are headers, footers, or background graphics.

If a text or spreadsheet document is displayed in a paged layout, the master pages are instantiated to generate a sequence of pages containing the document content. When a master page is instantiated, an empty page is generated with the properties of the page master and the static content of the master page. The body of the page is then filled with content. If multiple pages in a document use the same master page, the master page can be instantiated several times within the document.

In text and spreadsheet documents, you can assign a master page to paragraph and table styles using a `style:master-page-name` attribute. Each time the paragraph or table style is applied to text, a page break is inserted before the paragraph or table. The page that starts at the page break position uses the specified master page.

In drawings and presentations, you can assign master pages to drawing pages using a `style:parent-style-name` attribute.

Note: The OpenOffice.org XML paging methodology differs significantly from the methodology used in XSL. In XSL, headers and footers are contained within page sequences that also contain the document content. The

content of headers and footers can be changed or omitted without affecting the document content. In OpenOffice.org XML, headers and footers are contained in page styles.

2.3.1 Page Master

The <style:page-master> element specifies the physical properties of a page.

XML Code:	<style:page-master>
Rules:	This element contains a <style:properties> element which specifies the formatting properties of the page and three optional elements that specify the properties of headers, footers, and a footnote section.
DTD:	<!ELEMENT style:page-master (style:properties, style:header-style?, style:footer-style?, style:footnote-layout?)>

The attributes that you can associate with the <style:page-master> element are:

- Name
- Page usage
- Page size
- Page number format
- Paper tray
- Print orientation
- Margins
- Border
- Border line width
- Padding
- Shadow
- Background
- Columns
- Register-truth
- Print
- Print page order
- First page number
- Scale
- Table centering
- Maximum footnote height
- Footnote separator

Name

The `style:name` attribute specifies the name of the page master.

XML Code:	`style:name`
Rules:	
DTD:	`<!ATTLIST style:page-master style:name %styleName #REQUIRED>`

Page Usage

The `style:page-usage` attribute specifies the type of pages that the page master should generate.

XML Code:	`style:page-usage`
Rules:	The value of this attribute can be: • `all` – all pages are the same • `left` – all pages are left pages • `right` – all pages are right pages • `mirror` – left and right pages are the same except that the margins are mirrored
DTD:	`<!ATTLIST style:page-master style:page-usage (all|left|right|mirrored) "all">`
Note:	XSL supports left and right pages but uses a different model to OpenOffice.org XML.

Page Size

The `fo:page-width` and `fo:page-height` attributes specify the physical size of the page.

XML Code:	`fo:page-width` `fo:page-height`
Rules:	The `fo:page-width` attribute must correspond to the orientation of the page. For example, if a page is printed in portrait, the `fo:page-width` attribute specifies the width of the shorter page side. If the page is printed in landscape, the `fo:page-width` attribute specifies the width of the longer page side.
DTD:	`<!ATTLIST style:properties fo:page-width %length; #IMPLIED>` `<!ATTLIST style:properties fo:page-height %length; #IMPLIED>`
Note:	XSL supports some more value that are not supported by OpenOffice.org XML.

Page Number Format

You can specify a default number format for page styles, which is used to display page numbers within headers and footers. See Section 2.9 for detailed information on number format attributes.

XML Code:	`style:num-format` `style:num-letter-sync`
Rules:	The `style:num-format` attribute can be empty.
DTD:	`<!ATTLIST style:properties style:num-format CDATA #REQUIRED>` `<!ATTLIST style:properties style:num-letter-sync %boolean; "false">`

Paper Tray

The `style:paper-tray-name` attribute specifies the paper tray to use when printing the document. The names assigned to the printer trays depend on the printer.

XML Code:	`style:paper-tray-name`
Rules:	If the value of this attribute is `default`, the default tray specified in the printer configuration settings is used.
DTD:	`<!ATTLIST style:properties style:paper-tray-name CDATA #IMPLIED>`

Print Orientation

The `style:print-orientation` attribute specifies the orientation of the printed page.

XML Code:	`style:print-orientation`
Rules:	The value of this attribute can be `portrait` or `landscape`.
DTD:	`<!ATTLIST style:properties style:print-orientation (portrait| landscape) #IMPLIED>`

Margins

The margins attributes specify the size of the page margins. See *Paragraph Formatting Properties* in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:margin-top`
	`fo:margin-bottom`
	`fo:margin-left`
	`fo:margin-right`

Border

The border attributes specify the border properties of the page. See *Paragraph Formatting Properties* in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:border`
	`fo:border-top`
	`fo:border-bottom`
	`fo:border-left`
	`fo:border-right`

Note:	XSL does not have border properties for page masters.

Border Line Width

If a page contains borders, the border line width attributes specify the properties of the border lines of the page. See *Paragraph Formatting Properties* in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`style:border-line-width`
	`style:border-line-width-top`
	`style:border-line-width-bottom`
	`style:border-line-width-left`
	`style:border-line-width-right`

Padding

The padding attributes specify the padding properties of the page. See *Paragraph Formatting Properties* in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:padding`
	`fo:padding-top`
	`fo:padding-bottom`
	`fo:padding-left`
	`fo:padding-right`

Shadow

See *Paragraph Formatting Properties* in Chapter 3 of this manual for detailed information on this attribute.

XML Code:	`style:shadow`

Background

The background attributes specify the background properties of the page. See *Paragraph Formatting Properties* in Chapter 3 of this manual for detailed information on these attributes.

XML Code:	`fo:background-color` and `<style:background-image>`
Note:	XSL does not have background properties for page masters.

Columns

The `<style:columns>` element specifies if the page contains columns. See *Paragraph Formatting Properties* in Chapter 3 of this manual for detailed information on this attribute.

XML Code:	`<style:columns>`

Register-truth

The `style:register-truth-ref-style-name` attribute references a paragraph style. The line distance specified of the paragraph style is used as the reference line distance for all paragraphs that have the register-truth feature enabled.

XML Code:	`style:register-truth-ref-style-name`
Rules:	
DTD:	`<!ATTLIST style:properties style:register-truth-ref-style-name %styleName #IMPLED>`

Print

The `style:print` attribute specifies which components in a spreadsheet document to print.

XML Code:	`style:print`
Rules:	The value of this attribute is a list of the following values separated by blanks: • Headers • Grid • Annotations • Objects (graphics or OLE Objects) • Charts • Drawings • Formulas • Zero values
DTD:	`<!ATTLIST style:properties style:print CDATA #IMPLED>`

Print Page Order

The `style:print-page-order` attribute specifies the order in which data in a spreadsheet is numbered and printed when the data does not fit on one printed page.

XML Code:	`style:print-page-order`
Rules:	The value of this attribute can be `ttb` or `ltr`. Use `ttb` to print the data vertically from the left column to the bottom row of the sheet. Use `ltr` to print the data horizontally from the top row to the right column of the sheet.
DTD:	`<!ATTLIST style:properties style:print-page-order ("ttb" | "ltr") #IMPLED>`

First Page Number

The `style:first-page-number` attribute allows you to specify a number other than 1 for the first page.

XML Code:	`style:first-page-number`
Rules:	The value of this attribute can be an integer or `continue`. If the value is `continue`, the page number is the last page number incremented by 1.
DTD:	`<!ATTLIST style:properties style:first-page-number % positiveInteger; #IMPLED>`

Scale

The scale attributes specify how the application should scale the document for printing.

XML Code:	`style:scale-to` `style:scale-to-pages`
Rules:	You can use one the above attributes to specify how to scale the document for printing. If none of these attributes are present, the document is not scaled. If you use the `style:scale-to` attribute, the document is scaled to a percentage value. 100 percent is the normal value. You can enlarge or reduce all printed pages using this attribute. If you use the `style:scale-to-pages` attribute, you can specify the number of pages on which the the document should be printed. The document is then scaled to fit the defined number of pages.

DTD:	`<!ATTLIST style:properties style:scale-to %percentage; #IMPLED>`
	
	`<!ATTLIST style:properties style:scale-to-pages % positiveInteger; #IMPLED>`

Table Centering

The `style:table-centering` attribute specifies how the application should center tables on the page. This attribute only applies to spreadsheet documents.

XML Code:	`style:table-centering`
Rules:	The value of this attribute can be `horizontal`, `vertical`, `both`, or `none`. If this attribute is not present, the table is not centred.
DTD:	`<!ATTLIST style:properties style:table-centering (horizontal, vertical, both, none) #IMPLIED>`

Maximum Footnote Height

The `style:footnote-max-height` attribute specifies the maximum amount of space on the page that a footnote can occupy. The value of the attribute is a length, which determines the maximum height of the footnote area.

XML Code:	`style:footnote-max-height`
Rules:	If the value of this attribute is set to `0`, there is no limit to the amount of space that the footnote can occupy.
DTD:	`<!ATTLIST style:properties style:footnote-max-height %length; #IMPLIED>`

Footnote Separator

The `<style:footnote-sep>` element describes the line that separates the footnote area from the body text area on a page.

The `<style:footnote-sep>` element supports the following attributes:

- `style:width` – specifies the width or thickness of the line.
- `style:rel-width` – specifies the length of the line as a percentage of the body text area.

- `style:color` – specifies the color of the line.
- `style:adjustment` – specifies how the line is aligned on the page, that is left, right, or center.
- `style:distance-before-sep` – specifies the space between the body text area and the footnote line.
- `style:distance-after-sep` – specifies the space between the footnote line and the footnote text.

XML Code:	`<style:footnote-sep>`
Rules:	

DTD:	`<!ELEMENT style:footnote-sep EMPTY>`
	`<!ATTLIST style:footnote-sep style:width %length; #IMPLIED>`
	`<!ATTLIST style:footnote-sep style:rel-width %percentage; #IMPLIED>`
	`<!ATTLIST style:footnote-sep style:color %color; #IMPLIED>`
	`<!ATTLIST style:footnote-sep style:adjustment (left|center|right) "left">`
	`<!ATTLIST style:footnote-sep style:distance-before-sep %length; #IMPLIED>`
	`<!ATTLIST style:footnote-sep style:distance-after-sep %length; #IMPLIED>`

2.3.2 Master Pages

This section of the manual describes the master page features that are supported by text and spreadsheets documents. The master pages used in drawings and presentations have some additional features that are described in section .

Master pages are contained within a master style element.

XML Code:	`<style:master-page>`
Rules:	In text and spreadsheet documents, this element contains the content of headers and footers. In drawings and presentations it contains background shapes.

DTD:	`<!ELEMENT style:master-page`
	`( (style:header, style:header-left?)?,`
	`(style:footer, style:footer-left?)?,`
	`style:style*, (%shapes;)*, presentation:notes? )>`

The attributes that you can associate with the `<style:master-page>` attribute are:

- Page name
- Page master
- Next style name

Page Name

The `styles:name` attribute specifies the name of the master page. Each master page is referenced using its page name.

XML Code:	`styles:name`
Rules:	A page name is required for each master page and the name must be unique.
DTD:	`<!ATTLIST style:master-page style:name %styleName; #REQUIRED>`

Page Master

The `style:page-master-name` attribute specifies page master which contains the size, border, and orientation of the master page.

XML Code:	`style:page-master-name`
Rules:	This attribute is required for each master page.
DTD:	`<!ATTLIST style:master-page style:page-master-name % styleName; #REQUIRED>`

Next Style Name

The `style:next-style-name` attribute can be used to specify the master page used for the next page, if there is a next page.

XML Code:	`style:next-style-name`
Rules:	If the next style name is not specified, the current master page is used for the next page.
DTD:	`<!ATTLIST style:master-page style:next-style-name %styleName #IMPLIED>`
Note:	This attribute replaces the XSL page-sequence-master concept.

2.3.3 Headers and Footers

The header and footer elements specify the content of headers and footers. The `<style:header>` and `<style:footer>` elements contain the content of headers and footers. The two additional elements, `<style:header-left>` and `<style:footer-left>`, can be used to specify different content for left pages, if appropriate. If the latter two elements are missing, the content of the headers and footers on left and right pages is the same.

These elements are contained within a master page element.

XML Code:	`<style:header>`
	`<style:footer>`
	`<style:header-left>`
	`<style:footer-left>`

Rules:	If the `style:page-usage` attribute associated with the page master has a value of `all` or `mirrored` and there are no `<style:header-left>` or `<style:footer-left>` elements, the header and footer content is the same for left and right pages. If the `style:page-usage` attribute has a value of `left` or `right`, the `<style:header-left>` or `<style:footer-left>` elements are ignored. The content of headers and footers is either: • Standard text content, for example paragraphs, tables, or lists • A sequence of any of the following elements; `<style:region-left>`, `<style:region-center>` and `<style:region-reight>` • Empty, which switches off the display of all headers or footers. It is not possible to switch off the display of headers or footers for left pages only.

DTD:	`<!ENTITY %hd-ft-content`
	`"( %text; |`
	`(style:region-left?|style:region-center?|`
	`style:region-right?) )">`
	`<!ELEMENT style:header %hd-ft-content;>`
	`<!ELEMENT style:footer %hd-ft-content;>`
	`<!ELEMENT style:header-left %hd-ft-content;>`
	`<!ELEMENT style:footer-left %hd-ft-content;>`

Implementation limitation:	OpenOffice.org Writer only supports headers and footers that contain normal text, while OpenOffice.org Calc only supports headers and footers that use the region elements.

2.3.4 Header and Footer Styles

The header and footer style elements specify the formatting properties for headers and footers on a page.

XML Code:	`<style:header-style>` and `<style:footer-style>`
Rules:	These elements must be contained within a page master element.
DTD:	`<!ELEMENT style:header-style (style:properties?)>` `<!ELEMENT style:footer-style (style:properties?)>`

The attributes that you can associate with the header and footer elements are contained within a `<style:properties>` element. These attributes are:

- Fixed and minimum heights - see Section 2.6.11
- Left and right margins - see Section 2.6.11
- Bottom (for headers only) and top (for footers only) margins - see Section 2.6.11
- Borders - see Section 3.11.27 and 3.11.28
- Shadows – see Se ction 3.11.30
- Backgrounds – see Section 3.11.25 and 3.11.26

2.3.5 Footnote Layout

The <style:footnote-layout> element specifies the layout for footnotes that are contained on a page.

XML Code:	<style:footnote-layout>
Rules:	This element must be contained in the page master element. It contains a <style:properties> element that specifies the maximum height and spacing of the footnote area and a <style:footnote-sep> element that specifies the separator line between the page body area and the footnote area. If this element is not present, a default footnote layout is used.
DTD:	<!ELEMENT style:footnote-layout (style:properties?,style:footnote-sep?)>

The attributes that you can associate with the <style:footnote-layout> element in the <style:properties> element are:

- Maximum height
- Spacing

Maximum Height

The style:max-height attribute specifies the maximum height of the footnote area.

XML Code:	style:max-height
Rules:	This attribute supports a value of no-limit, which allows the footnote area to increase until it equals the height of the page height.
DTD:	<!ENTITY % lengthOrNoLimit "CDATA"> <!ATTLIST style:properties style:max-height %lengthOrNoLimit #IMPLED>

Spacing

The spacing attributes specify the distances before and after the line that separates the footnote area from the page body area.

XML Code:	style:distance-before-sep style:distance-after-sep
Rules:	These attributes are valid even if there is no footnote separator line specified.
DTD:	<!ATTLIST style:properties style:distance-before-sep %length #IMPLED> <!ATTLIST style:properties style:distance-after-sep %length #IMPLED>

2.3.6 Footnote Separator Line

The <style:footnote-sep> element specifies the separator line to use between the page body area and the footnote area.

XML Code:	`<style:footnote-sep>`
Rules:	This element can be contained in a `<style:footnote-layout>` element.
DTD:	`<!ELEMENT style:footnote-sep EMPTY>`

The attributes that you can associate with the `<style:footnote-sep>` element are:

- Line width
- Line length
- Horizontal line alignment

Line Width

The `style:width` attribute specifies the width of the separator line.

XML Code:	`style:width`
Rules:	
DTD:	`<!ATTLIST style:footnote-sep style:width %length; #REQUIRED>`

Line Length

The `style:length` attribute specifies the length of the separator line.

XML Code:	`style:length`
Rules:	The value of this attribute is a percentage that relates to the width of the page excluding the page margins.
DTD:	`<!ATTLIST style:footnote-sep style:length %percentage; "100%">`

Horizontal Line Alignment

The `style:horizontal-align` attribute specifies how to horizontally align a line that is less than 100% long.

XML Code:	`style:horizontal-align`
Rules:	The value of this attribute can be `left`, `center`, or right
DTD:	`<!ATTLIST style:footnote-sep style:horizontal-align (left|center|right) "left">`

2.4 Font Declarations

In XSL and CSS, a font is described by its font family. The OpenOffice.org XML file format also uses an additional set of attributes to describe a font. These additional attributes are evaluated if the font specified in the font family is not available, enabling the application to choose an alternative font. The additional attributes are:

- Style name
- Generic family

- Font pitch
- Character set

If a font is referenced, for example in a style, the additional font attributes can be either specified with the font family or using the `<office:font-decls>` element. A font declaration assigns a unique name to a set of font attributes. Font declarations help to reduce file sizes.

XML Code:	`<office:font-decls>`
Rules:	This element is a container for all font declarations. It must appear before any style container element.
DTD:	`<!ELEMENT office:font-decls (style:font-decl)*>`

2.4.1 Font Declaration

A font declaration assigns a set of font formatting properties to a unique font name

XML Code:	`<style:font-decl>`
Rules:	
DTD:	`<!ELEMENT style:font-decl EMPTY>`

The attributes that you associate with a `<style:font-decl>` element are:

- Font name
- Font properties

Font Name

The `style:name` attribute specifies the unique name of the font.

XML Code:	`style:name`
Rules:	
DTD:	`<!ENTITY % fontName "CDATA"` `<!ATTLIST style:font-decl style:name %fontName; #REQUIRED>`

Font Properties

The following font properties can be used to specify a font:

- Font family
- Font style name
- Generic font family
- Font pitch
- Font charset

The font family is required for every font declaration. All other properties are optional. See *Text Formatting Properties* in Chapter 3 of this manual for more information on these font properties.

XML Code:	`fo:font-family`
	`style:font-style-name`
	`style:font-family-generic`
	`style:font-pitch`
	`style:font-charset`

Rules:	

DTD:	`<!ATTLIST style:font-decl fo:font-family %string; #REQUIRED>`
	`<!ATTLIST style:font-decl style:font-style-name %string; #IMPLIED>`
	`<!ENTITY % fontFamilyGeneric "(roman|swiss|modern|decorative|script|system)">`
	`<!ATTLIST style:font-decl style:font-family-generic %fontFamilyGeneric; #IMPLIED>`
	`<!ENTITY % fontPitch "(fixed|variable)">`
	`<!ATTLIST style:font-decl style:font-pitch %fontPitch; #IMPLIED>`
	`<!ATTLIST style:font-decl style:font-charset CDATA #IMPLIED>`

2.5 Data Styles

Data styles describe how to display different types of data, for example, a number or a date. The elements and attributes that are used to represent data styles are contained in the namespace `http://openoffice.org/2000/datastyle`. The prefix `number` denotes the data styles namespace.

This section describes the OpenOffice.org XML representation of the following data styles:

- Number style
- Currency style
- Percentage style
- Date style
- Boolean style
- Text style

2.5.1 Number Style

The `<number:number-style>` element describes the style for decimal numbers.

XML Code:	`<number:number-style>`
Rules:	This element can contain *one* of the following elements: • `<number:number>` • `<number:scientific-number>` • `<number:fraction>` These elements describe the display format of the number. The elements can be preceded or followed by `<number:text>` elements, which contain any additional text to be displayed before or after the number. In addition, this element can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % any-number "( number:number | number:scientific-number | number:fraction )">`
	`<!ENTITY % number-style-content "( number:text | (number:text?,%any-number;,number:text?) )">`
	`<!ELEMENT number:number-style ( style:properties?, %number-style-content;, style:map? )>`

You can use the following elements with the `<number:number-style>` element:

- Number
- Scientific number
- Fraction

Number

The `<number:number>` element specifies the display properties for a decimal number.

XML Code:	`<number:number>`
Rules:	This element is contained in the `<number:number-style>` element. The `<number:number>` element can contain multiple `<number:embedded-text>` elements.
DTD:	`<!ELEMENT number:number ( number:embedded-text* )>`

See Section 2.5.10 for information about the attributes that you can associate with the number style elements.

Embedded Text

The `<number:embedded-text>` element specifies text that is displayed at one specific position within a number. This element is different to a grouping separator, which appears several times within a number.

XML Code:	`<number:embedded-text>`
Rules:	This element is contained in the `<number:number>` element. The `<number:number>` element can contain multiple occurrences of the `<number:embedded-text>` element to describe text at different positions in the number.
DTD:	`<!ELEMENT number:embedded-text (#PCDATA)>`

The `number:position` attribute specifies the position where the text appears.

Position Attribute

XML Code:	number:position
Rules:	The position is counted from right to left, from before the decimal point if one exists, or else from the end of the number. For example, position number 1 indicates that the text is inserted before the last digit. Position number 2 indicates that the text is inserted before the second last digit, and so on.
DTD:	<!ATTLIST number:embedded-text number:position %integer; #REQUIRED>

Scientific Number

The <number:scientific-number> element specifies the display properties for a number style that should be displayed in scientific format.

XML Code:	<number:scientific-number>
Rules:	This element is contained in the <number:number-style> element.
DTD:	<!ELEMENT number:scientific-number EMPTY>

See Section 2.5.10 for information on the attributes that you can associate with the number style elements.

Fraction

The <number:fraction> element specifies the display properties for a number style that should be displayed as a fraction.

XML Code:	<number:fraction>
Rules:	This element is contained in the <number:number-style> element.
DTD:	<!ELEMENT number:fraction EMPTY>

See Section 2.5.10 for information on the attributes that you can associate with the number style elements.

2.5.2 Currency Style

The <number:currency-style> element describes the style for currency values.

XML Code:	`<number:currency-style>`
Rules:	This element can contain one `<number:number>` element and one `<number:currency-symbol>` element. It can also contain `<number:text>` elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, this element can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % currency-symbol-and-text "number:currency-symbol, number:text?">`
	`<!ENTITY % number-and-text "number:number,number:text?">`
	`<!ENTITY % currency-style-content`
	`"( number:text |`
	`(number:text?,%number-and-text;,(currency-symbol-and-text)?) |`
	`(number:text?,%currency-symbol-and-text;,(number-and-text)?)`
	`)">`
	`<!ELEMENT number:currency-style ( style:properties?, %`
	`currency-style-content;, style:map? )>`

You can use the following elements with the `<number:currency-style>` element:

- Number, see Section 2.5.1.
- Currency symbol

Currency Symbol

The `<number:currency-symbol>` element determines whether or not a currency symbol is displayed in a currency style.

XML Code:	`<number:currency-symbol>`
Rules:	The content of this element is the text that is displayed as the currency symbol. If the element is empty or contains white space characters only, the default currency symbol for the currency style or the language and country of the currency style is displayed. This element is contained in the `<number:currency-style>` element.
DTD:	`<!ELEMENT number:currency-symbol (#PCDATA)>`

If the currency symbol contained in a currency style belongs to a different language or country to that of the currency style, you can use the currency language and country attributes to specify the language and country of the currency symbol.

Currency Language and Country Attributes

XML Code:	`number:language` `number:country`
Rules:	
DTD:	`<!ATTLIST number:currency-symbol number:language CDATA #IMPLIED>` `<!ATTLIST number:currency-symbol number:country CDATA #IMPLIED>`

See Section 2.5.9 for information on the other attributes that you can associate with the currency style elements.

2.5.3 Percentage Style

The <number:percentage-style> element describes the style for percentage values.

XML Code:	<number:percentage-style>
Rules:	This element can contain one <number:number> element, which describes the display format for the percentage. The element can be preceded or followed by <number:text> elements, which contain any additional text to display before or after the percentage. In addition, the <number:percentage-style> element can contain a <style:properties> element and a <style:map> element.
DTD:	<!ENTITY % percentage-style-content "(number:text | (number:text?,%number-and-text;))"> <!ELEMENT number:percentage-style (style:properties?, %percentage-style-content;, style:map?)>
Implementation limitation:	Currently, the OpenOffice.org software requires this element to contain at least one <number:text> element and the text must contain a "%" character.

See Section 2.5.9 for information on the attributes that you can associate with the percentage style element.

2.5.4 Date Style

The <number:date-style> element describes the style for date values.

XML Code:	<number:date-style>
Rules:	This element can contain *one* instance of each of the following elements: <number:day>, <number:month>, <number:year>, <number:era>, <number:day-of-week>, <number:week-of-year>, <number:quarter>, <number:hours>, <number:minutes>, <number:seconds>, and <number:am-pm>. The <number:date-style> element can also contain <number:text> elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, it can contain a <style:properties> element and a <style:map> element.
DTD:	<!ENTITY % any-date "(number:day | number:month | number:year | number:era | number:day-of-week | number:week-of-year | number:quarter | number:hours | number:am-pm | number:minutes | number:seconds)"> <!ENTITY % date-style-content "(number:text | (number:text?, (%any-date;,number:text?)+))"> <!ELEMENT number:date-style (style:properties?, %date-style-content;, style:map?)> **Note:** This DTD does not reflect the fact that some elements must not occur more than once.

See Section 2.5.9 for information on the attributes that you can associate with the date style elements.

The <number:date-style> element can contain the following elements:

- <number:day> – day of month
- <number:month> – month
- <number:year> – year
- <number:era> – era

- <number:day-of-week> – day of week
- <number:week-of-year> – week of year
- <number:quarter> – quarter

Day of Month

The <number:day> element specifies the day of the month in a date.

XML Code:	<number:day>
Rules:	If this element is used, it should be contained in the <number:date-style> element.
DTD:	<!ELEMENT number:day EMPTY>

The format attribute specifies whether the day of month element is displayed in short or long format.

Format Attribute

XML Code:	number:style
Rules:	The value of this attribute can be short or long. The meaning of these values depends on the value of the number:format-source attribute that is attached to the date style. For days, if the value of the number:format-source attribute is fixed: • short means that the day of the month is displayed using one or two digits • long means that the day of the month is displayed using two digits
DTD:	<!ATTLIST number:day number:style (short|long) short>

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Month

The <number:month> element specifies the month in a date.

XML Code:	<number:month>
Rules:	If used, this element must be contained in the <number:date-style> element.
DTD:	<!ELEMENT number:month EMPTY>

The number:textual attribute determines whether the name or number of a month is displayed in the month element of a date.

Textual Representation Attribute

XML Code:	number:textual
Rules:	If the value of this attribute value is true, the name of the month is displayed. If the attribute value is false, the number of the month is displayed.
DTD:	<!ATTLIST number:month number:textual %boolean; "false">

The number:style attribute specifies whether the month element is displayed in short or long format.

Format Attribute

XML Code:	number:style
Rules:	The value of this attribute can be short or long. The meaning of these values depends on the value of the number:format-source attribute that is attached to the date style. For months, if the value of the number:format-source attribute is fixed: • short means that the abbreviated name of the month is displayed or the month is displayed using one or two digits • long means that the full name of the month is displayed or the month is displayed using two digits
DTD:	<!ATTLIST number:month number:style (short|long) short>

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Year

The <number:year> element specifies the year in the date.

XML Code:	<number:year>
Rules:	If used, this element must be contained in the <number:date-style> element.
DTD:	<!ELEMENT number:year EMPTY>

The number:style attribute specifies whether the year element is displayed in short or long format.

Format Attribute

XML Code:	number:style
Rules:	The value of this attribute can be short or long. The meaning of these values depends on the value of the number:format-source attribute that is attached to the date style. For years, if the value of the number:format-source attribute is fixed: • short means that the year is displayed using two digits • long means that the year is displayed using four digits
DTD:	<!ATTLIST number:year number:style (short|long) short>

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Era

The <number:era> element specifies the era in which the year is counted.

XML Code:	<number:era>
Rules:	If used, this element must be contained in the <number:date-style> element.
DTD:	<!ELEMENT number:era EMPTY>

The number:style attribute specifies whether the era element is displayed in short or long format.

Format Attribute

XML Code:	number:style
Rules:	The value of this attribute can be short or long. The meaning of these values depends on the value of the number:format-source attribute that is attached to the date style. For eras, if the value of the number:format-source attribute is fixed: • short means that the abbreviated era name is used • long means that the full era name is used
DTD:	<!ATTLIST number:era number:style (short|long) short>

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Day Of Week

The <number:day-of-week> element specifies the day of the week in a date.

XML Code:	<number:day-of-week>
Rules:	If used, this element must be contained in the <number:date-style> element.
DTD:	<!ELEMENT number:day-of-week EMPTY>

The number:style attribute specifies whether the day of week element is displayed in short or long format.

Format Attribute

XML Code:	number:style
Rules:	The value of this attribute can be short or long. The meaning of these values depends on the value of the number:format-source attribute that is attached to the date style. For days of the week, the value of the number:format-source attribute is fixed: • short means that the abbreviated name of the day is displayed • long means that the full name of the day is displayed
DTD:	<!ATTLIST number:day-of-week number:style (short|long) short>

Week Of Year

The <number:week-of-year> element specifies the week of the year in the date.

XML Code:	<number:week-of-year>
Rules:	If used, this element must be contained in the <number:date-style> element.
DTD:	<!ELEMENT number:week-of-year EMPTY>

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

Quarter

The <number:quarter> element specifies the quarter of the year in the date.

XML Code:	`<number:quarter>`
Rules:	If used, this element must be contained in the `<number:date-style>` element.
DTD:	`<!ELEMENT number:quarter EMPTY>`

The `number:style` attribute specifies whether the quarter element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the date style. For quarters, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the abbreviated name of the quarter is displayed, for example, Q1 • `long` means that the full name of the quarter is displayed, for example, Quarter 1
DTD:	`<!ATTLIST number:quarter-of-year number:style (short|long) short>`

See Section 2.5.9 for information on the other attributes that you can associate with the date style elements.

2.5.5 Time Style

The `<number:time-style>` element describes the style for time values.

XML Code:	`<number:time-style>`
Rules:	This element can contain *one* instance of any of the following elements: `<number:hours>`, `<number:minutes>`, `<number:seconds>` and `<number:am-pm>`. The `<number:time-style>` element can also contain `<number:text>` elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, it can contain a `<style:properties>` element and a `<style:map>` element.

DTD:	`<!ENTITY % any-time "( number:hours | number:am-pm | number:minutes | number:seconds )">`
	`<!ENTITY % time-style-content "( number:text | (number:text?, (%any-time;,number:text?)+) )">`
	`<!ELEMENT number:time-style ( style:properties?, %time-style-content;, style:map? )>`
	
	Note: This DTD does not reflect the fact that some elements must not occur more than once.

See Section 2.5.9 for information on the attributes that you can associate with the time style elements.

The following elements can be contained in the `<number:time-style>` element:

- `<number:hours>` – hours
- `<number:minutes>` – minutes
- `<number:seconds>` – seconds
- `<number:am-pm>` – am/pm

Hours

The <number:hours> element specifies if hours are displayed as part of a date or time.

XML Code:	<number:hours>
Rules:	
DTD:	<!ELEMENT number:hours EMPTY>

The number:style attribute specifies whether the hours element is displayed in short or long format.

Format Attribute

XML Code:	number:style
Rules:	The value of this attribute can be short or long. The meaning of these values depends on the value of the number:format-source attribute that is attached to the time style. For hours, if the value of the number:format-source attribute is fixed: • short means that the hours are displayed using at least one digit • long means that the hours are displayed using at least two digits
DTD:	<!ATTLIST number:hours number:style (short|long) short>

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

Minutes

The <number:minutes> element specifies if minutes are displayed as part of a date or time.

XML Code:	<number:minutes>
Rules:	
DTD:	<!ELEMENT number:minutes EMPTY>

The number:style attribute specifies whether the minutes element is displayed in short or long format.

Format Attribute

XML Code:	number:style
Rules:	The value of this attribute can be short or long. The meaning of these values depends on the value of the number:format-source attribute that is attached to the time style. For minutes, if the value of the number:format-source attribute is fixed: • short means that the minutes are displayed using at least one digit • long means that the minutes are displayed using at least two digits
DTD:	<!ATTLIST number:minutes number:style (short|long) short>

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

Seconds

The <number:seconds> element specifies if seconds are displayed as part of a date or time.

XML Code:	`<number:seconds>`
Rules:	
DTD:	`<!ELEMENT number:seconds EMPTY>`

The `number:style` attribute specifies whether the seconds element is displayed in short or long format.

Format Attribute

XML Code:	`number:style`
Rules:	The value of this attribute can be `short` or `long`. The meaning of these values depends on the value of the `number:format-source` attribute that is attached to the time style. For seconds, if the value of the `number:format-source` attribute is `fixed`: • `short` means that the seconds are displayed using at least one digit • `long` means that the seconds are displayed using at least two digits
DTD:	`<!ATTLIST number:seconds number:style (short|long) short>`

When you are displaying fractions, you can include fractions of seconds. The `number:decimal-places` attribute determines the number of decimal places to use when displaying fractions.

Decimal Places Attribute

XML Code:	`number:decimal-places`
Rules:	If this attribute is not used or if the value of the attribute is `0`, fractions are not displayed.
DTD:	`<!ATTLIST number:seconds number:decimal-places %number; "0">`

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

AM/PM

The `<number:am-pm>` element specifies if AM/PM is included as part of the date or time.

XML Code:	`<number:am-pm>`
Rules:	If a `<number:am-pm>` element is contained in a date or time style, hours are displayed using values from `1` to `12` only.
DTD:	`<!ELEMENT number:am-pm EMPTY>`

See Section 2.5.9 for information on the other attributes that you can associate with the time style elements.

2.5.6 Boolean Style

The `<number:boolean-style>` element describes the style for Boolean values.

XML Code:	<number:boolean-style>
Rules:	This element can contain one <number:boolean> element, which can be preceded or followed by <number:text> elements. In addition, it can contain a <style:properties> element and a <style:map> element.
DTD:	<!ENTITY % boolean-style-content "(number:text| (number:text?, number:boolean,number:text?))"> <!ELEMENT number:boolean-style (style:properties?,%boolean-style-content;, style:map?)>

Boolean

The <number:boolean> element contains the Boolean value of a Boolean style.

XML Code:	<number:boolean>
Rules:	
DTD:	<!ELEMENT number:boolean EMPTY>

See Section 2.5.9 for information on the attributes that you can associate with the Boolean style elements.

2.5.7 Text Style

The <number:text-style> element describes the style for displaying text.

XML Code:	<number:text-style>
Rules:	This element can contain any number of <number:text-content> elements. It can also contain <number:text> elements , which display additional text, but it cannot contain two of these elements consecutively. In addition, it can contain a <style:properties> element and a <style:map> element.
DTD:	<!ENTITY % text-style-content "(number:text| (number:text?, number:text-content, number:text?))"> <!ELEMENT number:text-style (style:properties?,%text-style-content;, style:map?)>
Notes:	The <number:text-content> elements represent the variable text content to display, while the <number:text> elements contain any additional fixed text to display.

See Section 2.5.9 for information on the attributes that you can associate with the text style elements.

Fixed Text

The <number:text> element contains any fixed text for a data style.

XML Code:	<number:text>
Rules:	This element is contained in the data styles element.
DTD:	<!ELEMENT number:text (#PCDATA)>

Text Content

The <number:text-content> element contains the text content of a text style.

XML Code:	<number:text-content>
Rules:	
DTD:	<!ELEMENT number:text-content EMPTY>

2.5.8 Common Data Style Elements

You can use some style elements with any of the primary data style elements. These elements are:

- Formatting properties
- Style mappings

Formatting Properties

The <style:properties> element specifies the text formatting properties to apply to any text displayed in the data style. See Section 2.2.1 for information on the formatting properties element.

Style Mappings

The <style:map> element specifies an alternative data style to map to if a certain condition exists. See Section 2.2.5 for information on the <style:map> element.

Rules for using this element with data style elements:	• This element must be the last child element in the data style element. • The style referenced by the style:apply-style attribute must be of the same type as the style containing the map. • The condition must be in the format value() *op n*, where *op* is a relational operator and *n* is a number. For Boolean styles the condition value must be true and false.

2.5.9 Common Data Style Attributes

Many of the data style attributes are applicable to more than one data style element. The following data style attributes are common to many of the data style elements:

- Name
- Language
- Country
- Title
- Volatility
- Automatic Order
- Format Source

- Time Value Truncation
- Transliteration

Name

The `style:name` attribute specifies the name of the data style. It can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`style:name`
Rules:	
DTD:	`<!ATTLIST number:number-style style:name %style-name; #REQUIRED>`

Language

The `number:language` attribute specifies the language of the style. The value of the attribute is a language code conforming with ISO639. OpenOffice.org XML uses the language code to retrieve information about any display properties that are language-dependent. The language attribute can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`number:language`
Rules:	If a language code is not specified, either the system settings or the setting for the system's language are used, depending on the property whose value should be retrieved.
DTD:	`<!ATTLIST number:number-style number:language CDATA #IMPLIED>`

Country

The `number:country` attribute specifies the country of the style. The value of the attribute is a country code, conforming with ISO3166. OpenOffice.org XML uses the country code to retrieve information about any display properties that are country-dependent. The language attribute can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`number:country`
Rules:	If a country is not specified, either the system settings or the setting for the system's country are used, depending on the property whose value should be retrieved.
DTD:	`<!ATTLIST number:number-style number:country CDATA #IMPLIED>`

Title

The `number:title` attribute specifies the title of the data style. It can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

XML Code:	`number:title`
Rules:	
DTD:	`<!ATTLIST number:number-style number:title CDATA #IMPLIED>`

Volatility

Sometimes when a document is opened, not all of the styles are used. The unused styles can be retained or discarded; depending on the application you are using. The `style:volatile` attribute allows you to specify what to do with the unused styles. The volatility attribute can be used with any of the following data style elements:

- `<number:number-style>`

- <number:currency-style>
- <number:percentage-style>
- <number:date-style>
- <number:time-style>
- <number:boolean-style>
- <number:text-style>

XML Code:	style:volatile
Rules:	If the value of the attribute is true, the application keeps the style if possible. If the value is false, the application discards the unused styles.
DTD:	<!ATTLIST number:number-style style:volatile %boolean; #IMPLIED>
Note:	If a style is contained in a <style:styles> element, the default value of the style:volatile attribute is true. If a style is contained in a <style:automatic-styles> element, the default value is false.

Automatic Order

The number:automatic-order attribute can be used to automatically order data to match the default order for the language and country of the data style. This attribute is used with the following elements:

- <number:currency-style>, where number and currency symbols are reordered
- <number:date-style>, where the <number:date-style> child elements that are not <number:text> or <style:properties> elements are reordered

XML Code:	number:automatic-order
Rules:	The attribute value can be true or false.
DTD:	<!ATTLIST number:currency-style number:automatic-order %boolean; "false"> **or** <!ATTLIST number:date-style number:automatic-order %boolean; "false">
Note:	If automatic ordering is enabled, but the language and country are not specified, the system settings for the order of numbers and currency symbols is used.

Format Source

The number:format-source attribute specifies the source of the short and long display formats. It is used with the following elements:

- <number:date-style>
- <number:time-style>

XML Code:	`number:format-source`
Rules:	The value of this attribute can be `fixed` or `language`. If the value is `fixed`, the application determines the value of `short` and `long`. If the value is `language`, the value of `short` and `long` is taken from the language and country of the style.
DTD:	`<!ATTLIST number:date-style number:format-source (fixed| language) fixed>`
Notes:	If the value of the `number:format-source` attribute is `language`, the meaning of short and short depends on the language and country of the date style, or, if neither of these are specified, OpenOffice.org XML uses the system settings for short and long date and time formats.

Time Value Truncation

The `number:truncate-on-overflow` attribute is used with the `<number:time-style>` element. If a time or duration is too large to display using the default value range for a time component, (0 to 23 for `<number:hours>`), you can use the time value truncation attribute to specify if it can be truncated or the value range extended.

XML Code:	`number:truncate-on-overflow`
Rules:	
DTD:	`<!ATTLIST number:time-style number:truncate-on-overflow % boolean; "true">`

Transliteration

The number:transliteration-* attributes specify the native number system of the style to display the number using, for example, CJK number characters. The notation is inspired by the W3C XSLT 2.0 draft, see "Number to String Conversion Attributes" [http://www.w3.org/TR/xslt20/#convert]. However, to be able to fully distinguish between all possible native number systems additional attributes are needed in combination. For example, Korean uses 11 different systems where the digits are not always different but short and long and formal and informal forms exist.

The transliteration attributes can be used with the following data style elements:

- `<number:number-style>`
- `<number:currency-style>`
- `<number:percentage-style>`
- `<number:date-style>`
- `<number:time-style>`
- `<number:boolean-style>`
- `<number:text-style>`

Transliteration Format

The `number:transliteration-format` attribute specifies which number characters to use. The value of the attribute is the digit "1" expressed as a native number.

XML Code:	`number:transliteration-format`
Rules:	If no format is specified the default ASCII representation of Arabic digits is used, other transliteration attributes present in this case are ignored.
DTD:	`<!ATTLIST number:number-style number:transliteration-format CDATA "1">`

Transliteration Language

The `number:transliteration-language` attribute specifies which language the native number system belongs to. The value of the attribute is a language code conforming with ISO639.

XML Code:	`number:transliteration-language`
Rules:	If no language/country (locale) combination is specified the locale of the data style is used.
DTD:	`<!ATTLIST number:number-style number:transliteration-language CDATA #IMPLIED>`

Transliteration Country

The `number:transliteration-country` attribute specifies which country the native number system belongs to . The value of the attribute is a country code conforming with ISO3166.

XML Code:	`number:transliteration-country`
Rules:	If no language/country (locale) combination is specified the locale of the data style is used.
DTD:	`<!ATTLIST number:number-style number:transliteration-country CDATA #IMPLIED>`

Transliteration Style

The `number:transliteration-style` attribute specifies which style the native number system belongs to. If more than one native number system matches the transliteration-format this attribute selects one.

XML Code:	`number:transliteration-style`
Rules:	A short style should result in a one to one mapping of Arabic digits to native number digits if possible.
DTD:	`<!ATTLIST number:number-style number:transliteration-style (short|medium|long) "short">`

2.5.10 Common Number Style Attributes

Many of the number style attributes are applicable to more than one number style element. The following attributes are common to many of the number style elements:

- Decimal places
- Minimum integer digits
- Grouping separator
- Decimal replacement

- Minimum exponent digits
- Minimum numerator digits
- Minimum denominator digits
- Calendar system

Decimal Places

The `number:decimal-places` attribute specifies the number of decimal places to display. You can use this attribute with the following elements:

- `<number:number>`
- `<number:scientific-number>`

XML Code:	`number:decimal-places`
Rules:	If this attribute is not specified, a default number of decimal places is used.
DTD:	`<!ATTLIST number:number number:decimal-places %number; #IMPLIED>`

Minimum Integer Digits

The `number:min-integer-digits` attribute specifies the minimum number of integer digits to display in a number, a scientific number, or a fraction. You can use this attribute with the following elements:

- `<number:number>`
- `<number:scientific-number>`
- `<number:fraction>`

XML Code:	`number:min-integer-digits`
Rules:	If this attribute is not specified, a default number of integer digits is used.
DTD:	`<!ATTLIST number:number number:min-integer-digits %number; #IMPLIED>`

Grouping Separator

The `number:grouping` attribute specifies whether or not the integer digits of a number should be grouped using a separator character. You can use this attribute with the following elements:

- `<number:number>`
- `<number:scientific-number>`
- `<number:fraction>`

XML Code:	`number:grouping`
Rules:	The grouping character that is used and the number of digits that are grouped together depends on the language and country of the style.
DTD:	`<!ATTLIST number:number number:grouping %boolean; "false">`

Decimal Replacement

If a number style specifies that decimal places are used but the number displayed is an integer, you can display replacement text instead of the decimal places. The `number:decimal-replacement` attribute specifies the replacement text. You can use this attribute with the `<number:number>` element.

XML Code:	`number:decimal-replacement`
Rules:	
DTD:	`<!ATTLIST number:number number:decimal-replacement CDATA #IMPLIED>`
Implementation limitation:	Currently, OpenOffice.org only supports replacement text that consists of the same number of "-" characters as decimal places.

Display Factor

The `number:display-factor` attribute specifies a factor by which each number is scaled (divided) before displaying. A factor of 1000, for example, causes numbers to be displayed in thousands. You can use this attribute with the `<number:number>` element.

XML Code:	`number:display-factor`
Rules:	
DTD:	`<!ATTLIST number:number number:display-factor %float; "1">`
Implementation limitation:	Currently, OpenOffice.org only supports display factors of 1000 to the power of a non-negative integer number, that is 1, 1000, 1000000, 1000000000, etc.

Minimum Exponent Digits

The `number:min-exponent-digits` attribute specifies the minimum number of digits to use to display an exponent. You can use this attribute with the `<number:scientific-number>` element.

XML Code:	`number:min-exponent-digits`
Rules:	
DTD:	`<!ATTLIST number:scientific-number number:min-exponent-digits %number; #IMPLIED>`

Minimum Numerator Digits

The `number:min-numerator-digits` attribute specifies the minimum number of digits to use to display the numerator in a fraction. You can use this attribute with the `<number:fraction>` element.

XML Code:	`number:min-numerator-digits`
Rules:	
DTD:	`<!ATTLIST number:fraction number:min-numerator-digits %number; #IMPLIED>`

Minimum Denominator Digits

The `number:min-denominator-digits` attribute specifies the minimum number of digits to use to display the denominator of a fraction. You can use this attribute with the `<number:fraction>` element.

XML Code:	`number:min-denominator-digits`
Rules:	
DTD:	`<!ATTLIST number:fraction number:min-denominator-digits % number; #IMPLIED>`

Calendar System

The `number:calendar` attribute specifies the calendar system used to extract parts of a date. You can use this attribute with the following elements:

- `<number:day>`
- `<number:month>`
- `<number:year>`
- `<number:era>`
- `<number:day-of-week>`
- `<number:week-of-year>`
- `<number:quarter>`

XML Code:	`number:calendar`
Rules:	If this attribute is not specified, the default calendar system is used.
DTD:	`<!ATTLIST number:day number:calendar CDATA #IMPLIED>`

2.6 Frames

A **frame** is a rectangular container where you can place content that you want to position outside the default text flow of a document. In OpenOffice.org documents, frames can contain:

- Images
- Drawings
- Text boxes (OpenOffice.org Writer documents only)
- Applets
- Floating frames
- Plug-ins
- OpenOffice.org objects and common OLE objects

A frame has properties that apply to:

- The area around the frame or the frame neighborhood, for example, the anchor type, position or wrap mode.
- The frame content only, for example, the URL for a picture.

- Both frame neighborhood and content, for example, the frame size.

OpenOffice.org XML does not differentiate between the different types of frame properties. Frame formatting properties are stored in an automatically generated style belonging to the `graphics` family. The way a frame is contained in a document depends on the file format type and is explained in the application-specific chapters of this document.

There are several elements used to represent the different frame types. In this manual, these elements are called **frame elements** .This section describes the following frame types and the elements used to represent them:

- Text Boxes
- Images
- Drawings
- Controls
- Plug-ins, applets, and floating frames
- Objects

2.6.1 Text Boxes

You can use a text box to place text in a container that is outside of the normal flow of the document.

XML Code:	`<draw:text-box>`
Rules:	
DTD:	`<!ELEMENT draw:text-box (%frame;*,%text;)>`
Note:	SVG does not support text boxes, while XSL has limited support for text boxes.

The attributes that you can associate with the `<draw:text-box>` element are:

- Name
- Style
- Chain
- Position, size, and transformation (see Chapter 5)
- Layer ID (see Section 2.6.10)
- Z Index (see Section 2.6.10)

Name

The `draw:name` attribute specifies the name of the text box.

XML Code:	`draw:name`
Rules:	
DTD:	`<!ATTLIST draw:text-box text:name CDATA #IMPLIED>`
Implementation limitation:	This attribute is only supported by OpenOffice.org Writer. If a text box without a name is loaded into OpenOffice.org Writer, it is inserted as a drawing text box.

Style

The `draw:style-name` attribute specifies the name of the style for the text box.

XML Code:	`draw:style-name`
Rules:	This attribute links to a `<style:style>` element belonging to the `graphic` style family.
DTD:	`<!ATTLIST draw:text-box draw:style-name &style-name; #REQUIRED>`

Chain

Text boxes can be chained, in other words, if the content of a text box exceeds its capacity, the content flows into the next text box in the chain.

XML Code:	`style:chain-next-name`
Rules:	The value of this attribute is the name of the next text box in the chain.
DTD:	`<!ATTLIST style:properties style:chain-next-name CDATA #IMPLIED>`
Implementation limitation:	Chained text boxes are only supported by OpenOffice.org Writer.

2.6.2 Images

An image can be either:

- Contained in an OpenOffice.org document as a link to an external resource

or

- Embedded in an OpenOffice.org document

XML Code:	`<draw:image>`
Rules:	This element can be an XLink, in which case the element contains some attributes with fixed values that describe the link semantics.
DTD:	`<!ELEMENT draw:image (office:binary-data?,office:events?,draw:image-map?,svg:desc?,(draw:contour-polygon|draw:contour-path)?)>`

The attributes that you can associate with the `<draw:image>` element are:

- Name (see Section 2.6.10)
- Style
- Image data
- Position, size, and transformation (see Chapter 5)
- Filter name
- Layer ID (see Section 2.6.10)
- Z Index (see Section 2.6.10)

You can also use the following elements with the image element:

- Contour (see Section 2.6.9)
- Alternative Text (see Section 2.6.9)

Style

The `draw:style-name` attribute specifies the name of the style for the image.

XML Code:	`draw:style-name`
Rules:	This attribute links to a `<style:style>` element belonging to the `graphic` style family.
DTD:	`<!ATTLIST draw:text-box draw:style-name &style-name; #REQUIRED>`

Image Data

The image data can be stored in one of the following ways:

- The image data is contained in an external file. Use the `xlink:href` and associated attributes described below to link to the external file.
- The image data is contained in the `<draw:image>` element. The `<draw:image>` element contains an `<office:binary-data>` element that contains the image data in BASE64 encoding. In this situation the `xlink:href` attribute is not required.

XML Code:	`xlink:href`, `xlink:type`, `xlink:show`, and `xlink:actuate`
Rules:	

DTD:	`<!ATTLIST draw:image xlink:href %url; #IMPLIED>`
	`<!ATTLIST draw:image xlink:type (simple) #IMPLIED>`
	`<!ATTLIST draw:image xlink:show (embed) 'IMPLIED>`
	`<!ATTLIST draw:image xlink:actuate (onLoad) 'IMPLIED>`

Filter Name

If required, the `draw:filter-name` attribute can represent the filter name of the image.

XML Code:	`draw:filter-name`
Rules:	This attribute contains the internal filter name that the OpenOffice.org software used to load the graphic.
DTD:	`<!ATTLIST draw:image draw:filter-name #IMPLIED>`

2.6.3 Drawings

See Chapter 5 for information on drawing shapes.

2.6.4 Objects

An OpenOffice.org XML document can contain two types of objects, as follows:

- Objects that have an XML representation. These objects are those which the OpenOffice.org software can create and modify, as follows:
  - Formulas created using OpenOffice.org Math
  - Charts created using OpenOffice.org Chart
  - Spreadsheets created using OpenOffice.org Calc
  - Text documents created using OpenOffice.org Writer
  - Drawings created using OpenOffice.org Draw
  - Presentations created using OpenOffice.org Impress
- Objects that do not have an XML representation. These objects only have a binary representation and they are loaded and stored using OLE.

XML Code:	`<draw:object>` `<draw:object-ole>`
Rules:	The `<draw:object>` element represents objects that have a XML representation. The `<draw:object-ole>` element represents objects that only have a binary representation.
DTD:	`<!ELEMENT draw:object ((office:document|math:math)?,office:events?,draw:image-map?,svg:desc?,(draw:contour-polygon|draw:contour-path)?)>` `<!ELEMENT draw:object-ole (office:binary-data?,office:events?,draw:image-map?,svg:desc?,(draw:contour-polygon|draw:contour-path)?,draw:thumbnail?)>`

Object Data

The object data can be called in one of the following ways:

- The `xlink:href` attribute links to the object representation, as follows:
  - For objects that have an XML representation, the link references the subpackage of the OLE object.
  - For objects that do not have an XML representation, the link references a substream of the package that contains the binary representation of the object.
- The object data is contained in the `<draw:object>` or `<draw:object-ole>` element, as follows:
  - The `<draw:object>` element contains the XML representation of the object, for example, an `<office:document>` or a `<math:math>` element.
  - The `<draw:object-ole>` element contains an `<office:binary-data>` element, which contains the binary data for the object in BASE64 encoding.

  In these situations, the `xlink:href` attributes are not required.

XML Code:	`xlink:href`, `xlink:type`, `xlink:show`, and `xlink:actuate`
Rules:	

DTD:	`<!ATTLIST draw:object xlink:href %url; #IMLIED>`
	`<!ATTLIST draw:object xlink:type (simple) #IMPLIED>`
	`<!ATTLIST draw:object xlink:show (embed) #IMPLIED>`
	`<!ATTLIST draw:object xlink:actuate (onLoad) #IMPLIED>`
	`<!ATTLIST draw:object-ole xlink:href %url; #REQUIRED>`
	`<!ATTLIST draw:object-ole xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST draw:object-ole xlink:show (embed) "embed">`
	`<!ATTLIST draw:object-ole xlink:actuate (onLoad) "onLoad">`

Implementation limitation:	The current OpenOffice.org implementation does not support external objects. All objects must be located in the root of the package for the document.

Notification on Table Change

Some objects, especially charts, may require a notification when a table in the document changes. To enable this notification, use the `draw:notify-on-change-of-table` attribute, which contains the name of the table.

XML Code:	`draw:notify-on-change-of-table`
Rules:	This attribute can be associated with the `<draw:object>` element.
DTD:	`<!ATTLIST draw:object draw:notify-on-change-of-table %string; #IMPLIED>`
Note:	This attribute does not enable the automatic update of an OLE object if data within a table changes. The purposeof the attribute is to speed up updates to whole documents by only notifying the OLE objects that have a connection to any table.

2.6.5 Applets

An applet is a small Java-based program that is embedded in a document. The `<draw:applet>` element is based on the <applet> tag in HTML.

XML Code:	`<draw:applet>`
Rules:	This element must contain either the `draw:code` or `draw:object` attribute. The `draw:param*` elements must be located before other content.
DTD:	`<!ELEMENT draw:applet (draw:param*, svg:desc?, (draw:contour-polygon|draw:contour-path)?)>`

The attributes that you can associate with the `<draw:applet>` element are:

- Codebase
- Code
- Object
- Archive
- Name (see Section 2.6.10)
- Style (see Section 2.6.10)
- Position, size, and transformation (see Chapter 5)

- Layer ID (see Section 2.6.10)
- Z Index (see Section 2.6.10)
- Mayscript

You can also use the following elements:

- Alternative Text (see Section 2.6.9)
- Param

Codebase

The `draw:codebase` specifies the base URI for the applet. If this attribute is not specified, then it defaults the same base URI as for the current document. This is represented as an xlink:href.

XML Code:	`xlink:href`, `xlink:type`, `xlink:show`, and `xlink:actuate`
Rules:	

DTD:	`<!ATTLIST draw:applet xlink:href %url; #REQUIRED>`
	`<!ATTLIST draw:applet xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST draw:applet xlink:show (embed) "embed">`
	`<!ATTLIST draw:applet xlink:actuate (onLoad) "onLoad">`

Code

The `draw:code` attribute specifies one of the following:

- The name of the class file that contains the compiled applet subclass.
- The path to the class, including the class file itself.

XML Code:	`draw:code`
Rules:	Either this attribute or the `draw:object` attribute is required. The value of this attribute is interpreted in relation to the codebase for the applet.
DTD:	`<!ATTLIST draw:applet draw:code CDATA #REQUIRED>`

Object

The `draw:object` attribute specifies a resource that contains a serialized representation of the state of the applet. The serialized data contains the class name of the applet but not the implementation.

XML Code:	`draw:object`
Rules:	The value of this attribute is interpreted in relation to the codebase for the applet.
DTD:	`<!ATTLIST draw:applet draw:object CDATA #IMPLIED>`
Note:	This attribute is not used within the OpenOffice.org core and is stored for HTML export.

Archive

The `draw:archive` attribute specifies a comma-separated list of URLs for archives that contain classes and

other resources that are preloaded.

XML Code:	`draw:archive`
Rules:	
DTD:	`<!ATTLIST draw:applet draw:archive CDATA #IMPLIED>`
Note:	This attribute is not used within the OpenOffice.org core and is stored for HTML export.

Mayscript

The `draw:mayscript` attribute specifies whether or not the applet can be scripted.

XML Code:	`draw:mayscript`
Rules:	
DTD:	`<!ATTLIST draw:plugin draw:mayscript %boolean; "false">`

2.6.6 Plugins

A plugin is a binary object that is plugged into a document to represent a media-type that is not handled natively by the OpenOffice.org software.

XML Code:	`<draw:plugin>`
Rules:	This is used to describe plugins in the document<draw:plugin> is used for plugins
DTD:	`<!ELEMENT draw:plugin (draw:param*, svg:desc?, (draw:contour-polygon|draw:contour-path)?)>`

The attributes that you can associate with the `<draw:plugin>` element are:

- Mime type
- Source
- Name (see Section 2.6.10)
- Style (see Section 2.6.10)
- Position, size, and transformation (see Chapter 5)
- Layer ID (see Section 2.6.10)
- Z Index (see Section 2.6.10)

You can also use the following element:

- Alternative Text (see Section 2.6.9)
- Param

Mime type

The `draw:mimetype` attribute specifies the MIME type to which this plugin should be registered.

XML Code:	draw:mimetype
Rules:	
DTD:	<!ATTLIST draw:plugin draw:mimetype CDATA #REQUIRED>

Source

The XLink attributes specify the source of the plugin.

XML Code:	xlink:href, xlink:type, xlink:show, xlink:actuate
Rules:	
DTD:	<!ATTLIST draw:plugin xlink:href %url; #REQUIRED> <!ATTLIST draw:plugin xlink:type (simple) #FIXED "simple"> <!ATTLIST draw:plugin xlink:show (embed) "embed"> <!ATTLIST draw:plugin xlink:actuate (onLoad) "onLoad">

2.6.7 Parameters

The <draw:param> element contains paramaters that are passed to an applet or plugin when they are initialized.

XML Code:	<draw:param>
Rules:	This element contains parameters that are passed to applets and plugins.
DTD:	<!ELEMENT draw:param EMPTY>

The attributes that you can associate with the <draw:param> element are:

- Name (see Section 2.6.10)
- Value

Value

The draw:value specifies the value of a runtime parameter specified by the name.

XML Code:	draw:value
Rules:	
DTD:	<!ATTLIST draw:applet draw:value CDATA #IMPLIED>

2.6.8 Floating Frames

A floating frame is a frame embedded in a document, which may contain, for example, a text document or spreadsheet.

XML Code:	`<draw:floating-frame>`
Rules:	
DTD:	`<!ELEMENT draw:floating-frame (svg:desc?, (draw:contour-polygon|draw:contour-path)?)>`
Note:	We may need to add a `draw:shofloating-frame-border` attribute.

The attributes that you can associate with the `<draw:floating-frame>` element are:

- Name (see Section 2.6.10)
- Source (see Section 2.6.6)
- Style (see Section 2.6.10)
- Position, size, and transformation (see Chapter 5)
- Layer ID (see Section 2.6.10)
- Z Index (see Section 2.6.10)

You can also use the following element:

- Alternative Text (see Section 2.6.9)

2.6.9 Common Frame Elements

The elements contained in this section can be used with several of the frame elements.

Contour

You can use the `<draw:contour-polygon>` and `<draw:contour-path>` elements with the following elements:

- `<draw:image>`
- `<draw:object>`
- `<draw:object-ole>`

XML Code:	`<draw:contour-polygon>` `<draw:contour-path>`
Rules:	These elements describe the contour of an image or object. See Chapter 5 for a description of the attributes associated with the `<draw:contour-polygon>` and `<draw:contour-path>` elements. In contrast to any other element the `svg:width` and `svg:height` attributes may have a pixel length (i.e. 20px) as value (as well as traditional lengths like 2cm).

DTD:	`<!ELEMENT draw:contour-polygon EMPTY>`
	`<!ELEMENT draw:contour-path EMPTY>`
	`<!ATTLIST draw:contour-polygon svg:width %length; #REQUIRED>`
	`<!ATTLIST draw:contour-polygon svg:height %length; #REQUIRED>`
	`<!ATTLIST draw:contour-polygon svg:viewbox CDATA #REQUIRED>`
	`<!ATTLIST draw:contour-polygon svg:points %Points; #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:width %length; #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:height %length; #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:viewbox CDATA #REQUIRED>`
	`<!ATTLIST draw:contour-path svg:d %PathData; #REQUIRED>`

Note:	XSL and SVG do not support contours.

Alternative Text

You can use the `<draw:desc>` element with the following elements:

- `<draw:image>`
- `<draw:object>`
- `<draw:applet>`
- `<draw:floating-frame>`
- `<draw:plugin>`
- `<draw:object-ole>`

XML Code:	`<draw:desc>`
Rules:	
DTD:	`<!ELEMENT draw:desc (#PCDATA)>`
Notes:	This element is the same as the SVG `<desc>` element. SVG does not support a description of foreign objects.

2.6.10 Common Frame Attributes

The attributes contained in this section can be used with several of the frame elements.

Name

You can use the `office:name` attribute with the following elements:

- `<draw:text-box>`

- <draw:image>
- <draw:object>
- <draw:applet>
- <draw:param>
- <draw:plugin>
- <draw:floating-frame>
- <draw:object-ole>

XML Code:	office:name
Rules:	
DTD:	<!ATTLIST draw:image office:name CDATA #IMPLIED>
Note:	SVG does not support names of image or foreign objects.

Style

You can use the style:style attribute with the following elements:

- <draw:object>
- <draw:applet>
- <draw:plugin>
- <draw:floating-frame>
- <draw:object-ole>

XML Code:	style:style
Rules:	
DTD:	<!ATTLIST draw:object style:style CDATA #REQUIRED>
Note:	SVG does not support styles.

Layer ID

You can use the office:layer-id attribute with the following elements:

- <draw:text-box>
- <draw:image>
- <draw:applet>
- <draw:plugin>
- <draw:floating-frame>
- <draw:object>
- <draw:object-ole>

XML Code:	`office:layer-id`
Rules:	
DTD:	`<!ATTLIST draw:text-box office:layer-id %number; #IMPLIED>`
Note:	SVG does not support layers.

Layer ID

You can use the `office:layer-id` attribute with the following elements:

- `<draw:text-box>`
- `<draw:image>`
- `<draw:applet>`
- `<draw:plugin>`
- `<draw:floating-frame>`
- `<draw:object>`
- `<draw:object-ole>`

XML Code:	`office:layer-id`
Rules:	
DTD:	`<!ATTLIST draw:text-box office:layer-id %number; #IMPLIED>`
Note:	SVG does not support layers.

Z Index

You can use the `draw:z-index` attribute with the following elements:

- `<draw:text-box>`
- `<draw:image>`
- `<draw:applet>`
- `<draw:plugin>`
- `<draw:floating-frame>`
- `<draw:object>`
- `<draw:object-ole>`

XML Code:	`draw:z-index`
Rules:	
DTD:	`<!ATTLIST draw:text-box draw:z-index %number; #IMPLIED>`

2.6.11 Frame Formatting Properties

The attributes and elements described in this section can be assigned to a graphic style.

Fixed and Minimum Widths

There are two types of frame widths; fixed widths and minimum widths.

XML Code:	**For fixed widths:**
	`svg:width`
	For minimum widths:
	`fo:min-width`

Rules:	The value of both types of width can be either a length or a percentage. The consequences of assigning both types of width to a frame simultaneously are undefined. If the anchor for the frame is in a table cell, the percentage value relates to the surrounding table box. If the anchor for the frame is in a frame, the percentage value relates to the surrounding frame. In other cases, the percentage values relate to the width of the page or window.
DTD:	`<!ATTLIST style:properties svg:width %length_or_pecentage; #IMPLIED>` `<!ATTLIST style:properties fo:min-width CDATA #IMPLIED>`

Fixed and Minimum Heights

There are two types of frame heights; fixed heights and minimum heights.

XML Code:	**For fixed heights:**
	`svg:height`
	For minimum heights:
	`fo:min-height`

Rules:	The value of both types of height can be either a length or a percentage. The consequences of assigning both types of height to a frame simultaneously are undefined. If the anchor for the frame is in a table cell, the percentage value relates to the surrounding table box. If the anchor for the frame is in a frame, the percentage value relates to the surrounding frame. In other cases, the percentage values relate to the height of the page or window.
DTD:	`<!ATTLIST style:properties fo:height CDATA #IMPLIED>` `<!ATTLIST style:properties fo:min-height CDATA #IMPLIED>`

Maximum Width and Height

For frames that can increase in size automatically when content is added, these attributes specify a maximum width and height for the frame. When the maximum values are reached, the frame stops increasing in size.

XML Code:	`fo:max-width` `fo:max-height`
Rules:	
DTD:	`<!ATTLIST style:properties fo:max-width CDATA #IMPLIED>` `<!ATTLIST style:properties fo:max-height CDATA #IMPLIED>`

Left and Right Margins

These properties determine the left and right margins to set around a frame. The properties are similar to those used to set the margins of a paragraph, as described in Chaper 3.

XML Code:	`fo:margin-left` and `fo:margin-right`
Rules:	The value of these properties must be a length.
DTD:	`<!ATTLIST style:properties fo:margin-left %length; #IMPLIED>` `<!ATTLIST style:properties fo:margin-right %length; #IMPLIED>`
Note:	In contrast to paragraph styles, when these properties are used for graphic styles as is the case for frames, percentage values are not supported.

Top and Bottom Margins

These properties determine the top and bottom margins to set around a frame. The properties are similar to those used to set the margins of a paragraph, as described in Chapter 3.

XML Code:	`fo:margin-top` and `fo:margin-bottom`
Rules:	The value of these properties must be a length.
DTD:	`<!ATTLIST style:properties fo:margin-top %length; #IMPLIED>` `<!ATTLIST style:properties fo:margin-bottom %length; #IMPLIED>`
Note:	In contrast to paragraph styles, when these properties are used for graphic styles as is the case for frames, percentage values are not supported.

Print Content

The `style:print-content` property specifies whether or not you can print the content of a frame.

XML Code:	`style:print-content`
Rules:	
DTD:	`<!ATTLIST style:properties style:print-content %boolean; #IMPLIED>`
Note:	XSL does not support this property.

Protect

The `style:protect` property specifies whether the content, size, or position of a frame is protected.

XML Code:	`style:protect`
Rules:	The value of this property can be either `none` or a space separated list that consists of any of the values `content`, `position`, or `size`.
DTD:	`<!ATTLIST style:properties fo:protect CDATA #IMPLIED>`
Note:	XSL does not support this property.

Horizontal Position

The `style:horizontal-pos` property specifies the horizontal alignment of the frame in relation to the specific area.

XML Code:	`style:horizontal-pos`
Rules:	The value of this property can be one of the following: `from-left`, `left`, `center`, `right`, `from-inside`, `inside`, or `outside`. The area that the position relates to is specified by the `style:horizontal-rel` property. The values `from-inside`, `inside` and `outside` correspond to the values `from-left`, `left`, and `right` on pages that have an odd page number and to the opposite values on pages that have an even page number. If the property value is `from-left` or `from-inside`, the `svg:x` attribute associated with the frame element specifies the horizontal position of the frame. Otherwise the `svg:x` attribute is ignored for text documents. It is also possible to use an `svg:x` attribute within a graphic style. If this is the case, then the attribute specifies a default position for new frames that are created using this style. Some values may be used in connection with certain frame anchor and relation types only.
DTD:	`<!ATTLIST style:properties style:horizontal-pos (from-left|left|center|right|from-inside|inside|outside)#IMPLIED>`

Horizontal Relation

The `style:horizontal-rel` property specifies the area to which the horizontal position of a frame relates. See the previous section for information on the `style:horizontal-pos` property.

XML Code:	`style:horizontal-rel`
Rules:	The value of this property can be one of the following: `page`, `page-content`, `page-start-margin`, `page-end-margin`, `frame`, `frame-content`, `frame-start-margin`, `frame-end-margin`, `paragraph`, `paragraph-content`, `paragraph-start-margin`, `paragraph-end-margin`, or `char`. You can use some values with certain frame anchor types only. The value `start-margin` determines the left margin, except when the horizontal position is `from-inside`, `inside` or `outside` and the anchor for the frame is on a page with an even page number, in which case it determines the right margin. The value `end-margin` determines the opposite margin to the `start-margin` values.
DTD:	`<!ATTLIST style:properties style:horizontal-rel (page|page-content|page-start-margin| page-end-margin|frame|frame-content| frame-start-margin|frame-end-margin| paragraph|paragraph-content|paragraph-start-margin| paragraph-end-margin|char) #IMPLIED>`

Vertical Position

The style:vertical-pos property specifies the vertical alignment of the frame in relation to a specific area.

XML Code:	style:vertical-pos
Rules:	The value of this property can be one of the following: from-top, top, middle, or bottom. The area that the position relates to is specified by the style:vertical-rel property. If the value of this property is from-top, the svg:y attribute associated with the frame element specifies the vertical position of the frame. Otherwise, the svg:y attribute is ignored for text documents. It is also possible to use an svg:y attribute within a graphic style. If this is the case, the attribute specifies a default position for new frames that are created using this style. Some values may be used in connection with certain frame anchor and relation types only.
DTD:	<!ATTLIST style:properties style:vertical-pos (from-top|top|middle|bottom) #IMPLIED>

Vertical Relation

The style:vertical-rel property specifies the area to which the vertical position of a frame relates. See the previous section for information on the style:vertical-pos property.

XML Code:	style:vertical-rel
Rules:	The value of this property can be one of the following: page, page-content, frame, frame-content, paragraph, paragraph-content, line, baseline, or char. You can use some values with certain frame anchor types only.
DTD:	<!ATTLIST style:properties style:vertical-rel (page|page-content|frame|frame-content| paragraph|paragraph-content|line|baseline|char| text) #IMPLIED>

Frame Anchor

In text documents, every frame must have an anchor. Frame anchors are described in detail in Chapter 3.

Frame Background

The background properties for a frame are specified in the same way as the background properties for a paragraph. See Chapter 3 for more information.

Border , Border Line Width, Padding, Shadow, and Columns

See Chapter 3 for information on these properties.

Editable

A text box can be editable even if the document in which it is contained is a read-only document. The style:

editable property specifies if a text box can be edited.

XML Code:	style:editable
Rules:	
DTD:	<!ATTLIST style:properties style:editable %boolean; #IMPLIED>
Note:	XSL does not support this property.
Implementation limitation:	This property is only supported by OpenOffice.org Writer.

Wrapping

The style:wrap property specifies how text around a frame is treated. For example, text can run around the left side of the frame, around the right side of the frame, or through the frame.

XML Code:	style:wrap
Rules:	
DTD:	<!ATTLIST style:properties style:wrap (none|left|right|parallel|dynamic|run-through)
Note:	XSL does not support the concept of wrapping. Instead, the wrap mode of a frame is determined by the anchor type. If wrapping is disabled or allowed for a single paragraph only, this can be simulated by an fo:clear property that is attached to an element following the frame.
Implementation limitation:	Currently, this property is only evaluated by OpenOffice.org Writer.

Paragraph-only Wrapping

If the anchor position of a frame is a paragraph or a character, and the wrap mode specified by the style:wrap property is left, right, parallel, or dynamic, you can specify the number of paragraphs that wrap around the frame.

XML Code:	style:number-wrapped-paragraphs
Rules:	This property is only recognized by frames or styles that have a style:wrap property attached with a value of left, right, parallel, or dynamic. If the value is no-limit, there is no limit on the number of paragraphs that are allowed to wrap around a frame.
DTD:	<!ATTLIST style:properties style:number-paragraphs-wrapped % number_or_no_limit; #IMPLIED>
Implementation limitation:	This property is only evaluated by OpenOffice.org Writer. Currently, if the value of this property is set to any number other than 1, the effect on the frame is the same as if the value was set to no-limit.

Contour Wrapping

For some frame types you can specify that the text should wrap around the shape of the object in the frame rather than around the frame itself. This is called contour wrapping.

XML Code:	`style:wrap-contour`
Rules:	This property is only recognized by frames or styles that have a `style:wrap` property attached.
DTD:	`<!ATTLIST style:properties style:wrap-contour %boolean; #IMPLIED>`
Implementation Limitation:	This property is only evaluated by OpenOffice.org Writer.

Contour Wrapping Mode

If the `style:wrap-contour` attribute is present, you can further specify how the text should wrap around the contour.

XML Code:	`style:wrap-contour-mode`
Rules:	This attribute is only recognized by frames or styles that already have the `style:wrap` and `style:wrap-contour` attributes attached. The value of the attribute can be `outside` or `full`. If the value of the attribute is `outside`, the text wraps around the general area to the left and right of the shape. If the value of the attribute is `full`, the text wraps around the shape and fills any possible spaces and indentations in the shape.
DTD:	`<!ATTLIST style:properties style:wrap-contour-mode (full|outside) #IMPLIED>`
Implementation Limitation:	This property is only evaluated by OpenOffice.org Writer.

Run Through

If the value of the `style:wrap` attribute is `run-through`, you can further specify whether the content of the frame should be displayed in the background or in the foreground. This attribute is usually used for transparent objects.

XML Code:	`style:run-through`
Rules:	The value of this attribute can be `foreground` or `background`. If the value is `foreground`, the frame content is displayed in front of the text. If the value is `background`, the frame content is displayed behind the text.
DTD:	`<!ATTLIST style:attributed style:run-through (foreground|background)>`
Note:	XSL does not support this property.

Mirroring

The `style:mirror` property specifies whether or not an image is mirrored before it is displayed. The mirroring can be vertical or horizontal. Horizontal mirroring can be restricted to images that are only located on either odd or even pages.

XML Code:	`style:mirror`
Rules:	The value of this attribute can be `none`, `vertical`, `horizontal`, `horizontal-on-odd`, or `horizontal-on-even`. You can specify the value `vertical` and the various horizontal values together, separating them by a white space.
DTD:	`<!ATTLIST style:properties style:mirror CDATA #IMPLIED>`

Clipping

The `fo:clip` property specifies whether to display:

- A rectangular section of an image

or

- The entire image

XML Code:	`fo:clip`
Rules:	
DTD:	`<!ATTLIST style:properties fo:clip CDATA #IMPLIED>`
Note:	This attribute is the same in XSL.

2.6.12 Floating Frame Formatting Properties

The attributes described in this section can be assigned to a graphic style that is assigned to floating frames.

Display Scrollbar

This attribute specifies whether or not vertical and horizontal scrollbars are displayed.

XML Code:	`draw:display-scrollbar`
Rules:	This attribute can be assigned to automatic styles only.
DTD:	`<!ATTLIST style:properties draw:display-scrollbar %boolean; #IMPLIED>`
Note:	If this attribute is not specified, scrollbars are displayed if they are required.

Display Border

This attribute specifies whether or not a border is displayed on the floating frame.

XML Code:	`draw:display-border`
Rules:	This attribut ecan be assigned to automatic styles only.
DTD:	`<!ATTLIST style:properties draw:display-border %boolean; #IMPLIED>`
Note:	This attribute is not the same as the `fo:border` property.

Margins

These attributes specify the horizontal and vertical margins between the border and the content of the floating frame. If these attributes are not specified, the default margins are used.

XML Code:	`draw:margin-horizontal` `draw:margin-vertical`
Rules:	These attributes can be assigned to automatic styles only. The value of these attributes must be a length in pixels.

DTD:	`<!ENTITY % nonNegativePixelLength "CDATA">`
	`<!ATTLIST style:properties draw:margin-horizontal %`
	`nonNegativePixelLength; #IMPLIED>`
	`<!ATTLIST style:properties draw:margin-vertical %`
	`nonNegativePixelLength; #IMPLIED>`

2.6.13 Object Formatting Properties

The attributes described in this section can be assigned to a graphic style that is assigned to objects.

Visible Area

The visible area of an object is the rectangular area of the object that is currently visible. When the entire object is visible, the values of the `draw:visible-area-left` and `draw:visible-area-top` attributes are `0` and the `draw:visible-area-width` and `draw:visible-area-height` attributes specify the size of the object.

XML Code:	`draw:visible-area-left`
	`draw:visible-area-top`
	`draw:visible-area-width`
	`draw:visible-area-height`

Rules:	These attributes can be assigned to automatic styles only.

DTD:	`<!ATTLIST style:properties draw:visible-area-left %`
	`nonNegativeLength; #IMPLIED>`
	`<!ATTLIST style:properties draw:visible-area-top %`
	`nonNegativeLength; #IMPLIED>`
	`<!ATTLIST style:properties draw:visible-area-width %`
	`positiveLength; #IMPLIED>`
	`<!ATTLIST style:properties draw:visible-Area-height %`
	`positiveLength; #IMPLIED>`

Implementation limitation:	Not all objects aupport these attributes. Some objects, such as all OpenOffice.org objects, store and load their own visible area. These attributes specify a default visible area that the object has the option to use.

2.6.14 Frame Events

You can assign events to a frame. The events that are attached to, for example, a text box or an image, are represented by an event element as described in Section 2.11. This element is contained within the frame type element, for example, the `<draw:text-box>` element or the `<draw:image>` element.

2.7 Forms and Controls

See the chapter entitled *Form Content* for information about forms and controls.

2.8 Hyperlinks

This section describes how the OpenOffice.org XML file format represents hyperlinks in documents.

2.8.1 Simple Hyperlinks

A simple hyperlink is a link that locates one resource only. Simple hyperlinks are represented by the `<office:a>` element.

XML Code:	`<draw:a>`
Rules:	This element is an XLink and has some attributes with fixed values and describe the semantics of the link. The element's content is the frame that should be the source of the link.

DTD:	`<!ELEMENT draw:a %frame;>`
	`<!ATTLIST draw:a xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST draw:a xlink:actuate (onRequest) "onRequest">`

The attributes that you can associate with the `<draw:a>` element are:

- Link location
- Link target frame
- Name
- Server side image map

Link Location

The `xlink:href` attribute specifies the target location of the link.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST draw:a xlink:href %url; #REQUIRED>`

Link Target Frame

The `office:target-frame` attribute specifies the target frame of the link.

XML Code:	`office:target-frame`
Rules:	This attribute can have one of the following values: • `_self` : The referenced document replaces the content of the current frame. • `_blank` : The referenced document is displayed in a new frame. • `_parent` : The referenced document is displayed in the parent frame of the current frame. • `_top` : The referenced document is displayed in the topmost frame, that is the frame that contains the current frame as a child or descendent but is not contained within another frame. • A frame name : The referenced document is displayed in the named frame. If the named frame does not exist, a new frame with that name is created. To conform with the XLink Specification, an additional `xlink:show` attribute is attached to the `<draw:a>` element. See page 24 for a pointer to the XLink Specification. If the value of the this attribute is `_blank`, the `xlink:show` attribute value is `new`. If the value of the this attribute is any of the other value options, the value of the `xlink:show` attribute is `replace`.
DTD:	`<!ATTLIST draw:a office:target-frame CDATA "_blank">` `<!ATTLIST draw:a xlink:show (new|replace) "replace">`

Name

A simple link can have a name, but it is not essential. The `office:name` attribute specifies the name of the link. The name can serve as a target for other hyperlinks.

XML Code:	`office:name`
Rules:	The name does not have to be unique.
DTD:	`<!ATTLIST draw:a office:name CDATA #IMPLIED>`
Notes:	This attribute is specified for compatibility with HTML only, where an `<a>` element may serve as a link source and target simultaneously. We strongly recommend that you do not use this attribute for any purpose other than to represent links that originally came from a HTML document.

Server Side Image Map

A link can be a server side image map. The `office:server-map` attribute is used by the server to determine which link to activate within the image map. If this attribute is present, the mouse coordinates of the click position of the frame are appended to the URL of the link.

XML Code:	`office:server-map`
Rules:	
DTD:	`<!ATTLIST draw:a office:server-map %boolean; "FALSE">`

2.8.2 Client Side Image Maps

An image map is a collection of hyperlinks that are associated with graphic elements. The image map is a sequence of image map elements. Each image map element associates a hyperlink with an area. The area can be one of the following shapes:

- Rectangular
- Circular
- Polygonal

The <draw:image-map> element represents an image map.

XML Code:	<draw:image-map>
Rules:	
DTD:	<!ELEMENT draw:image-map (draw:area-rectangle|draw:area-circle|draw:area-polygon)*>

The <draw:image-map> element can contain three types of image map elements, which represent the three types of image map areas as follows:

- Rectangular image map elements
- Circular image map elements
- Polygonal image map elements

Note: Image map elements are described in terms of absolute positions. When loading the XML file, OpenOffice.org maps the image map onto its associated graphical element, for example an image, in its original size. The application scales the image map to match the current size of the image, but in the file format the image is always saved in its unscaled version, matching the dimensions of the unscaled image.

Rectangular Image Map Areas

The <draw:area-rectangle> element describes a rectangular image map area.

XML Code:	<draw:area-rectangle>
Rules:	The attributes displayed below are required. The attributes described in the *Common Image Map Attributes* section are optional.
DTD:	<!ELEMENT draw:area-rectangle (svg:desc?,office:events?)> <!ATTLIST draw:area-rectangle svg:x %coordinate; #REQUIRED> <!ATTLIST draw:area-rectangle svg:y %coordinate; #REQUIRED> <!ATTLIST draw:area-rectangle svg:width %coordinate; #REQUIRED> <!ATTLIST draw:area-rectangle svg:height %coordinate; #REQUIRED>

Circular Image Map Areas

The <draw:area-circle> element describes a circular image map area. The additional attributes for circular image maps are described below in the common attributes section.

XML Code:	`<draw:area-circle>`
Rules:	The attributes displayed below are required. The `svg:cx` and `svg:cy` attibutes specify the center point of the circle. The `svg:r` attribute specifies the radius of the circle. The attributes described in the *Common Image Map Attributes* section are optional.

DTD:	`<!ELEMENT draw:area-circle (svg:desc?,office:events?)>`
	`<!ATTLIST draw:area-circle svg:cx %coordinate; #REQUIRED>`
	`<!ATTLIST draw:area-circle svg:cy %coordinate; #REQUIRED>`
	`<!ATTLIST draw:area-circle svg:r %coordinate; #REQUIRED>`

Polygonal Image Map Areas

The `<draw:area-polygon>` element describes a polygonal image map area. A polygonal image map area is comprised of the following components:

- A bounding box.
  The bounding box, which is represented in the same way as a rectangular image map area using the `svg:x`, `svg:y`, `svg:width`, and `svg:height` attributes, establishes the reference frame for the view box and the polygon point sequence. The reference frame enables the coordinates to be translated into absolute coordinates.
- A view box.
  The view box establishes a coordinate system for the point sequence. The view box obviates the need to record every point of the point sequence as absolute coordinates with length and unit of measurement.
- A sequence of points in view box coordinates.

For more information about how to represent polygons, see Chapter 55.3.4.

XML Code:	`<draw:area-polygon>`
Rules:	The attributes displayed below are required. The attributes described in the *Common Image Map Attributes* section are optional.

DTD:	`<!ELEMENT draw:area-polygon (svg:desc?,office:events?)>`
	`<!ATTLIST draw:area-polygon svg:x %coordinate; #REQUIRED>`
	`<!ATTLIST draw:area-polygon svg:y %coordinate; #REQUIRED>`
	`<!ATTLIST draw:area-polygon svg:width %coordinate; #REQUIRED>`
	`<!ATTLIST draw:area-polygon svg:height %coordinate; #REQUIRED>`
	`<!ATTLIST draw:area-polygon svg:points %points; #REQUIRED>`
	`<!ATTLIST draw:area-polygon svg:viewBox CDATA #REQUIRED>`

Example: Polygonal image map area

The element shown in the following example defines a triangle that is located in the middle of a 2cm by 2cm image. The bounding box covers an area of 2cm by 1.5cm. One view box unit corresponds to 0.01mm.

```
<draw:area-polygon …
   svg:x="0" svg:y="0" svg:width="2.0cm" svg:height="2.0cm"
   svg:viewBox="0 0 2000 2000"
   svg:points="400,1500 1600,1500 1000,400"/>
```

Common Image Map Attributes

In addition to the shape attributes, each image map element can contain the following information:

- Link, including a URL and link target frame.

- Name.
- Inactive flag.
- Description. Use the <svg:description> child element.
- Events associated with the area. Use the <office:events> child element.

Other attributes of the image maps are taken from the HTML image map representation.

Each image map element identifies a hyperlink and uses the Xlinkhref, type, and show attributes, and the office:target-frame-name attribute to describe the link.

XML Code:	xlink:href
Rules:	
DTD example:	<!ATTLIST draw:area-polygon xlink:href %url; #IMPLIED> <!ATTLIST draw:area-polygon xlink:type (simple) #IMPLIED> <!ATTLIST draw:area-polygon xlink:show (new|replace) #IMPLIED> <!ATTLIST draw:area-polygon office:target-frame-name CDATA #IMPLIED>

The office:name attribute assigns a name to each image map element.

XML Code:	office:name
Rules:	
DTD example:	<!ATTLIST draw:area-polygon office:name CDATA #IMPLIED>

The draw:nohref attribute declares that the image map element and the associated area is inactive. The URL that is contained in the image map element is not used.

XML Code:	draw:nohref
Rules:	
DTD:	<!ATTLIST draw:area-polygon draw:nohref (nohref) #IMPLIED>
Note:	In HTML, this is achieved by using the nohref attribute.

2.9 Number Format

The OpenOffice.org XML number format consists of three parts:

- Prefix – the text that is displayed before the number
- Display format specification, for example, A, B, C, or 1, 2, 3
- Suffix – the text that is displayed after the number

2.9.1 Prefix and Suffix

The style:num-prefix and style:num-suffix attributes specify what to display before and after the number.

XML Code:	`style:num-prefix` and `style:num-suffix`
Rules:	If the prefix and suffix do not contain alphanumeric characters, an XSLT `format` attribute can be created from the OpenOffice.org attributes by concatenating the values of the `style:num-prefix`, `style:num-format,` and `style:num-suffix` attributes.
DTD:	`<!ATTIST style:properties style:num-prefix CDATA #IMPLIED>` `<!ATTIST style:properties style:num-prefix CDATA #IMPLIED>`
Note:	Within the XSLT `format` attribute, the prefix and suffix can only be non-alphanumeric characters. This restriction does not apply to OpenOffice.org software.

2.9.2 Format Specification

The `style:num-format` attribute specifies the format of the number.

XML Code:	`style:num-format`
Rules:	The value of this attribute can be "1", "a", "A", "i", or "I".
DTD:	`<!ATTLIST style:properties style:num-format CDATA #IMPLIED>`
Note:	The valid values for this attribute have the same meanings as a single alphanumeric token that appears within an XSLT `format` attribute.

2.9.3 Letter Synchronization in Number Formats

If letters are used in alphabetical order for numbering, there are two ways to process overflows within a digit, as follows:

- You can insert a new digit starting with a value of a or A, that is incremented every time an overflow occurs in the following digit. The numbering sequence in this case is something like a,b,c, ..., z, aa,ab,ac, ...,az, ba, ..., and so on.
- You can insert a new digit that always has the same value as the following digit. The numbering sequence in this case is something like a, b, c, ..., z, aa, bb, cc, ..., zz, aaa, ..., and so on. This is called **letter synchronization**.

XML Code:	`style:num-letter-sync`
Rules:	
DTD:	`<!ATTLIST style:properties style:num-letter-sync %boolean; #IMPLIED>`
Note:	XSLT does not support letter synchronization.

2.10 Scripts

Scripts do not imply a scripting language or an object model. For this reason, a script can operate on the Document Object Model (DOM) of an OpenOffice.org XML document or on the OpenOffice.org API.

Scripts cannot modify a document while the document is loading. However, some events are called immediately after the document is loaded.

Each script is represented by a `<script:script>` element.

XML Code:	`<script:script>`
Rules:	This element contains the source code of the script. It can also optionally contain a compiled version of the script. The compiled version of the script is contained within a `<script:byte-code>` block, which can contain any content. All scripts are contained within the scripting section of a document.
Note:	JavaScript scripts that operate on the HTML DOM are special scripts that have their own representation and they can only be used in text documents.

2.11 Event Tables

Many objects such as controls, images, text boxes, or an entire document support events. An event binds the occurrence of a particular condition to an action that is executed if the condition arises. For example, if a user places the cursor over a graphic, this condition triggers an action that is supported by OpenOffice.org. This event, called "on-mouse-over", can be associated with an OpenOffice.org macro that is executed whenever the condition occurs, that is, whenever a user places the cursor over a graphic.

The XML representation of events and event tables is structured as follows:

- All of the event elements that are associated with an object are located in a container element called `<office:events>`.
- Each event-to-action association is recorded in one `<script:event>` element.
- Depending on the type of action that the event triggers, the following elements are used:
  - The `<script:event>` element represents events that are bound to a macro or script.
  - The `<presentation:event>` element represents events that are bound to an action that is specific to a presentation, for example, go to the next page.

The `<office:events>` element specifies the table of events that are associated with an object.

XML Code:	`<office:events>`
Rules:	
DTD:	`<!ELEMENT office:events (script:event,presentation:event)+>`

2.11.1 Event

The `<script:event>` element binds an event to an OpenOffice.org macro.

XML Code:	`<script:event>`
Rules:	
DTD:	`<!ELEMENT script:event (#PCDATA)>`
Implementation limitation:	The only script language that is currently supported is StarBasic. Other script languages may be supported in the future.

The attributes that you can associate with the `<script:event>` element are:

- Event name
- Script language
- Library name

- Macro name

Event Name

The `script:event-name` attribute specifies the name of the event.

XML Code:	`script:event-name`
Rules:	
DTD:	`<!ATTLIST script:event script:event-name CDATA #REQUIRED>`

Script Language

The `script:language` attribute specifies the scripting language in which the macro or script which is associated with the event is written.

XML Code:	`script:language`
Rules:	
DTD:	`<!ATTLIST script:event script:language CDATA #REQUIRED>`

Macro Name

The `script:macro-name` attribute specifies the name of the OpenOffice.org macro associated with the event.

XML Code:	`script:macro-name`
Rules:	
DTD:	`<!ATTLIST script:event script:macro-name #IMPLIED>`

Library Name

The `script:library` attribute specifies the library in which the OpenOffice.org macro associated with the event is contained.

XML Code:	`script:library`
Rules:	
DTD:	`<!ATTLIST script:event script:library #IMPLIED>`

2.12 Change Tracking

Change tracking content and structure varies depending on the type of document you are tracking. For example, change tracking in text documents is very different from change tracking in spreadsheets. The same applies to the XML file formats of these document types. However, the integration of change tracking information follows the same design principles in both applications and one XML element is used by both document types. For more information on change tracking in a particular type of the document, see the appropriate chapter of this book.

In both text documents and spreadsheets, the default document flow of an XML document reflects the current state of the document. An XSLT stylesheet or any other application that does not acknowledge change tracking

information, does for example, process all insertions into the document but does not process any deletions. Insertions are part of the default document flow, while deletions appear outside the default document flow. When an application processes a document in its current state, it does not interpret change tracking information. When an application processes the content of insertions or deletions to the document, it must interpret the change tracking information.

XSLT stylesheets are not intended to process change tracking information and the representation of change tracking information is not optimized to be processed by such scripts. Therefore, it is almost impossible for an XSLT spreadsheet to make deletions visible or insertions invisible. To do this, you need a binary application or an application that processes the DOM of a document.

2.12.1 Change Information

There is some information that all tracked changes have in common. The change information is represented by a `<office:change-info>` element and the common pieces of information are represented by attributes associated with the element.

XML Code:	`<office:change-info>`
Rules:	
DTD:	`<!ELEMENT office:change-info (text:p)*>`

The attributes that you can associate with the `<office:change-info>` element are:

- Author
- Date and time
- Comment (optional)

Author

The `office:chg-author` attribute specifies the name of the author who changed the document.

XML Code:	`office:chg-author`
Rules:	
DTD:	`<!ATTLIST office:change-info office:chg-author CDATA #REQUIRED>`

Date and Time

The `office:chg-date-time` attribute specifies the date and time when the change took place.

XML Code:	`office:chg-date-time`
Rules:	
DTD:	`<!ATTLIST office:change-info office:chg-date-time %date; #REQUIRED>`

Comment

The `<office:change-info>` element contains a comment from the author about a change.

Example: Sample change information

```
<office:change-info office:chg-author="Michael Brauer"
                    office:chg-date-time="1999-06-18T17:05:28">
  <text:p>Section about sections reworked to meed requirements of page
styles</text:p>
</office:changed>
```

2.13 OpenOffice.org Application Settings

The OpenOffice.org application settings are contained in a <office:settings> element.

XML Code:	<office:settings>
Rules:	
DTD:	<!ELEMENT office:settings (config:config-item-set+)>

The settings for OpenOffice.org applications are divided into the following two categories:

- Document settings, for example default printer.
- View settings, for example zoom level.

Settings that are shared by different applications have the same name and are of the same type.

2.13.1 Base Settings

The <config:config-item> element contains all base settings.

XML Code:	<config:config-item>
Rules:	The config:name attribute identifies the name of the setting. The config:type attribute identifies the type of setting. The value of the setting is stored in the element.
DTD:	<!ELEMENT config:config-item (#PCDATA)> <!ATTLIST config:config-item config:name CDATA #REQUIRED config:type (boolean | short | int | long | double | string | datetime | base64Binary) #REQUIRED>

2.13.2 Sequence of Settings

The <config:config-item-set> element is a container element for all types of setting elements. The settings can be contained in the element is any order.

XML Code:	`<config:config-item-set>`
Rules:	
DTD:	`<!ELEMENT config:config-item-set (config:config-item | config:config-item-set | config:config-item-map-named | config:config-item-map-indexed)+>` `<!ATTLIST config:config-item config:name CDATA #REQUIRED>`

2.13.3 Index Access of Sequences

The `<config:config-item-map-indexed>` element is a container element for sequences. The order specifies the index of the elements.

XML Code:	`<config:config-item-map-indexed>`
Rules:	The `config:name` attribute is required with this element.
DTD:	`<!ELEMENT config:config-item-map-indexed (config:config-item-map-entry)+>` `<!ATTLIST config:config-item-map-indexed config:name CDATA #REQUIRED>`

Map Entry

The `<config:config-item-map-entry>` element is a container element for all types of setting elements.

XML Code:	`<config:config-item-map-entry>`
Rules:	The `config:name` attribute is optional with this element.
DTD:	`<!ELEMENT config:config-item-map-entry (config:config-item | config:config-item-set | config:config-item-map-named | config:config-item-map-indexed)+>` `<!ATTLIST config:config-item-map-entry config:name CDATA #IMPLIED>`

2.13.4 Name Access of Sequences

The `<config:config-item-map-named>` element is a container element for sequences.

XML Code:	`<config:config-item-map-named>`
Rules:	The `config:name` attribute is required with this element.
DTD:	`<!ELEMENT config:config-item-map-named (config:config-item-map-entry)+>` `<!ATTLIST config:config-item-map-named config:name CDATA #REQUIRED>`

CHAPTER 3

Text Content

This chapter describes the OpenOffice.org XML representation of text content. It contains the following sections:

- Headings and Paragraphs
- Sections
- Bulleted and Numbered Lists
- Outline Numbering
- Line Numbering
- Footnotes and Endnotes
- Fields
- Variable Fields
- Frames in Text Documents
- Ruby
- Text Formatting Properties
- Paragraph Formatting Properties
- Section Formatting Properties
- Change Tracking in Text Documents
- Optional Information

3.1 Headings and Paragraphs

This section describes the XML elements and attributes that you use to represent heading and paragraph components in a text document.

3.1.1 Primary Heading and Paragraph Components

The XML elements that represent both headings and paragraphs are called **paragraph element**.

XML Code:	For headings:
	`<text:h>`
	For paragraphs:
	`<text:p>`

Rules:	The text of the paragraph is contained in the paragraph element. It can be mixed with many other elements.

DTD:	`<!ENTITY % inline-text "text:s|text:tab-stop|text:line-break| text:span|text:a| %frames;|%fields;|text:footnote| text:ref-point|text:bookmark| text:bookmark-start|text:bookmark-end| text:change|text:change-start| text:change-end">`
	`<!ELEMENT text:p (#PCDATA|%inline-text;)*>`
	`<!ELEMENT text:h (#PCDATA|%inline-text;)*>`

If the paragraph element or any of its child elements contains white-space characters, they are collapsed, in other words they are processed in the same way that HTML processes them. The following Unicode characters are normalized to a SPACE character:

- HORIZONTAL TABULATION (0x0009)
- CARRIAGE RETURN (0x000D)
- LINE FEED (0x000A)
- SPACE (0x0020)

In addition, these characters are ignored if the preceding character is a white-space character. The preceding character can be contained in the same element, in the parent element, or in the preceding sibling element, as long as it is contained within the same paragraph element and the element in which it is contained processes white-space characters as described above.

White-space processing takes place within the following elements:

- `<text:p>`
- `<text:h>`
- `<text:span>`
- `<text:a>`
- `<text:ref-point>`
- `<text:ref-point-start>`
- `<text:ref-point-end>`
- `<text:bookmark>`
- `<text:bookmark-start>`
- `<text:bookmark-end>`

Note: In XSL, you can enable white-space processing of a paragraph of text by attaching an `fo:white-space="collapse"` attribute to the `<fo:block>` element that corresponds to the paragraph element.

The attributes that you can associate with heading and paragraph elements are:

- Heading level
- Style and conditional style
- Paragraph formatting properties

Heading Level

The `text:level` attribute associated with the heading element determines the level of the heading, for example, Heading 1, Heading 2, and so on.

XML Code:	`text:level`
Rules:	This attribute is associated with a `<text:h>` element.
DTD:	`<!ATTLIST text:h text:level %number; "1">`
Note:	You must apply the same style to all headings of the same level.

Style and Conditional Style

The style and conditional style attributes are optional.

XML Code:	`text:style-name` and `text:cond-style-name`
Rules:	The values of these attributes are style names, which is sufficient to identify a style because every style addressed must be a paragraph style. If a conditional style is applied to a paragraph, the `text:style-name` attribute contains the name of the style that is applied under that condition. The `text:style-name` attribute does not contain the name of the conditional style that contains the conditions and maps to other styles. The conditional style name is the value of the `text:cond-style-name` attribute.

DTD:	`<!ATTLIST text:p text:style-name %style-name; #REQUIRED>`
	`<!ATTLIST text:p text:cond-style-name %style-name; #IMPLIED>`
	`<!ATTLIST text:h text:style-name %style-name; #REQUIRED>`
	`<!ATTLIST text:h text:cond-style-name %style-name; #IMPLIED>`

Note:	This XML structure simplifies XSLT transformations because XSLT only has to acknowledge the conditional style if the formatting attributes are relevant. The referenced style can be a common style or an automatic style.

Since most documents use one paragraph style for the majority of paragraphs in the document, there are plans to develop a default style name for paragraphs.

Example: Styles and conditional styles in OpenOffice.org XML

```
<text:p text:style-name="Heading 1">
  "Heading 1" is not a conditional style.
</text:p>
<text:p text:style-name="Numbering 1" text:cond-style-name="Text body">
  "Text body" is a conditional style. If it is contained in a numbered
  paragraph, it maps to "Numbering 1". This is assumed in this example.
</text:p>
```

Paragraph Formatting Properties

The default formatting properties that are assigned to a paragraph are represented by an automatic style. When the document is exported, an automatic style is generated with the formatting properties of the paragraph. The parent style of the generated style is the common style that is assigned to the paragraph from the viewpoint of the OpenOffice.org user interface.

If a paragraph has a conditional style assigned and this style is mapped to another style because of a condition, two automatic styles are generated when the document is exported. Both styles have the same formatting properties assigned, but the parent of one style is the conditional style while the parent of the other style is the style that is applied because of the condition.

Example: Paragraph formatting properties in OpenOffice.org XML

```
<office:styles>
  <style:style name="Text Body" ...>
</office:styles>
...
<office:automatic-styles>
  <style:style name="P001" family="paragraph"
                           style:parent-style-name="Text Body">
    <style:properties fo:font-weight="bold"/>
  </style:style>
</office:automatic-styles>
...
<office:body>
  <text:p style:style-name="P001">
    This is a bold paragraph in "Text Body" style.
  </text:p>
</office:body>
```

3.1.2 White-Space Characters

In general, consecutive white-space characters in a paragraph are collapsed. For this reason, there is a special XML element used to represent the Unicode character SPACE (0x0020).

XML Code:	`<text:s>`
Rules:	This element uses an attribute called `text:c` to specify the number of SPACE characters that the element represents. This element is required to represent the second and all following SPACE characters in a sequence of SPACE characters. You do not get an error if the character preceding the element is not a white-space character, but it is good practice to use this element for the second and all following SPACE characters in a sequence. This way, an application recognizes a single space character without recognizing this element.
DTD:	`<!ELEMENT text:s EMPTY>` `<!ELEMENT text:s text:c %number “1”>`

3.1.3 Tab Stops

The `<text:tab-stop>` element represents tab stops in a heading or paragraph.

XML Code:	`<text:tab-stop>`
Rules:	
DTD:	`<!ELEMENT text:tab-stop EMPTY>`

3.1.4 Line Breaks

The `<text:line-break>` element represents a line break in a heading or paragraph.

XML Code:	`<text:line-break>`
Rules:	
DTD:	`<!ELEMENT text:line-break EMPTY>`
Note:	XSL does not support line breaks.

3.1.5 Text Styles

The `<text:span>` element represents portions of text that are formatted using a certain text style.

XML Code:	`<text:span>`
Rules:	The content of this element is the text that uses the text style. The name of the text style is the value of a `text:style-name` attribute attached to the `<text:span>` element. The `text:style-name` can be an automatic style. Since a `<text:span>` element never addresses any other type of style other than text styles, the style name is sufficient to identify the style. You can nest `<text:span>` elements. White-space characters contained in this element are collapsed.
DTD:	`<!ELEMENT text:span (#PCDATA|%inline-text;)*>` `<!ATTLIST text:span text:style-name %style-name; #IMPLIED>`
Notes:	The `<text:span>` attribute is similar to the HTML `<span>` attribute.

Example: Text style in OpenOffice.org XML

```
<text:p>
  The last word of this sentence is
  <text:span text:style-name="emphasize">emphasized</text:span>.
</text:p>
```

3.1.6 Text Formatting Properties

Formatting properties that are applied to a portion of text inside a paragraph are represented by an automatic text style, which is attached to the text portion in the same way as common text styles. See Section3.1.5 for more information. When the document is exported, an automatic text style is generated for all formatting properties that are attached to a text portion. You can assign two formatting properties to the same text portion using nested `<text:span>` elements, and the formatting properties can be represented by one or by two automatic text styles.

In most cases, automatic text styles do not have a parent style. The only situation where an automatic text style

might have a parent style is when a text portion has formatting properties and a common text style assigned. The text style can be the parent style of the automatic style, but it is not essential.

Note: In OpenOffice.org applications, the text portions that have a certain formatting property applied may overlap but the <text:span> elements cannot overlap.

Example: Text formatting properties in OpenOffice.org XML

This example shows the OpenOffice.org **XML code required to display the following sentence:**

The rain in *Spain* *stays* mainly in the plain.

```
<office:automatic-styles>
  <style:style name="T001" family="text">
    <style:properties fo:font-style="italic"/>
  </style:style>
  <style:style name="T002" family="text">
    <style:properties style:text-underline="single"/>
  </style:style>
</office:automatic-styles>
...
<office:body>
  <text:p>
    The rain in
    <text:span text:style-name="T001">
      Spain
      <text:span text:style-name="T002">
        stays
      </text:span>
    </text:span>
    <text:span text:style-name="T002">
      mainly in
    </text:span>
    the plain.
  </text:p>
  ...
</office:body>
```

3.1.7 Hyperlinks

Hyperlinks in text documents are represented by a <text:a> element.

XML Code:	<text:a>
Rules:	If this element contains white-space characters, the characters are collapsed.
DTD:	<!ELEMENT text:a (script:events?,(%inline-text;)*)>

This element also contains an event table element, <script:events>, which contains the events assigned to the hyperlink. See Chapter 2 for more information on the event table element.

The attributes that you can associate with the <text:a> element are:

- Name
- Link location
- Target frame
- Text styles

Name

A hyperlink can have a name, but it is not essential. The `text:name` attribute specifies the name of the hyperlink if one exists. This name can serve as a target for some other hyperlinks.

XML Code:	`text:name`
Rules:	This name does not have to be unique.
DTD:	`<!ATTLIST text:a text:name CDATA #IMPLIED>`
Notes:	This attribute is specified for compatibility with HTML only, where an `<a>` element may serve as a link source and target simultaneously. Do not use this attribute for any purpose other than to represent links that originally came from a HTML document.

Link Location

The `xlink:href` attribute specifies the URL for the target location of the link.

XML Code:	`xlink:href`
Rules:	

DTD:	`<!ATTLIST text:a xlink:href %url #REQUIRED>`
	`<!ATTLIST text:a xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST text:a xlink:actuate (onRequest) "onRequest">`

Target Frame

The `office:target-frame-name` attribute specifies the target frame of the link.

XML Code:	`office:target-frame-name`
Rules:	This attribute can have one of the following values: • `_self` – The referenced document replaces the content of the current frame. • `_blank` – The referenced document is displayed in a new frame. • `_parent` – The referenced document is displayed in the parent frame of the current frame. • `_top` – The referenced document is displayed in the uppermost frame, that is the frame that contains the current frame as a child or descendent but is not contained within another frame. • A frame name – The referenced document is displayed in the named frame. If the named frame does not exist, a new frame with that name is created. To conform with the XLink Specification, an additional `xlink:show` attribute is attached to the `<text:a>` element. See page 24 for a pointer to the XLink Specification. If the value of the attribute is `_blank`, the `xlink:show` attribute value is `new`. If the value of the attribute is any of the other value options, the value of the `xlink:show` attribute is `replace`.
DTD:	`<!ATTLIST text:a office:target-frame-name CDATA #REQUIRED>` `<!ATTLIST text:a xlink:show (new|replace) "replace">`

Text Styles

Every hyperlink has two text styles as follows:

- If the link location of the hyperlink was not visited, the text style specifies by the `text:style-name` attribute is applied to the text of the hyperlink.
- If the link location of the hyperlink was already visited, the text style specified by the `text:visited-style-name` attribute is applied to the text of the hyperlink.

XML Code:	`text:style-name` and `text:visited-style-name`
Rules:	
DTD:	`<!ATTLIST text:a text:style-name %style-name; #IMPLIED>` `<!ATTLIST text:a text:visited-style-name %style-name; #IMPLIED>`

3.1.8 Bookmarks

Bookmarks can either mark a text position or a text range. A text range can start at any text position and end at another text position. In particular, a bookmark can start in the middle of one paragraph and end in the middle of another paragraph. The XML element used to represent a bookmark varies depending on the type of bookmark, as follows:

- `<text:bookmark>` – to mark one text position
- `<text:bookmark-start>` – to mark the start position in a text range
- `<text:bookmark-end>` – to mark the end position in a text range

XML Code:	As above
Rules:	

DTD:	`<!ELEMENT text:bookmark EMPTY>`
	`<!ELEMENT text:bookmark-start EMPTY>`
	`<!ELEMENT text:bookmark-end EMPTY>`
	`<!ATTLIST text:bookmark text:name CDATA #REQUIRED>`
	`<!ATTLIST text:bookmark-start text:name CDATA #REQUIRED>`
	`<!ATTLIST text:bookmark-end text:name CDATA #REQUIRED>`

Example: Bookmarks in OpenOffice.org XML

```
<text:p>
  <text:bookmark text:name="Mark 1"/>There is a text mark in front of this
  paragraph.
  <text:bookmark-start text:name="Mark 2"/>In front of this paragraph
  there is
  the start of a bookmark.
</text:p>
<text:p>
  This bookmark ends
  <text:bookmark-end text:name="Mark 2"/>
  amid this sentence.
</text:p>
```

3.1.9 Index Entries

Information to be supplied.

3.1.10 References

The OpenOffice.org XML representation of references is modeled on the XML representation of bookmarks. There are two types of reference marks, as follows:

- A point reference
  A point reference marks a particular position in text and is represented by a single `<text:reference-mark>` element.
- A range reference
  A range reference marks a range of characters in text and is represented by two elements; `<text:reference-mark-start>` to mark the start of the range and `<text:reference-mark-end>` to mark the end of the range.

Every reference is identified by its name, which must be unique. In a range reference, the start and end elements must use the same reference name.

Note: The current version of the OpenOffice.org software does not support range references that span multiple paragraphs. If these types of range references exist, during import the OpenOffice.org software truncates the reference to the paragraph in which the `<text:reference-mark-start>` element appears.

Point References

The `<text:reference-mark>` element represents a point reference in XML.

XML Code:	`<text:reference-mark>`
Rules:	The name must not be reused for any other reference.
DTD:	`<!ELEMENT text:reference-mark EMPTY>` `<!ATTLIST text:reference-mark text:name %string; #REQUIRED>`

Range References

The `<text:reference-mark-start>` and `<text:reference-mark-end>` elements represent a range reference in XML.

XML Code:	`<text:reference-mark-start>` `<text:reference-mark-end>`
Rules:	The name must not be reused for any other reference.

DTD:	`<!ELEMENT text:reference-mark-start EMPTY>`
	`<!ELEMENT text:reference-mark-end EMPTY>`
	`<!ATTLIST text:reference-mark-start text:name % string; #REQUIRED>`
	`<!ATTLIST text:reference-mark-end text:name % string; #REQUIRED>`

In OpenOffice.org XML, three elements are used to represent references instead of one element because refer-

ences represented as a single XML element:

- Cannot support overlapping references
- Do not interact well with other elements

Take the following example:

Example: Overlapping range references

```
<text:p>
  <text:reference-mark-start text:name="first"/>This is an
  <text:reference-mark-start text:name="second"/>example of a sentence
  <text:reference-mark-end text:name="first"/>with overlapping references.
  <text:reference-mark-end text:name="second"/>
</text:p>
```

The example paragraph shows two references that cover the following text:

reference “first”	“This is an example of a sentence”
reference “second”	“example of a sentence with overlapping references.”

This overlapping structure cannot be represented using a single reference element to contain the referenced text. Similarly, a reference spanning multiple paragraphs creates the same situation as two overlapping XML elements, as does character formatting either starts or ends, but not both, within the referenced text.

3.1.11 Soft Hyphens, Hyphens, and Non-breaking Blanks

Soft hyphens, hyphens, and non-breaking blanks are represented by UNICODE characters.

The UNICODE character...	**Represents...**
SOFT HYPHEN (00AD)	soft hyphens
NON-BREAKING HYPHEN (2011)	non-breaking hyphens
NO-BREAK SPACE (00A0)	non-breaking blanks

3.2 Sections

A text section is a named region of text that can be associated with certain formatting properties. The section starts and ends on paragraph boundaries and can contain any number of paragraphs. Sections can contain regular text content or the text can be contained in another file and linked to the section. Sections can also be write-protected or hidden.

If a section is linked to another document, the link can be through one of the following:

- A resource identified by an Xlink, represented by a `text:section-source` element
- Dynamic Data Exchange (DDE,) represented by a `office:dde-source` element

If these elements are used, they must be the first element in a `<text:section>`.

Sections can have settings for text columns, background color or pattern, footnote and endnote configuration. These settings form the section style, which is represented in a `<style:styles>` element. The formatting properties for sections are explained in Section 3.12.

XML Code:	`<text:section>`
Rules:	
DTD:	`<!ELEMENT text:section ((text:section-source|office:dde-source)?, %text;)>`
Note:	In OpenOffice.org Writer, lists and sections can overlap. Therefore, a text section can start on any paragraph boundary even if the paragraph is part of a numbered or bulleted list. This causes a problem in the XML representation of lists because lists are represented using their own container elements and XML cannot represent overlapping elements. To solve the problem, a section boundary within a list causes the list to split.

Section Style

The `text:style-name` attribute specifies the section style.

XML Code:	`text:style-name`
Rules:	This attribute must refer to a section style.
DTD:	`<!ATTLIST text:section text:style-name %styleName; #IMPLIED>`

Section Name

Every section must have a name that uniquely identifies the section. The `text:name` attribute contains the name of the section.

XML Code:	`text:name`
Rules:	You should not change a section that is marked as protected.
DTD:	`<!ATTLIST text:section text:name %string; #REQUIRED>`

Hidden Sections and Conditional Sections

Sections can be hidden based on a condition or they can be hidden unconditionally.

The `text:display` attribute specifies whether or not the section is hidden.

XML Code:	`text:display`
Rules:	The value of this attribute can be: • `true`, the section is displayed. This is the default setting. • `none`, the section is hidden unconditionally. • `condition`, the section is hidden under the condition specified in the `text:condition` attribute.
DTD:	`<!ATTLIST text:section text:display (true|none|condition) "true">`

The `text:condition` attribute specifies the condition under which the section is hidden. The condition is encoded as a string.

XML Code:	`text:condition`
Rules:	If the value of `text:display` is `condition`, the `text:condition` attribute must be present.
DTD:	`<!ATTLIST text:section text:condition %formula; #IMPLIED>`

Protected Sections

You can protect a section, which means that a user can not edit the section. The `text:protected` attribute indicates whether or not a section is protected. The user interface must enforce the protection attribute if it is enabled.

XML Code:	`text:protected`
Rules:	
DTD:	`<!ATTLIST text:section text:protected %boolean; "false">`

A user can use the user interface to reset the protection flag, unless the section is further protected by a password. In this case, the user must know the password in order to reset the protection flag. The `text:protection-key` attribute specifies the password that protects the section. To avoid saving the password directly into the XML file, only a hash value of the password is stored.

XML Code:	`text:protection-key`
Rules:	The value of the this attribute is the MIME64 representation of password hash value.
DTD:	`<!ATTLIST text:section text:protection-key CDATA #IMPLIED>`
Note:	Currently, the OpenOffice.org XML file format uses the MD5 hash function to compute the protection key value.

3.2.1 Section Source

The `<text:section-source>` element indicates that the enclosed section is a linked section. If this element is used, it must be the first element in the `<text:section>` element.

XML Code:	`<text:section-source>`
Rules:	This element does not have any contents.
DTD:	`<!ELEMENT text:section-source EMPTY>`

The attributes that you can associate with the `<text:section-source>` attribute are:

- Section source URL
- Name of linked section
- Filter name

Section Source URL

These attributes identify the document or section to which the section is linked.

XML Code:	`xlink:href, xlink:type, xlink:show`
Rules:	The name of the target section is identified by the local part of the URL, following the hash mark.
DTD:	`<!ATTLIST text:section-source xlink:href %string; #IMPLIED xlink:type (simple) #FIXED "simple" xlink:show (embed) #FIXED "embed">`
Note:	The `xlink:href` attribute is implied because `<text:section-source>` elements may also link to internal sections.

Name of Linked Section

If the link targets a section of a document, the attribute `text:section` name contains the name of the target section.

XML Code:	`text:section-name`
Rules:	If the attribute is not present, the link targets the entire document.
DTD:	`<!ATTLIST text:section-source text:section-name %string; #IMPLIED>`

Filter Name

The `text:filter-name` attribute specifies the file type of the link target.

XML Code:	`text:filter-name`
Rules:	The value of this attribute is implementation-dependent.
DTD:	`<!ATTLIST text:section-source text:filter-name %string; #IMPLIED>`

3.2.2 DDE Source

If sections are linked via DDE, they are represented by a `<office:dde-source>` element. It contains attributes that specify the application, topic and item of the DDE connection.

XML Code:	`<office:dde-source>`
Rules:	
DTD:	`<!ELEMENT office:dde-source EMPTY>`

The attributes that you can associate with the `<office:dde-source>` element are:

- Target application
- Target topic
- Target item
- Automatic update

Target Application

The `office:dde-application` attribute specifies the name of the target application to use for the DDE connection.

XML Code:	`office:dde-application`
Rules:	
DTD:	`<!ATTLIST office:dde-source office:dde-application %string; #REQUIRED>`

Target Topic

The `office:dde-topic` attribute specifies the topic to use for the DDE connection.

XML Code:	`office:dde-topic`
Rules:	
DTD:	`<!ATTLIST office:dde-source office:dde-topic % string; #REQUIRED>`

Target Item

The `office:dde-item` attribute specifies the information that the target application will deliver.

XML Code:	`office:dde-item`
Rules:	
DTD:	`<!ATTLIST office:dde-source office:dde-item % string; #REQUIRED>`

Automatic Update

The `office:automatic-update` attribute indicates whether or not the linked section should be automatically updated.

XML Code:	`office:automatic-update`
Rules:	If this attribute is set `true`, the linked sections should be updated automatically.
DTD:	`<!ATTLIST office:dde-source office:automatic-update %boolean; "false">`

3.3 Bulleted and Numbered Lists

Bulleted and numbered lists consist of structural and layout information.

Structural information includes the following:

- List type – bulleted or numbered.
- List level – for example, main or sublist.

- Information about whether or not a certain paragraph contained in a list has a label, for example number or bullet.
- The number of a paragraph within a numbered list. This information is optional because it can be recalculated.

Layout information includes the following:

- The indentation of paragraphs in a list.
- The label width and the distance between it and the text.
- The bullet character or image for bulleted lists.
- The number format for numbered lists.

The structural information is contained in the document body, with the content. The OpenOffice.org XML representation of structural information is very similar to HTML. The layout information is contained within **list styles**. There are common list styles and automatic list styles.

3.3.1 List Blocks

A list is represented by the one of the following elements:

- `<text:ordered-list>`
  This element specifies an ordered list, that is a list where every list item is preceded by a number that is incremented for each list item.
- `<text:unordered-list>`
  This element specifies an unordered list, that is a list where every list item is preceded by the same bullet character or image.

XML Code:	As above
Rules:	Both elements have the same content; an optional list header followed by any number of list items. Every list has a **list level.** If a list is not contained within another list, the list level is 1. If the list in contained within another list, the list level is the list level of the list in which is it contained incremented by one. If a sublist is contained in a table cell or text box, the list level returns to 1, even though the list elements are nested.

DTD:	`<!ENTITY % list-items "((text:list-header|text:list-item), text:list-item*)">`
	`<!ELEMENT text:ordered-list %list-items;>`
	`<!ELEMENT text:unordered-list %list-items;>`

The attributes that you can associate with the list block elements are:

- Style name
- Continue numbering

Style Name

The `text:style-name` attribute specifies the name of the list style that is applied to the list.

XML Code:	`text:style-name`
Rules:	This attribute is optional and can be used with the `<text:ordered-list>` or `<text:unordered-list>` element. If this attribute is not included and therefore a list style is not specified, one of the following actions is taken: • If the list is contained within another list, the list style defaults to the style of the surrounding list. • If there is no list style specified for the surrounding list but the list contains paragraphs that have paragraph styles attached specifying a list style, this list style is used for any of these paragraphs. A default list style is applied to any other paragraphs.
DTD:	`<!ATTLIST text:ordered-list text:style-name %style-name; #IMPLIED>` `<!ATTLIST text:unordered-list text:style-name %style-name; #IMPLIED>`
Note:	To determine which formatting properties are applied to a list, the list level and list style name are taken into account. See Section 3.3.4 for more information on list formatting properties.

Continue Numbering

By default, the first list item in an ordered list starts with the number specified in the list style. If the list follows another ordered list and you want to continue the numbering from the preceding list, you can use thecontinue numbering attribute.

XML Code:	`text:continue-numbering`
Rules:	This attribute can be used with the `<text:ordered-list>` element and can have a value of `true` or `false`. If the value of the attribute is `true` and the numbering style of the preceding list is the same as the current list, the number of the first list item in the current list is the number of the last item in the preceding list incremented by one.
DTD:	`<!ATTLIST text:ordered-list text:continue-numbering %boolean; "false">`

3.3.2 List Header

A list header contains one or more paragraphs that are displayed before a list. The paragraphs are formatted like list items but they do not have a preceding number or bullet. The list header is represented by the list header element.

XML Code:	`<text:list-header>`
Rules:	This element contains paragraphs or sections. The element cannot contain headings, tables, or lists.
DTD:	`<!ELEMENT text:list-header (text:p|text:section)+>`

3.3.3 List Item

A `<text:list-item>` element can contain paragraphs, sections, or lists.

XML Code:	`<text:list-item>`
Rules:	The first line in a list item is preceded by a bullet or number, depending on the list style assigned to the list. If a list item starts another list immediately and does not contain any text, no bullet or number is displayed. A list item cannot contain headings or tables.
DTD:	`<!ENTITY % list-item-content "(text:p|text:section| text:ordered-list|text:unordered-list)+" <!ELEMENT text:list-item %list-item-content;>`

The attributes that you can associate with the `<text:list-item>` element are:

- Restart numbering
- Restart numbering value

Restart Numbering

You can restart the numbering of a list and the numbering of the surrounding lists by attaching the `text:restart-numbering` attribute to the `<text:list-item>` element.

XML Code:	`text:restart-numbering`
Rules:	This attribute can have a value of `true` or `false`. It can be used for list items in ordered or unordered lists. If the attribute is applied to a list item in an unordered list, it affects the numbering of all surrounding ordered lists and ordered lists that are contained within the item.
DTD:	`<!ATTLIST text:list-item text:restart-numbering %boolean; "false">`

Restart Numbering Value

You can restart the numbering of the current list at a certain number. Use the `text:start-value` attribute to specify the number with which to restart the list.

XML Code:	`text:start-value`
Rules:	This attribute can only be applied to paragraphs with a numbering list style. Unlike the `text:restart-numbering` attribute, it restarts the numbering of the current list only.
DTD:	`<!ATTLIST text:list-item text:start-value %number; #IMPLIED>`

Example: Ordered and unordered lists and sublists

```
<text:ordered-list text:style-name="List 1">
  <text:list-item>
    <text:p>This is the first list item</text:p>
    <text:p>This is a continuation of the first list item.</text:p>
  </text:list-item>
  <text:list-item>
    <text:p>This is the second list item.
            It contains an unordered sub list.</text:p>
    <text:unordered-list>
      <text:list-item><text:p>This is a sub list item.</text:p>
      <text:list-item><text:p>This is a sub list item.</text:p>
      <text:list-item><text:p>This is a sub list item.</text:p>
    </text:unordered-list>
  </text:list item>
  <text:list-item>
    <text:p>This is the third list item</text:p>
  </text:list-item>
</text:ordered-list>
```

3.3.4 List Styles

List styles specify the formatting properties for lists. A list style contains a set of specifications, each specification containing a set of properties to apply to a list of a certain list level. These specifications are called **list level styles**. If a list style is applied to a list but it does not contain a list level specification for the level of the list, the list level style of the nearest lower level is used. If a suitable list level style does not exist, a default style is used.

XML Code:	`<text:list-style>`
Rules:	This element contains a set of number or bullet list styles for different list levels. The list styles can be common or automatic styles.
DTD:	`<!ELEMENT text:list-style (text:list-level-style-number| text:list-level-style-bullet| text:list-level-style-image)+>`
Note:	List styles contain different properties than paragraph or text styles. This is why they are represented by a different element.

The attributes that you can associate with the `<text:list-style>` element are:

- Name
- Flag for recognition of current numbers
- Consecutive numbering

Name

The `style:name` attribute specifies the name of the list style.

XML Code:	`style:name`
Rules:	
DTD:	`<!ATTLIST text:list-style style:name %style-name; #REQUIRED>`

Consecutive Numbering

The `text:consecutive-numbering` attribute specifies whether or not the list style uses consecutive numbering for all list levels or whether each list level restarts the numbering.

XML Code:	`text:consecutive-numbering`
Rules:	This attribute can have a value of `true` or `false`.
DTD:	`<!ATTLIST text:list-style text:consecutive-numbering %boolean; "false">`

3.3.5 Number Level Style

A number level style specifies a list style where the list items are preceded by numbers.

XML Code:	`<text:list-level-style-number>`
Rules:	This element is contained in list style elements (`<text:list-style>`) only.
DTD:	`<!ELEMENT text:list-level-style-number (style:properties?)>`

The attributes that you can associate with the `<text:list-level-style-number>` element are:

- Level
- Start indent
- Minimum label width
- Minimum label distance
- Label alignment
- Text style
- Number format
- Display levels
- Start value

Level

The `text:level` attribute specifies the level of the number list style.

XML Code:	`text:level`
Rules:	The value of this attribute is a number. The number of the highest level is "1".
DTD:	`<!ATTLIST text:list-level-style-numbering text:level %number; #REQUIRED>`

Start Indent

The `text:space-before` attribute specifies the space to include before the number for all paragraphs at this level. If a paragraph has a left margin that is greater than 0, the actual position of the list label box is the left margin width plus the start indent value.

XML Code:	`text:space-before`
Rules:	This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element. The value of the attribute is an absolute value. This means that when the position of a label is calculated the start indent value of the current level is only considered. The start indent values for lower levels do not affect the label position.
DTD:	`<!ATTLIST style:properties text:space-before %length; #IMPLIED>`
Note:	This attribute does not conform with XSL or CSS specifications.

Minimum Label Width

The `text:min-label-width` attribute specifies the minimum width of a number.

XML Code:	`text:min-label-width`
Rules:	This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element. You can align the label horizontally with the width using an `fo:text-align` property. See the Label Alignment attribute below for more information.
DTD:	`<!ATTLIST style:properties text:min-label-width %length; #IMPLIED>`
Note:	This attribute does not conform with XSL or CSS specifications.

Minimum Label Distance

The `text:min-label-distance` attribute specifies the minimum distance between the number and the text of the list item.

XML Code:	`text:min-label-distance`
Rules:	This attribute can be associated with an item set element that is contained in a `<text:list-level-style-*>` element.
DTD:	`<!ATTLIST style:properties text:min-label-distance %length; #IMPLIED>`
Note:	This attribute does not conform with XSL or CSS specifications.

Label Alignment

The `fo:text-align` attribute specifies the horizontal alignment of a label (number) within the width specified by the `text:min-label-width` attribute.

XML Code:	`fo:text-align`
Rules:	This attribute is associated with the `text:min-label-width` attribute, which can be associated with an item set element that is contained in a `<text:list-level-style-*>` element.
DTD:	See Section 3.11.4 for a DTD and more information.
Note:	This attribute does not conform with XSL or CSS specifications.

Text Style

The `text:style-name` attribute specifies the name of the style to use to format the number of the list.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-numbering text:style-name % style-name #IMPLIED>`

Number Format

See Chapter 2 for detailed information on number format attributes. The attributes described in Chapter 2 can also be associated with the `<text:list-level-style-numbering>` element.

DTD:	`<!ATTLIST text:list-level-style-numbering style:num-format CDATA #REQUIRED>`
	`<!ATTLIST text:list-level-style-numbering style:num-prefix CDATA #IMPLIED>`
	`<!ATTLIST text:list-level-style-numbering style:num-suffix CDATA #IMPLIED>`
	`<!ATTLIST text:list-level-style-numbering style:num-letter-sync %boolean; "false">`

Note:	The `style:num-format` attribute can be empty.

Display Levels

The `text:display-levels` attribute specifies the number of levels whose numbers are displayed at the current level. For example, it could specify that you display all three numbers (1.2.1) for a level three heading or that you only display two levels (2.1).

XML Code:	`text:display-levels`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-numbering text:display-levels %number; "1">`

Start Value

The `text:start-value` attribute specifies the first number of an list item of the current level.

XML Code:	`text:start-value`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-numbering text:start-value % number; "1">`

3.3.6 Bullet Level Style

A bullet level style element specifies a list style where the list items are preceded by bullets.

XML Code:	`<text:list-level-style-bullet>`
Rules:	This element is contained in list style elements (`<text:list-style>`) only.
DTD:	`<!ELEMENT text:list-level-style-bullet (style:properties?)>`
Note:	The result of including this element in an ordered list is undefined. It can be either an ordered list using a default style or an unordered list using the bullet level style.

The attributes that you can associate with the `<text:list-level-style-bullet>` element are:

- Level, spacing, alignment, and text style
- Font
- Bullet character
- Bullet relative size

Level, Spacing, Alignment, and Text Style

These attributes are the same as those described for use with the number level style, see Section 3.3.5.

Font

The font attributes that can be attached to an item set element are described in Sections 3.10.9 to 3.10.13.

Bullet Character

The bullet character attribute specifies the UNICODE character to use as the bullet in a bullet level style.

XML Code:	`text:bullet-char`
Rules:	The value of this attribute must be *one* UNICODE character.
DTD:	`<!ENTITY % char "CDATA">` `<!ATTLIST text:list-level-style-bullet text:bullet-char %char; #REQUIRED>`

Bullet Relative Size

The `text:bullet-relative-size` attribute specifies a percentage value for the bullet size relative to the font size of the paragraphs in the bullet list. For example, if the value of the `text:bullet-relative-size` attribute is `75`, the bullet used in the list is 75% of the font size for the paragraph.

XML Code:	`text:bullet-relative-size`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-bullet text:bullet-relative-size %percentage; #IMPLIED>`

3.3.7 Image Level Style

An image level style element specifies a list style where the list items are preceded by images.

XML Code:	`<text:list-level-style-image>`
Rules:	This element can be an XLink and can only be contained in list style elements.

DTD:	`<!ELEMENT text:list-level-style-image (style:properties?, office:binary-data?)>`
	`<!ATTLIST text:list-level-style-image xlink:type (simple) #IMPLIED>`
	`<!ATTLIST text:list-level-style-image xlink:show (embed) #IMPLIED>`
	`<!ATTLIST text:list-level-style-image xlink:actuate (onLoad) #IMPLIED>`

Note:	The result of including this element in an ordered list is undefined. It can be either an ordered list using a default style or an unordered list using the image level style.

The elements and attributes that you can associate with the `<text:list-level-style-image>` element are:

- Level, spacing, and alignment
- Image location
- Image size
- Vertical alignment

Level, Spacing, and Alignment

These attributes are the same as those described for use with the number level style, see Section 3.3.5.

Image Location

The image data can be stored in one of the following ways:

- The image data is located in an external file. Use the `xlink:href` attribute described below to specify the location of the file.
- The image data is contained in the `<text:list-level-style-image>` element. The `<text:list-level-style-image>` element must contain an `<office:binary-data>` element that contains the image data in BASE64 encoding. In this situation, the `xlink:href` attribute is not required.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST text:list-level-style-image xlink:href %url; #IMPLIED>`
Note:	Bullet images can be embedded. Currently, embedded images must be stored in separate files.

Image Size

The size of the image is specified by the following attributes:

XML Code:	`fo:width` `fo:height`
Rules:	These attributes are attached to a `<style:properties>` element that is contained in the `<text:list-level-style-image>` element.

See Chapter 2 for more information.

Vertical Alignment

The vertical alignment of the image is specified by the following attributes:

XML Code:	`style:vertical-pos` `style:vertical-rel`
Rules:	These attributes are attached to a `<style:properties>` element that is contained in the `<text:list-level-style-image>` element.

See Chapter 2 for more information.

Example: Image level style

```
<text:list-style style:name="List 1">
  <text:list-level-style-numbering text:level="1"
    fo:num-format="1"/>
  <text:list-level-style-bullet text:level="2"
    text:bullet-char="-"
    text:style-name="Bullet Char"/>
  <text:list-level-style-image text:level="3" xlink:href="bullet.gif">
    <style:properties fo:width=".7cm" fo:height=".7cm"
                      style:vertical-pos="middle" style:vertical-
rel="line"/>
  </text:list-level-style-image>
</text:list-style>
```

The following is the output from the above example:

1. This is the first list item.

   This is a continuation of the first list item.

2. This is the second list item. It contains an unordered sub list.

   - This is a sub list item.
   - This is a sub list item.
   - This is a sub list item.

3. This is the third list item.

3.4 Outline Numbering

You can link outline numbering to paragraph styles.

3.4.1 Outline Style

The way in which OpenOffice.org XML represents outline numbering styles is very similar to the way it represents list styles. The `<text:outline-style>` element contains elements that specify the style of each outline level.

XML Code:	`<text:outline-style>`
Rules:	
DTD:	`<!ELEMENT text:outline-style (text:outline-level-style)+>`

3.4.2 Outline Level Style

The `<text:outline-level-style>` element specifies the style for each outline level.

XML Code:	`<text:outline-level-style>`
Rules:	This element is contained in `<text:outline-style>` elements only.
DTD:	`<!ELEMENT text:outline-level-style EMPTY>`

The attributes that you can associate with the `<text:outline-level-style>` element are:

- Level
- Spacing and alignment
- Text style
- Number format
- Display levels
- Start value

Level

See Section 3.3.5 for a description of this attribute.

Spacing and Alignment

The `<text:outline-level-style>` element contains a `<style:properties>` element that can contain attributes specifying the spacing and alignment for the outline numbering list. The attributes are the same as the attributes for the numbering level style element, `<text:list-level-style-numbering>`, as follows:

- Start indent – `text:space-before`
- Minimum label width – `text:min-label-width`
- Minimum label distance – `text:min-label-distance`
- Label alignment – `fo:text-align`

See Section 3.3.5 for detailed information on these attributes.

Text Style

See Section 3.3.5 for information on the text style attribute.

Number Format

Please refer to Chapter 2 for detailed information on number format attributes. You can associate the attributes described in Chapter 2 with the `<text:outline-numbering-level-style>` element.

DTD:	`<!ATTLIST text:outline-level-style style:num-format CDATA #REQUIRED>`
	`<!ATTLIST text:outline-level-style style:num-prefix CDATA #IMPLIED>`
	`<!ATTLIST text:outline-level-style style:num-suffix CDATA #IMPLIED>`
	`<!ATTLIST text:outline-level-style style:num-letter-sync % boolean; "false">`

Note:	The `style:num-format` attribute can be empty.

Display Levels

See Section 3.3.5 for information on the display level element.

Start Value

See Section 3.3.5 for information on the start value attribute.

3.5 Line Numbering

3.5.1 Line Numbering Configuration

An OpenOffice.org document can contain *none* or *one* line numbering configuration element. If the element is not present, a default line numbering configuration is used. The default line numbering may vary depending on the version of OpenOffice.org software but every document saved using the OpenOffice.org software contains a line numbering configuration element.

XML Code:	`<text:linenumbering-configuration>`
Rules:	
DTD:	`<!ELEMENT text:linenumbering-configuration (text: linenumbering-seperator?)>`

The attributes that you can associate with the `<text:linenumbering-configuration>` element are:

- Line numbering enable
- Number format
- Text style
- Increment

- Position
- Offset
- Count empty lines
- Count in floating frames
- Restart numbering on every page

The element that you can associate with the `<text:linenumbering-seperator>` element is:

- Separator and its associated separator offset attribute

Line Numbering Enable

The `text:number-lines` attribute controls whether or not lines are numbered.

XML Code:	`text:number-lines`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:number-lines % boolean; "true">`

Number Format

See Chapter 2 for detailed information on number formats.

Text Style

The `text:style-name` attribute specifies the text style for all line numbers.

XML Code:	`text:style-name`
Rules:	The value of this attribute is the name of the text style that is applied to all line numbers.
DTD:	`<!ATTLIST text:linenumbering-configuration text:style-name CDATA #IMPLIED>`

Increment

The `text:increment` attribute causes line numbers that are a multiple of the given increment to be numbered. For example, if the increment is 5, only lines number 5, 10, 15, and so on are numbered.

XML Code:	`text:increment`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:increment CDATA #IMPLIED>`

Position

The `text:position` attribute determines whether the line numbers are printed on the left , right, inner, or

outer margins.

XML Code:	`text:position`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:position (left|right|inner|outer) "left">`

Offset

The `text:offset` attribute determines the distance between the line number and the margin.

XML Code:	`text:offset`
Rules:	
DTD:	`<!ATTLIST text:linenumbering-configuration text:offset % nonNegativeLength; #IMPLIED>`

Count Empty Lines

The `text:count-empty-lines` attribute determines whether or not empty lines are included in the line count.

XML Code:	`text:count-empty-lines`
Rules:	If the value of this attribute is `true`, empty lines are included in the line count.
DTD:	`<!ATTLIST text:linenumbering-configuration text:count-empty-lines %boolean; "true">`

Count Lines in Floating Frames

The `text:count-in-floating-frames` attribute determines whether or not text in floating frames is included in the line count.

XML Code:	`text:count-in-floating-frames`
Rules:	If the value of this attribute is `true`, text within floating frames is included in the line count.
DTD:	`<!ATTLIST text:linenumbering-configuration text:count-in-floating-frames %boolean; "false">`

Restart Numbering on Every Page

The `text:restart-on-page` attribute determines whether or not the line count is reset to 1 at the start of every page.

XML Code:	`text:restart-on-page`
Rules:	If the value of this attribute is `true`, the line count is reset to 1 at the beginning of every page, resulting in page -specific numbering of lines. The default value of this attribute is `false`, resulting in document-specific numbering of lines.
DTD:	`<!ATTLIST text:linenumbering-configuration text:restart-on-page %boolean; "false">`

3.5.2 Separator

The <text:linenumbering-seperator> element contains the text that is displayed as a separator.

XML Code:	<text:linenumbering-seperator>
Rules:	This element is contained in the line numbering configuration element. If the element is not present, no separator is displayed.
DTD:	<!ELEMENT text:linenumbering-separator (#PCDATA)>

Separator Offset Attribute

The text:increment attribute specifies the separator offset.

XML Code:	text:increment
Rules:	This attribute is associated with the line numbering separator element.
DTD:	<!ATTLIST text:linenumbering-separator text:increment %number; #REQUIRED>

3.5.3 Line Numbering Properties

Some of the text formatting properties that you can apply to paragraphs and paragraph styles also influence line numbering. These text formatting properties are as follows:

- Line numbering application
- Line number start value

Line Numbering Application

This property controls whether or not paragraph lines are numbered.

XML Code:	text:number-lines
Rules:	This attribute can be contained in an item set element that belongs to a paragraph or paragraph style.
DTD:	<!ATTLIST style:properties text:number-lines %boolean; #IMPLIED>

Line Number Start Value

This property specifies a new start value for line numbering.

XML Code:	text:line-number
Rules:	This attribute can be contained in an item set element that belongs to a paragraph or paragraph style. The attribute is only recognized if there is also a text:number-lines attribute with a value of true in the same item set element.
DTD:	<!ATTLIST style:properties text:line-number %number; #IMPLIED>

3.6 Footnotes and Endnotes

3.6.1 Footnotes Configuration

An OpenOffice.org document contains either *none* or *one* footnotes configuration element. If there is no footnote configuration element, a default footnote configuration is used. Therefore, every saved OpenOffice.org document contains a footnote configuration element.

XML Code:	`<text:footnotes-configuration>`
Rules:	
DTD:	`<!ELEMENT text:footnotes-configuration (text:footnote-continuation-notice-forward?, text:footnote-continuation-notice-backward?)>`

The attributes that you can associate with the `<text:footnotes-configuration>` element are:

- Citation text style
- Citation body text style
- Default footnote paragraph style
- Master page
- Start value
- Number format
- Numbering scheme
- Footnote position

You can include the following element in the `<text:footnotes-configuration>` element:

- Footnote continuation notice (forward and backward)

Citation Text Style

The `text:citation-style` attribute specifies the text style to use for the footnote citation within the footnote.

XML Code:	`text:citation-style`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:citation-style CDATA #IMPLIED>`

Citation Body Text Style

The `text:citation-body-style-name` attribute specifies the text style to use for the footnote citation in the text flow.

XML Code:	`text:citation-body-style-name`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:citation-body-style-name %styleName; #IMPLIED>`

Default Footnote Paragraph Style

The default footnote paragraph style is only used for footnotes that are inserted into an existing document. It is not used for footnotes that already exist.

XML Code:	`text:default-style`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:default-style CDATA #IMPLIED>`

Master Page

To display the footnotes at the end of the document, the pages that contain the footnotes must be instances of the master page specified by the `text:master-page-name` attribute.

XML Code:	`text:master-page-name`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:master-page-name % styleName; #IMPLIED>`

Start Value

The `start:value` attribute specifies the value at which the footnote numbering starts.

XML Code:	`text:start-value`
Rules:	
DTD:	`<!ATTLIST text:footnotes-configuration text:start-value % number; "1">`

Number Format

See Chapter 2 for information on the number format for footnotes.

Numbering Scheme

The `text:start-numbering-at` attribute specifies if footnote numbers start with a new number at the beginning of the document or at the beginning of each chapter or page.

XML Code:	`text:start-numbering-at`
Rules:	The value of this attribute can be `document`, `chapter`, or `page`.
DTD:	`<!ATTLIST text:footnotes-configuration text:start-numbering-at (document|chapter|page) "document">`
Note:	XSLT does not have the capability to start with new footnote numbers on every page.

Footnotes Position

The `text:footnotes-position` attribute specifies one of the following positions for footnotes:

- The bottom of the page where the footnote citation is located.
- The end of the document.

XML Code:	`text:footnotes-position`
Rules:	The value of this attribute can be `page` or `document`.
DTD:	`<!ATTLIST text:footnotes-configuration text:footnotes-position (document|page) "page">`
Note:	XSL does not have the capability to display footnotes at the end of the document. However, you can use an XSLT stylesheet to generate some other flow objects to display such footnotes.

Footnote Continuation

The footnote continuation elements specify:

- Text displayed at the end of a footnote that is continued on the next page
- Text displayed before the continued text

XML Code:	`<text:footnote-continuation-notice-forward>` `<text:footnote-continuation-notice-backward>`
Rules:	These elements can be contained in the footnotes configuration element.
DTD:	`<!ELEMENT text:footnote-continuation-notice-forward (#PCDATA)>` `<!ELEMENT text:footnote-continuation-notice-backward (#PCDATA)>`
Note:	XSL and XSLT do not support footnote continuation.

Example: Footnote configuration in OpenOffice.org XML

```
<text:footnotes-configuration text:citation-style="Footnote symbol"
                              text:default-style="Footnote">
  <text:footnote-continuation-notice-forward>" .."
  </text:footnote-continuation-notice-forward>
  <text:footnote-continuation-notice-forward>".. "
  </text:footnote-continuation-notice-forward>
</text:footnotes-configuration>
```

3.6.2 Endnotes Configuration

An OpenOffice.org document contains either *none* or *one* endnotes configuration element.

XML Code:	`<text:endnotes-configuration>`
Rules:	
DTD:	`<!ELEMENT text:endnotes-configuration EMPTY>`

Citation Text Style, Default Endnote Paragraph Style, Page Master, Offset, and Number Format

See Section 3.6.1 for descriptions of these attributes. The application of these attributes to the endnote configuration element is the same as for the footnote configuration element.

```
DTD:        <!ATTLIST text:endnotes-configuration
                      text:citation-body-style-name %styleName; #IMPLIED
                      text:citation-style-name %string; #IMPLIED
                      text:default-style-name %styleName; #IMPLIED
                      text:page-master-name %styleName; #IMPLIED
                      text:start-value %number; "0"
                      style:num-format %string; #IMPLIED
                      style:num-prefix %string; #IMPLIED
                      style:num-suffix %string; #IMPLIED>
```

3.6.3 Footnotes

The footnote element contains the footnote citation element and the elements that make up the footnote content.

OpenOffice.org XML represents footnotes in a similar fashion to XSL. In XSL, the first child of the footnote element contains the citation in the form of an `<fo:inline>` element. OpenOffice.org XML uses the same structure but introduces a `text:footnote-citation` element. The second child contains the footnote body, just as in XSL.

Additionally, OpenOffice.org features the `<text:footnotes-configuration>` element. To achieve a similar effect to the footnote configuration in XSL, every footnote and footnote citation element must be formatted appropriately.

XML Code:	`<text:footnote>`
Rules:	
DTD:	`<!ELEMENT text:footnote (text:footnote-citation, text:footnote-body)>`

Footnote Citation

The `<text:footnote-citation>` element specifies the formatted footnote number or characters.

XML Code:	`<text:footnote-citation>`
Rules:	This element is contained in the footnote element (`<text:footnote>`) and it contains the formatted footnote number as text.
DTD:	`<!ELEMENT text:footnote-citation (#PCDATA)>`

Footnote Label

Footnote citation elements can be labeled or numbered. If they are numbered, the number is chosen automatically according to the footnotes configuration element. If they are labeled, the user must supply a label for every footnote he/she inserts into the document. This label is stored in the `text:label` attribute of the `<text:footnote-citation>` element.

XML Code:	`text:label`
Rules:	If this attribute is not present, the footnote is numbered automatically.
DTD:	`<!ATTLIST text:footnote-citation text:label %string; #IMPLIED>`

Footnote Reference ID

The footnote reference ID is used by references to footnotes to identify the footnote that is referenced.

XML Code:	`text:id`
Rules:	This attribute is used by the `<text:footnote>` element. It contains a value of type `ID`, where `ID` is a predefined XML attribute type.
DTD:	`<!ATTLIST text:footnote text:id ID #IMPLIED>`

Footnote Body

The `<text:footnote-body>` element contains the actual content of the footnote. It does not have any attributes.

XML Code:	`<text:footnote-body>`
Rules:	
DTD:	`<!ELEMENT text:footnote-body %inline-text;>`

Examples: Footnotes

```
<text:p>
   This paragraph contains a footnote
   <text:footnote text:id="ftn001">
      <text:footnote-citation>
         1
      </text:footnote-citation>
      <text:footnote-body>
         <text:p>
            This footnote has a generated sequence number
         </text:p>
      </text:footnote-body>
   </text:footnote>
   .
</text:p>
<text:p>
   This paragraph contains a footnote
   <text:footnote text:id="ftn002">
      <text:footnote-citation text:label="*">
         *
      </text:footnote-citation>
      <text:footnote-body>
         <text:p>
            This footnote has a fixed citation
         </text:p>
      </text:footnote-body>
   </text:footnote>
   , too
</text:p>
```

3.6.4 Endnotes

Endnotes are represented in the same way as footnotes. They contain the endnote citation element and the endnote body element that makes up the endnote content. For a full description of the elements and attributes associated with the `<text:endnote>` element, please refer to the previous section.

XML Code:	`<text:endnote>`
Rules:	
DTD:	`<!ELEMENT text:endnote (text:endnote-citation, text:endnote-body)>` `<!ATTLIST text:endnote text:id ID #IMPLIED>`
Limitations:	XSL does not support endnotes but you can use an XSLT stylesheet to generate some other flow objects to display endnotes.

Endnote Citation

The `<text:endnote-citation>` element specifies the formatted endnote number or characters.

XML Code:	`<text:endnote-citation>`
Rules:	
DTD:	`<!ELEMENT text:endnote-citation (#PCDATA)>` `<!ATTLIST text:endnote-citation text:label %string; #IMPLIED>`

Endnote Body

The <text:endnote-body> element is defined as follows:

XML Code:	<text:endnote-body>
Rules:	
DTD:	<!ELEMENT text:endnote-body %inline-text;>

3.7 Fields

OpenOffice.org text documents or OpenOffice.org text content embedded in other types of documents can contain variable text elements called fields. There are several different types of field, each of which implements a different type of variable text element. Fields are most commonly used for:

- Page numbers
  A page number field displays the number of the page it appears on. This field is useful for footers. For every page on which the footer appears, the field assumes the current page number so that all pages are numbered correctly.
- Creation dates
  A creation date field displays the date on which the current document was created. This field is useful for document templates. Every document created using the template contains the date when it was created.
- Number ranges
  A number range field allows the user to number certain elements, for example, images or tables. A number range field displays its own position in relation to the other number range fields for the same range. Therefore, if you move an image and its associated number range field within a document, the fields are automatically updated to reflect the new order.

This section describes how OpenOffice.org software represents fields in the XML file format.

3.7.1 Common Characteristics of Field Elements

Each field type is represented by a corresponding element type. A field in a document is encoded as a single element of the appropriate type. The content of the element is the textual representation of the current field value as it is displayed in the OpenOffice.org user interface. Therefore, ignoring all field elements and displaying only the textual content of the elements provides an approximate text-only version of the document.

The value of a field is usually stored in an attribute. It is necessary to store the value so that the presentation of the field can be recomputed if necessary, for example, if the user decides to change the formatting style of the field. It is also necessary to store the presentation style of the element content, to facilitate easy processing of the XML document. For example, if complete processing of a field is impossible or undesirable, the application can ignore the field and use only the content in this situation. For string values, if the value is identical to the presentation, the value attribute is omitted to avoid duplicate storage of information.

For fields that can store different types of content, for example, numbers, strings, or dates, a value type is stored in addition to the actual value. The value and value type attributes are explained later in Section 3.7.43. If more information is needed to restore a field, it is stored in additional attributes.

The most common attributes of field elements are:

- Fixed fields
  Many fields have a variant where the content does not change after the initial value is assigned. These fields are generally marked by the attribute text:fixed. See Section 3.7.43 for more information on this

attribute.

- Formatting style
  Several field types, particularly those representing number, date, or time data, contain a formatting style. In OpenOffice.org XML, this formatting style is represented by a `style:data-style-name` attribute. Since the user can change the presentation style for fields, OpenOffice.org must be able to recompute a new representation of the field content at any time. See Section 3.7.43 for more information on this attribute.

3.7.2 Document Fields

OpenOffice.org Writer fields can display information about the current document or about a specific part of the current document, such as the author, the current page number, or the document creation date. These fields are collectively referred to as document fields.

Document fields are often fixed. A field can be marked fixed to indicate that its content is preserved, rather than re-evaluated, when the document is edited. For example, a date field shows the current date. If the date field is marked fixed, the value of the field is preserved during subsequent edits and always reflects the original date on which the field was inserted into the document. If the field is not marked fixed, its value changes whenever the document is edited. In the same way, the author field can show the original author or the last author of a document, depending on whether the field is marked fixed or not.

The group of document fields includes:

- Date and time fields
- Sender and author fields
- Page number fields
- Chapter fields
- File name fields
- Document template fields
- Statistics fields[1]

3.7.3 Date Fields

Date fields display the current date. You can adjust the date to display a date other than the current date. For example, you can change the date on a document that was edited late at night so that it displays the date of the following day or several days later.

XML Code:	`<text:date>`
Rules:	This element contains the presentation of the date field value, depending on the data style specified. The default date is the current date. You can preserve the value of this element using the `text:fixed` attribute described in Section 3.7.43.
DTD:	`<!ELEMENT text:date (#PCDATA)>`

The attributes that you can associate with the `<text:date>` element are:

- Date value
- Date adjustment

1 These fields are not currently part of the OpenOffice.org XML file format.

- Fixed (see Section 3.7.43)
- Formatting style (see Section 3.7.43). The formatting style must be a date data style, see Section 2.5.4 for more information.

Date Value

The `text:date-value` attribute specifies a particular date value. For example, if the date field is marked fixed, you can use this attribute to specify the date on which the field was marked as fixed. You can also use this attribute to specify a future date. Some applications support date and time in addition to date-only values, in accordance with ISO 8601.

XML Code:	`text:date-value`
Rules:	The date value should conform with the extended date format described in Section 5.2.1.1 of ISO 8601. See page 24 for a pointer to ISO 8601. If no value is specified, the current date is assumed, even if the field is marked fixed.
DTD:	`<!ATTLIST text:date text:date-value %timeInstance; #IMPLIED>`

Date Adjustment

You can adjust the value of a date field by a certain time period, which you specify using the `text:date-adjust` attribute. OpenOffice.org Writer truncates the specified time period to a period of full days and adds it to the value of the date field. If the time period is negative, OpenOffice.org Writer subtracts it from the value of the date field yielding a date before the current date.

XML Code:	`text:date-adjust`
Rules:	The value of this attribute must conform to the time period format described in Section 5.5.3.2 of ISO 8601. See page 24 for a pointer to ISO 8601. The value can be preceded by an optional minus sign to indicate a negative time duration.
DTD:	`<!ATTLIST text:date text:date-adjust %c-duration; #IMPLIED>`
Implementation limitation:	Currently, OpenOffice.org does not support month or year durations.

3.7.4 Time Fields

Time fields display the current time. They are very similar to the date fields described in the previous section, supporting the same attributes except that for time fields, they are called `text:time-value` and `text:time-adjust` attributes.

XML Code:	`<text:time>`
Rules:	This element contains the presentation of the time field value, depending on the data style specified. The default time is the current time. You can preserve the value of this element using the `text:fixed` attribute described in Section 3.7.43.
DTD:	`<!ELEMENT text:time (#PCDATA)>`

The attributes that you can associate with the `<text:time>` element are:

- Time value

- Time adjustment
- Fixed (see Section 3.7.43)
- Formatting style (see Section 3.7.43). The formatting style must be a time data style, see Section 2.5.5 for more information.

Time Value

The `text:time-value` attribute records the time at which the document was last edited.

XML Code:	`text:time-value`
Rules:	The value of this attribute must conform with the extended time format described in Section 5.4.1 of ISO 8601. See page 24 for a pointer to ISO 8601. If no value is specified, the current time is assumed, even if the field is marked `fixed`.
DTD:	`<!ATTLIST text:time text:time-value %time; #IMPLIED>`

Time Adjustment

You can adjust the value of a time field by a certain time period, which you specify using the `text:time-adjust` attribute. The OpenOffice.org software truncates the time period to a period of full minutes and adds it to thevalue of the time field. If the time period is negative, the OpenOffice.org software subtracts it from the value of the time field yielding a time in the past.

XML Code:	`text:time-adjust`
Rules:	The value of this attribute must conform to the time period format described in Section 5.5.3.2 of ISO 8601. See page 24 for a pointer to ISO 8601. The value can be preceded by an optional minus sign to indicate a negative time duration. Positive values adjust the time to a time in the future, while negative values adjust the time to a time in the past. The duration is truncated to full minutes.
DTD:	`<!ATTLIST text:time text:time-adjust %c-duration; #IMPLIED>`

Example: Time adjust attributes and their effects

If the attribute `text:time-adjust="PTM15"`, the time field displays a time which is 15 minutes later than the actual time specified by the time field value.

If the attribute `text:time-adjust="-PTH1"`, the time field displays a time which is one hour before the actual time specified by the time field value.

3.7.5 Page Number Fields

Page number fields display the current page number. These fields are particularly useful for recurring content, such as headers and footers. If you insert a page number field into a footer, the current page number is displayed on every page on which the footer appears.

XML Code:	`<text:page-number>`
Rules:	If the `text:page-adjust` attribute is used, the page number of the page at the specified offset is displayed, if it exists.
DTD:	`<!ELEMENT text:page-number (#PCDATA)>`

The attributes that you can associate with the <text:page-number> element are:

- Page adjustment
- Display previous or following page numbers
- Fixed (see Section 3.7.43)
- Formatting style (see Section 3.7.43)
  Page numbers can be formatted according to the number format described in Section 2.9. If a number style is not specified, the page numbers are formatted according to the number style defined in the current page style.

Page Adjustment

You can adjust the value of a page number field by a specified number, allowing you to display the page numbers of following or preceding pages. You specify the adjustment number using the text:page-adjust attribute. When you use this attribute, the application:

1. Adds the value of the attribute to the current page number.
2. Checks to see if the resulting page exists.
3. If the page exists, the number of that page is displayed.
4. If the page does not exist, the value of the page number field remains empty and no number is displayed.

XML Code:	text:page-adjust
Rules:	
DTD:	<!ATTLIST text:page-number text:page-adjust %integer; #IMPLIED>

Display Previous or Following Page Numbers

The text:select-page attribute allows you to display the number of the previous or the following page rather than the number of the current page.

XML Code:	text:select-page
Rules:	The value of this attribute can be previous, current, or next.
DTD:	<!ATTLIST text:page-number text:select-page (previous | current | next) "current">

Note: To display the current page number on all pages except the first or last page, you should use a combination of the text:select-page and text:page-adjust attributes.

Example: Displaying the current page number on all pages except the first page

```
<text:page-number text:select-page="previous" text:page-adjust="1" text:
num-format="1"/>
```

3.7.6 Page Continuation Text

In some publications, a continuation reminder is printed at the bottom of the page in addition to the page number. To include a continuation reminder, use the <text:page-continuation> element.

XML Code:	`<text:page-continuation>`
Rules:	
DTD:	`<!ELEMENT text:page-continuation (#PCDATA)>`

The attributes associated with the `<text:page-continuation>` element are:

- Previous or following page
- String value

Previous or Following Page

This attribute specifies whether to check for a previous or next page and if the page exists, the continuation text is printed.

XML Code:	`text:select-page`
Rules:	The value of this attribute can be `previous` or `next`.
DTD:	`<!ATTLIST text:page-continuation text:select-page ( previous | next ) #REQUIRED>`

String Value

This attribute specifies the continuation text to display.

XML Code:	`text:string-value`
Rules:	If this attribute is omitted, the element content is used.
DTD:	`<!ATTLIST text:page-continuation text:string-value CDATA #IMPLIED>`

3.7.7 Sender Fields

There are several fields which contain information about the sender of the current document, for example, name and email address. The information about the sender is taken from the OpenOffice.org user information dialog. If a sender field is marked fixed using the `text:fixed` attribute, the original sender information in the sender fields is preserved. Otherwise, the information is updated each time the file is edited, causing the fields to change value when the document is edited by a different user.

First Name

This element represents the first name of the sender.

XML Code:	`<text:sender-firstname>`
Rules:	
DTD:	`<!ELEMENT text:sender-firstname (#PCDATA)>` `<!ATTLIST text:sender-firstname text:fixed %boolean "true">`

Last Name

This element represents the last name of the sender.

XML Code:	`<text:sender-lastname>`
Rules:	
DTD:	`<!ELEMENT text:sender-lastname (#PCDATA)>` `<!ATTLIST text:sender-lastname text:fixed %boolean "true">`

Initials

This element represents the initials of the sender.

XML Code:	`<text:sender-initials>`
Rules:	
DTD:	`<!ELEMENT text:sender-initials (#PCDATA)>` `<!ATTLIST text:sender-initials text:fixed %boolean "true">`

Title

This element represents the title of the sender.

XML Code:	`<text:sender-title>`
Rules:	
DTD:	`<!ELEMENT text:sender-title (#PCDATA)>` `<!ATTLIST text:sender-title text:fixed %boolean; "true">`

Position

This element represents the position of the sender.

XML Code:	`<text:sender-position>`
Rules:	
DTD:	`<!ELEMENT text:sender-position (#PCDATA)>` `<!ATTLIST text:sender-position text:fixed "true">`

Email Address

This element represents the email address of the sender.

XML Code:	`<text:sender-email>`
Rules:	
DTD:	`<!ELEMENT text:sender-email (#PCDATA)>` `<!ATTLIST text:sender-email text:fixed "true">`

Private Telephone Number

This element represents the private telephone number of the sender.

XML Code:	`<text:sender-phone-private>`
Rules:	
DTD:	`<!ELEMENT text:sender-phone-private (#PCDATA)>` `<!ATTLIST text:sender-phone-private text:fixed "true">`

Fax Number

This element represents the facsimile number of the sender.

XML Code:	`<text:sender-fax>`
Rules:	
DTD:	`<!ELEMENT text:sender-fax (#PCDATA)>` `<!ATTLIST text:sender-fax text:fixed "true">`

Company Name

This element represents the name of the company that employs the sender.

XML Code:	`<text:sender-company>`
Rules:	
DTD:	`<!ELEMENT text:sender-company (#PCDATA)>` `<!ATTLIST text:sender-company text:fixed "true">`

Office Telephone Number

This element represents the office telephone number of the sender.

XML Code:	`<text:sender-phone-work>`
Rules:	
DTD:	`<!ELEMENT text:sender-phone-work (#PCDATA)>` `<!ATTLIST text:sender-phone-work text:fixed "true">`

Street

This element represents the street name of the address of the sender.

XML Code:	`<text:sender-street>`
Rules:	
DTD:	`<!ELEMENT text:sender-street (#PCDATA)>` `<!ATTLIST text:sender-street text:fixed "true">`

City

This element represents the city name of the address of the sender.

XML Code:	`<text:sender-city>`
Rules:	
DTD:	`<!ELEMENT text:sender-city (#PCDATA)>` `<!ATTLIST text:sender-city text:fixed "true">`

Postal Code

This element represents the postal code of the address of the sender.

XML Code:	`<text:sender-postal-code>`
Rules:	
DTD:	`<!ELEMENT text:sender-postal-code (#PCDATA)>` `<!ATTLIST text:sender-postal-code text:fixed "true">`

Country

This element represents the country of the address of the sender.

XML Code:	`<text:sender-country>`
Rules:	
DTD:	`<!ELEMENT text:sender-country (#PCDATA)>` `<!ATTLIST text:sender-country text:fixed "true">`

State or Province

This element represents the state or province of the address of the sender, if applicable.

XML Code:	`<text:sender-state-or-province>`
Rules:	
DTD:	`<!ELEMENT text:sender-state-or-province (#PCDATA)>` `<!ATTLIST text:sender-state-or-province text:fixed true">`

3.7.8 Author Fields

There are two OpenOffice.org XML elements available to represent the author of a document. One element displays the full name of the author and the other element displays the initials of the author.

You can fix the value of author fields using the `text:fixed` attribute. Marking an author field as fixed preserves the original field content. Otherwise, the field content changes each time the document is updated, to reflect the last author of the document.

Name of the Author

This element represents the full name of the author.

XML Code:	`<text:author-name>`
Rules:	
DTD:	`<!ELEMENT text:author-name (#PCDATA)>` `<!ATTLIST text:author-name text:fixed "true">`

Initials of the Author

This element represents the initials of the author.

XML Code:	`<text:author-initials>`
Rules:	
DTD:	`<!ELEMENT text:author-initials (#PCDATA)>` `<!ATTLIST text:author-initials text:fixed "true">`

3.7.9 Placeholders

OpenOffice.org Writer uses placeholder fields to indicate locations in a document where the user must fill in some information. For example in a letter template, you can have a section of the document reserved for the address of the recipient. A placeholder field displays text informing the user about the purpose of the placeholder and sometimes includes a description. Placeholder fields can represent different text elements, such as text or tables.

XML Code:	`<text:placeholder>`
Rules:	This element contains some brief text which is displayed with the placeholder.
DTD:	`<!ELEMENT text:placeholder (#PCDATA)>`

The attributes that you can associate with the `<text:placeholder>` element are:

- Placeholder type
- Placeholder description

Placeholder Type

There are five different types of placeholder, representing the five possible types of content: text, tables, text boxes, images, or objects. The `text:placeholder-type` attribute represents the content type.

XML Code:	`text:placeholder-type`
Rules:	This attribute is mandatory and it indicates which type of text content the placeholder represents. The value of the attribute can be `text`, `text-box`, `image`, `table`, or `object`.
DTD:	`<!ATTLIST text:placeholder text:placeholder-type ( text | table | text-box | image | object ) #REQUIRED>`

Placeholder Description

In addition to the brief text stored in the element content, you can associate a `text:description` attribute with the placeholder element. This attribute is optional. The purpose of the attribute is to contain a more elaborate description of the purpose of the placeholder than the description stored in the element content. See Section 3.7.43 for information on using the `text:description` attribute.

DTD: `<!ATTLIST text:placeholder text:description %string; #IMPLIED>`

3.7.10 Database Fields

OpenOffice.org Writer documents can connect to OpenOffice.org Base databases and display database information as text content. To display database information, OpenOffice.org Writer uses a group of text fields, collectively called database fields. OpenOffice.org Base can use database tables from SQL servers, therefore you can use database fields to access any SQL database, provided that the appropriate drivers are available.

In OpenOffice.org Base, a database contains the following components:

- Tables, which store the actual data.
- Queries, which extract a subset of data from one or more tables.
- Forms, which present the data.
- Reports, which summarize the database content.

Database forms and reports are not relevant to XML text content, therefore they are not discussed in this chapter. From the point of view of embedding database information in OpenOffice.org text documents, queries and tables are considered the same. Therefore for the remainder of this section, the phrase *database table* refers to both database tables and database queries.

Every database in OpenOffice.org Base has a name and this name is used by all of the OpenOffice.org components to identify a database. All database fields contain a database name and most database fields also contain the name of a database table, which must be stored in the named database. An additional attribute determines whether the database table refers to an SQL table, an OpenOffice.org query, or the result of an SQL command.

The following entity is defined for database fields:

XML Code:	`%database-table;`

DTD:	`<!ENTITY % database-table`
	`"text:database-name CDATA #REQUIRED`
	`text:table-name CDATA #REQUIRED"`
	`text:table-type (table|query|command)`
	`#IMPLIED" >`

Example:	`<!ATTLIST database-element %database-table;>`

Database fields alone do not retrieve information from a database. In addition to the database fields, a set of database rows is also added to the document. When new data is added to the document, all database fields belonging to the added database table are updated. Using the OpenOffice.org user interface, you can add database rows in one of the following ways:

- Manually, using the Beamer and the Data to Fields function.
- Using the Form Letter menu item on the File menu. This menu item adds each row in the chosen data set into a newly created copy of the form letter.

To display data from a database table use the `<text:database-display>` element. Using the `<text:database-select>` and `<text:database-next>` elements, you can determine which row within the

current selection to display. You can display the current row number for a particular table using the <text:database-row-number> element. Finally, the <text:database-name> field displays the name of the most recently used database, which is the address book file database by default.

3.7.11 Displaying Database Content

The <text:database-display> element displays data from a database. When a new data set is added to a document, all fields that display data from that database table update their content.

XML Code:	<text:database-display>
Rules:	
DTD:	<!ELEMENT text:database-display (#PCDATA)>

The attributes that you can associate with the <text:database-display> element are:

- text:database-name, text:table-name and text:table-type

  These attributes specify the database and database table that this field uses.

DTD:	<!ATTLIST text:database-display %database-table;>

- text:database-column-name

  See the section *Column Name* for information about this attribute.

- style:data-style-name

  If the column specifies a numeric, boolean, date, or time value, the data is formatted according to the appropriate data style. If no data style is specified, the data style assigned to this column in OpenOffice.org Base is used. See Section 3.7.43 for more information about using this attribute.

DTD:	<!ATTLIST text:database-display style:data-style-name % style-name;>

Column Name

The text:column-name attribute specifies the column from which to display the data.

XML Code:	text:column-name
Rules:	The value of this attribute must be a column contained in the specified database.
DTD:	<!ATTLIST text:database-display text:column-name CDATA #REQUIRED>

3.7.12 Selecting the Next Database Row

The <text:database-next> element changes the row in the current selection which is used for display in all following <text:database-display> fields. The next row from the current selection is chosen if it satisfies a given condition. If the next row is wanted regardless of any condition, the condition may be omitted or set to true.

XML Code:	<text:database-next>
Rules:	
DTD:	<!ELEMENT text:database-next EMPTY>

The attributes that you can attach to the <text:database-next> are:

- text:database-name, text:table-name and text:table-type

  These attributes specify the database and the database table that this field uses.

DTD:	<!ATTLIST text:database-next %database-table;>

- text:condition

  See the section *Condition* for information about this attribute.

Condition

The text:condition attribute specifies the condition expression. The expression is evaluated and if the result interpreted as a boolean value is true, the next row is used as the new current row. Please note that you can use database field values in the expression by enclosing in square brackets, the database name, the table name, and the column name, separated by dots.

If the text:condition attribute is not present, OpenOffice.org assumes the formula true, meaning that the next row is selected unconditionally.

XML Code:	text:condition
Rules:	
DTD:	<!ATTLIST text:database-next text:condition %formula; #IMPLIED>

Example:

```
text:formula='[address book file.address.FIRSTNAME] == "Julie"'
```

This example specifies a condition that is true if the current row from the OpenOffice.org address book is the address for a person named Julie. If the condition shown in this example is used in a <text:database-next> element, the following happens:

- The <text:database-display> elements display the data from the first row of the current selection.
- If the FIRSTNAME column of the current row reads Julie, the current row is changed. Otherwise, nothing happens.
- If the first row is Julie, the following <text:database-display> elements display data from the second row. Otherwise, they display data from the first row.

See Section 3.7.43 for more information on the formula syntax of a text:condition attribute, which is the same as that of the text:formula attribute.

3.7.13 Selecting a Row Number

The <text:database-row-select> element selects a specific row from the current selection. As with the <text:database-row-next> element, you can specify a condition so that the given row is only selected if

the condition is `true`.

XML Code:	`<text:database-row-select>`
Rules:	
DTD:	`<!ELEMENT text:database-row-select EMPTY>`

The attributes that you can associate with the `<text:database-row-select>` are:

- `text:database-name`, `text:table-name` and `text:table-type`

  These attributes determine the database and the database table that this field uses.

DTD:	`<!ATTLIST text:database-row-select %database-table;>`

- `text:condition`

  This attribute specifies the condition expression. See Section 3.7.12 for a full explanation of how to use this attribute.

DTD:	`<!ATTLIST text:database-row-select text:condition %formula; #IMPLIED>`

- `text:row-number`

  See the section *Selecting the Row Number* for information about this attribute.

Selecting the Row Number

This attribute specifies the row number to select when a condition is `true`.

XML Code:	`text:row-number`
Rules:	
DTD:	`<!ATTLIST text:database-row-select text:row-number % integer; #REQUIRED>`

3.7.14 Displaying the Row Number

The `<text:database-row-number>` element displays the current row number for a given table. Note that the element displays the actual row number from the database and not the row number of the current selection that is used as an attribute value in the `<text:database-row-select>` element.

XML Code:	`<text:database-row-number>`
Rules:	
DTD:	`<!ELEMENT text:database-row-number (#PCDATA)>`

The attributes that you can associate with the `<text:database-row-number>` are:

- `text:database-name`, `text:table-name` and `text:table-type`

  These attributes determine the database and the database table that this field uses.

DTD:	`<!ATTLIST text:database-row-number %database-table; #REQUIRED>`

- `text:num-format` and `text:num-letter-sync`

  These attributes determine how the number should be formatted. See Section 3.7.12 for more information on how to use this attribute.

DTD:	`<!ATTLIST text:database-row-number %num-format; #IMPLIED>`

- `text:value`

  This attribute specifies the current row number. The number changes when new data is added to the current document.

DTD:	`<!ATTLIST text:database-row-number text:value %integer; #IMPLIED>`

3.7.15 Display Current Database and Table

OpenOffice.org keeps track of the last database and table that was used in the document. In other words, the table that is used by the last field that was inserted into the document. In the OpenOffice.org user interface, the database is displayed in the Beamer. The `<text:database-name>` element displays the database and table name of the most recently used table.

XML Code:	`<text:database-name>`
Rules:	
DTD:	`<!ELEMENT text:database-name (#PCDATA)>`

The attributes that you can associate with the `<text:database-name>` element are:

- `text:database-name`, `text:table-name` and `text:table-type`

  These attributes determine the database and the database table that this field uses.

DTD:	`<!ATTLIST text:database-name %database-table;>`

3.7.16 Metadata Fields

Metadata fields display meta information about the document, such as, the document creation date or the time at which the document was last printed. The names of the metadata field elements correspond to the metadata elements described in Chapter 2.

All metadata field elements can be marked as fixed using the `text:fixed` attribute.

Several metadata fields display a date or a time. The elements for these fields require an associated `text:date-value` or a `text:time-value` attribute, and optionally, they can also have a `style:data-style-name` attribute. See Section 3.7.43 for more information on these attributes.

Initial Creator

This element represents the name of the author who created the original document.

XML Code:	`<text:initial-creator>`
Rules:	

XML Code:	`<text:initial-creator>`
DTD:	`<!ELEMENT text:initial-creator (#PCDATA)>` `<!ATTLIST text:initial-creator text:fixed %boolean; "false">`

Document Creation Date

This element represents the date on which the document was created.

XML Code:	`<text:creation-date>`
Rules:	

DTD:	`<!ELEMENT text:creation-date (#PCDATA)>`
	`<!ATTLIST text:creation-date`
	`text:fixed %boolean; "false"`
	`text:date-value %date; #IMPLIED`
	`style:data-style-name %style-name; #IMPLIED>`

Document Creation Time

This element represents the time at which the document was created.

XML Code:	`<text:creation-time>`
Rules:	

DTD:	`<!ELEMENT text:creation-time (#PCDATA)>`
	`<!ATTLIST text:creation-time`
	`text:fixed %boolean; "false"`
	`text:time-value %timeInstance; #IMPLIED`
	`style:data-style-name %style-name; #IMPLIED>`

Document Description

This element contains a brief description of the document.

XML Code:	`<text:description>`
Rules:	

DTD:	`<!ELEMENT text:description (#PCDATA)>`
	`<!ATTLIST text:description`
	`text:fixed %boolean; "false">`

User-Defined Document Information

This group of elements contains user-defined information about the document. The fields are not used or interpreted by OpenOffice.org, so the user may use these elements for any purpose.

XML Code:	`<text:user-defined>`
Rules:	

XML Code:	`<text:user-defined>`

DTD:	`<!ELEMENT text:user-defined (#PCDATA)>`
	`<!ATTLIST text:user-defined`
	`text:name %string; #REQUIRED`
	`text:fixed %boolean; "false">`

Print Time

This element represents the time at which the document was last printed.

XML Code:	`<text:print-time>`
Rules:	

DTD:	`<!ELEMENT text:print-time (#PCDATA)>`
	`<!ATTLIST text:print-time`
	`text:fixed %boolean; "false"`
	`text:time-value %timeInstance; #IMPLIED`
	`style:data-style-name %style-name; #IMPLIED>`

Print Date

This element represents the date on which the document was last printed.

XML Code:	`<text:print-date>`
Rules:	

DTD:	`<!ELEMENT text:print-date (#PCDATA)>`
	`<!ATTLIST text:print-date`
	`text:fixed %boolean; "false`
	`text:date-value %date; #IMPLIED`
	`style:data-style-name %style-name; #IMPLIED>`

Printed By

This element represents name of the last person who printed the document.

XML Code:	`<text:printed-by>`
Rules:	

DTD:	`<!ELEMENT text:printed-by (#PCDATA)>`
	`<!ATTLIST text:printed-by`
	`text:fixed %boolean; "false">`

Document Title

This element represents the title of the document.

XML Code:	`<text:title>`
Rules:	

DTD:	`<!ELEMENT text:title (#PCDATA)>`
	`<!ATTLIST text:title`
	`text:fixed %boolean; "false">`

Document Subject

This element represents the subject of the document.

XML Code:	`<text:subject>`
Rules:	

DTD:	`<!ELEMENT text:subject (#PCDATA)>`
	`<!ATTLIST text:subject`
	`text:fixed %boolean; "false">`

Document Keywords

This element contains a list of keywords used to describe the document.

XML Code:	`<text:keywords>`
Rules:	

DTD:	`<!ELEMENT text:keywords (#PCDATA)>`
	`<!ATTLIST text:keywords`
	`text:fixed %boolean; "false">`

Document Revision Number

This element contains the document revision number. When the document is created, the revision number is set to 1. Each time the document is saved, the document revision number is incremented.

XML Code:	`<text:editing-cycles>`
Rules:	

DTD:	`<!ELEMENT text:editing-cycles (#PCDATA)>`
	`<!ATTLIST text:editing-cycles`
	`text:fixed %boolean; "false">`

Note: Since the `<text:editing-cycles>` field can not be formatted, the revision number can be read from the element content. Therefore, no extra attribute is needed.

Document Edit Duration

Every time a document is edited, OpenOffice.org records the duration between the time the document is opened and the time the document is closed. It then adds the duration to an internal counter, thereby keeping track of the total time that has been spent editing the document.

XML Code:	`<text:editing-duration>`
Rules:	

XML Code:	`<text:editing-duration>`

DTD:	`<!ELEMENT text:editing-duration (#PCDATA)>`
	`<!ATTLIST text:editing-duration`
	`text:fixed %boolean; "false"`
	`text:duration %timeDuration; #IMPLIED`
	`style:data-style-name %style-name; #IMPLIED>`

Document Modification Time

This element represents the time at which the document was last modified.

XML Code:	`<text:modification-time>`
Rules:	

DTD:	`<!ELEMENT text:modification-time (#PCDATA)>`
	`<!ATTLIST text:modification-time`
	`text:fixed %boolean; "false"`
	`text:time-value %timeInstance; #IMPLIED`
	`style:data-style-name %style-name; #IMPLIED>`

Note:	This element displays the information from the `<meta:date>` element. The name was chosen to avoid confusion with `<text:date>` fields.

Document Modification Date

This element represents the date on which the document was last modified.

XML Code:	`<text:modification-date duration>`
Rules:	

DTD:	`<!ELEMENT text:modification-date (#PCDATA)>`
	`<!ATTLIST text:modification-date`
	`text:fixed %boolean; "false"`
	`text:date-value %date; #IMPLIED`
	`style:data-style-name %style-name; #IMPLIED>`

Document Modified By

This element represents the name of the person who last modified the document.

XML Code:	`<text:creator>`
Rules:	

DTD:	`<!ELEMENT text:creator (#PCDATA)>`
	`<!ATTLIST text:creator`
	`text:fixed %boolean; "false">`

3.7.17 Conditional Text Fields

Text fields can be used to display one text or another, depending on the condition. Conditional text fields are given a condition and two text strings. If the condition is true, one of the text strings is displayed. If the condition is false, the other text string is displayed.

XML Code:	`<text:conditional-text>`
Rules:	
DTD:	`<!ELEMENT text:conditional-text (#PCDATA)>`

The attributes that you can associate with the `<text:conditional-text>` element are:

- Condition
- Text to display if the condition is true
- Text to display if the condition is false
- Current condition

The `text:condition` attribute contains a boolean expression. Depending on the result, the value of the `text:display-if-true` or `text:display-if-false` attribute is displayed.

XML Code:	`text:condition`
Rules:	
DTD:	`<!ATTLIST text:conditional-text text:condition % formula; #REQUIRED>`

Text to Display if the Condition is True

The `text:string-value-if-true` attribute contains the text string to display if the condition is `true`.

XML Code:	`text:string-value-if-true`
Rules:	If the condition is `true`, the value of this attribute is displayed.
DTD:	`<!ATTLIST text:conditional-text text:string-value-if-true %string; #REQUIRED>`

Text to Display if the Condition is False

The `text:string-value-if-false` attribute contains the text string to display if the condition is `false`.

XML Code:	`text:string-value-if-false`
Rules:	If the condition evaluates to `false`, the value of this attribute is displayed.
DTD:	`<!ATTLIST text:conditional-text text:string-value-if-false %string; #REQUIRED>`

Current Condition

The `text:current-value` attribute contains the evaluation result of the condition given by the expression in the `text:condition` attribute. Explicitly giving the result allows applications to delay evaluating the result until necessary. This attribute is valuable for the following reasons:

- If the expression is costly to evaluate, for example, the expression contains references to several databases.
- To allow transformations to correctly display the state of the document without having to parse and evaluate the condition.

XML Code:	`text:current-value`
Rules:	The value of this attribute is overwritten with a new value as soon as the application evaluates the expression.
DTD:	`<!ATTLIST text:conditional-text text:current-value %boolean; "false">`
Note:	This attribute has no function other than to ease transformation or initially display the document.

3.7.18 Hidden Text Field

The hidden text field is closely related to the conditional text field. It displays fixed text, except when the condition is `true` when it does not display anything.

XML Code:	`<text:hidden-text>`
Rules:	
DTD:	`<!ELEMENT text:hidden-text (#PCDATA)>`

The attributes that you can associate with the `<text:hidden-text>` element are:

- Condition
- Text
- Is hidden

Condition

The `text:condition` attribute contains a boolean expression. If the expression evaluates to `true`, the text is hidden.

XML Code:	`text:condition`
Rules:	
DTD:	`<!ATTLIST text:hidden-text text:condition % formula; #REQUIRED>`

Text

The `text:string-value` attribute specifies the text to display if the condition is `false`.

XML Code:	`text:string-value`
Rules:	The value of this attribute is displayed if the condition evaluates to `false`.
DTD:	`<!ATTLIST text:hidden-text text:string-value % formula; #REQUIRED>`

Is Hidden

The `text:is-hidden` attribute specifies whether or not the field is currently visible. The purpose of this attribute is similar to that of the `text:current-value` attribute in the `text:condition` field. Recording

the result allows transformations to correctly represent the document without having to parse the condition expression or evaluate the condition when loading the document.

XML Code:	`text:is-hidden`
Rules:	The value of this attribute is overwritten with a new value as soon as the application evaluates the expression.
DTD:	`<!ATTLIST text:hidden-text text:is-hidden % boolean; "false">`
Note:	This attribute has no function other than to ease transformation or initially display the document.

3.7.19 Hidden Paragraph Fields

The hidden paragraph field has a similar function to the hidden text field. However, the hidden paragraph field does not have any content. It hides the paragraph in which it is contained. This allows you to hide or display a paragraph of formatted text, depending on whether a condition is `true` or `false`.

Hidden paragraph fields are often used together with form letters. For example, if a condition depends on a database field, a hidden paragraph field can be used to selectively include paragraphs in the form letter depending on the database content. Multiple paragraph fields can be contained one paragraph. The paragraph is displayed if the condition associated with at least one hidden paragraph field is `false`. Alternatively, you can combine the conditions associated with several hidden paragraph fields into a single condition for a single field using logical operations on the conditions.

XML Code:	`<text:hidden-paragraph>`
Rules:	
DTD:	`<!ELEMENT text:hidden-paragraph EMPTY>`
Note:	Unlike most fields, this field does not display text, but it affects the entire paragraph in which it is contained.

The attributes that you can associate with the `<text:hidden-paragraph>` element are:

- Condition
- Is hidden

Condition

The `text:condition` attribute contains a boolean expression.

XML Code:	`text:condition`
Rules:	If the condition is `true`, the paragraph is hidden. If the condition is `false`, the paragraph is displayed.
DTD:	`<!ATTLIST text:hidden-paragraph text:condition % formula; #REQUIRED>`

Is Hidden

The `text:is-hidden` attribute records whether the paragraph is currently visible or not. It has the same purpose as the corresponding attribute of the hidden text field, namely to allow correct display of the paragraph

without having to evaluate the condition first.

XML Code:	`text:is-hidden`
Rules:	The value of this attribute is overwritten with a new value as soon as the application evaluates the expression.
DTD:	`<!ATTLIST text:hidden-paragraph text:is-hidden % boolean; "false">`
Note:	This attribute has no function other than to ease transformation or initially display the document.

3.7.20 Chapter Fields

Chapter fields display one of the following:

- The name of the current chapter
- The number of the current chapter
- Both the name and number of the current chapter

If the chapter field is placed inside a header or footer, it displays the current chapter name or number on every page.

XML Code:	`<text:chapter>`
Rules:	
DTD:	`<!ELEMENT text:chapter (#PCDATA)>`

The attributes that you can associate with the `<text:chapter>` element are:

- Display
- Outline level

Display

The `text:display` attribute specifies the information that the chapter field should display.

XML Code:	`text:display`
Rules:	
DTD:	`<!ATTLIST text:chapter text:display ( name | number | number-and-name | plain-number-and-name | plain-number ) "number-and-name">`
Implementation Limitation:	In the current version of the OpenOffice.org Writer user interface, `plain-number-and-name` chapter fields are not supported.

Example: If the current chapter number is 2.4, the chapter title is Working with Tables, the prefix is [, and suffix is], the possible display options and results are as follows:

Value of `text:display` attribute	**Field content displayed**
`number`	[2.4]
`name`	Working with Tables

Value of `text:display` attribute	Field content displayed
`number-and-name`	[2.4] Working with Tables
`plain-number`	2.4
`plain-number-and-name`	2.4 Working with Tables

Outline Level

This attribute allows you to specify the outline level to use. The chapter field displays the chapter number or title up to the specified outline level.

XML Code:	`text:outline-level`
Rules:	
DTD:	`<!ATTLIST text:chapter text:outline-level % integer; "1">`
Note:	OpenOffice.org Writer currently supports up to ten outline levels.

3.7.21 File Name Fields

File name fields display the name of the file that is currently being edited.

XML Code:	`<text:file-name>`
Rules:	
DTD:	`<!ELEMENT text:file-name (#PCDATA)>`

The attributes that you can associate with the `<text:file-name>` element are:

- Display
- Fixed

Display

The `text:display` attribute specifies how much of the file name to display. You can choose whether to display:

- The full file name including the path and the extension
- The file path only
- The file name only
- The file name and the extension

XML Code:	`text:display`
Rules:	
DTD:	`<!ATTLIST text:file-name text:display ( full | path | name | name-and-extension ) "full">`

Fixed File Name Fields

If a file name field is fixed, its value does not change when the file is edited.

XML Code:	`text:fixed`
Rules:	
DTD:	`<!ATTLIST text:file-name text:fixed %boolean; "false">`

3.7.22 Document Template Name Fields

The document template name field displays information about the document template in use, such as the template title or the file name.

XML Code:	`<text:template-name>`
Rules:	
DTD:	`<!ELEMENT text:template-name (#PCDATA)>`

The attribute that you can associate with the `<text:template-name>` element is:

- Display

Display

This attribute specifies which information about the document template to display. You can choose to display:

- The full file name including the path and the extension
- The file path only
- The file name only
- The file name and the extension
- The title
- The area of the document template

The latter two values are used in the OpenOffice.org Writer user interface document template dialog. The display values are a superset of the display values available for the `<text:file-name>` element.

XML Code:	`text:display`
Rules:	
DTD:	`<!ATTLIST text:template-name text:display ( full | path | name | name-and-extension | area | title ) "full">`

3.7.23 Page Variable Fields

Page variables allow you to define an alternative page numbering scheme. There is only one page variable, and it is set by any set page variable field in the document. The value of the page variable is increased on each page, in

the same way as regular page numbers.

Setting Page Variable Fields

To set a page variable field, you use the `<text:set-page-variable>` element.

XML Code:	`<text:set-page-variable>`
Rules:	
DTD:	`<!ELEMENT text:set-page-variable EMPTY>`

Turning Page Variables On or Off

At the beginning of a document, the page variable is inactive. You can use the `text:active` attribute to disable a page variable after it was used in the document.

XML Code:	`text:active`
Rules:	
DTD:	`<!ATTLIST text:set-page-variable text:active % boolean; "true">`

Page Variable Adjustment

The `text:page-adjust` attribute determines the page adjustment. The value of the active page variable is the current page number plus the closest page adjustment value that was previously set.

XML Code:	`text:page-adjust`
Rules:	
DTD:	`<!ATTLIST text:set-page-variable text:page-adjust %integer; "0">`

Displaying Page Variable Fields

The `<text:get-page-variable>` element displays the value of the page variable. The field can be formatted in the same way as regular page number fields.

XML Code:	`<text:get-page-variable>`
Rules:	
DTD:	`<!ELEMENT text:get-page-variable (#PCDATA)>`

The attributes that you can associate with the `<text:get-page-variable>` element are:

- `text:num-format` and `text:num-letter-sync`

  These attributes determine how the number should be formatted. See Section 3.7.12 for more information on how to use these attributes.

DTD:	`<!ATTLIST text:get-page-variable %num-format; #IMPLIED>`

3.7.24 Macro Fields

The macro field contains the name of a macro that is executed when the field is activated. The field also contains a description that is displayed as the field content.

XML Code:	`<text:execute-macro>`
Rules:	
DTD:	`<!ELEMENT text:execute-macro (#PCDATA)>`

The attribute that you can associate with the `<text:execute-macro>` element is:

- Macro name

Macro Name

The `text:name` attribute specifies the macro to invoke when the field is activated.

XML Code:	`text:name`
Rules:	
DTD:	`<!ATTLIST text:execute-macro text:name %string; #REQUIRED>`

3.7.25 DDE Connections

A Dynamic Data Exchange (DDE) connection consists of the parameters for the DDE target application, a file name, and a command string. A DDE connection also takes a parameter that specifies whether it will be updated automatically or only on the user's request. Every DDE connection must be named.

Container for DDE Connection Declarations

The DDE connection declarations are contained in one declarations element.

XML Code:	`<text:dde-connection-decls>`
Rules:	
DTD:	`<!ELEMENT text:dde-connections-decls (text:dde-connection-decl)*>`

Declaring DDE Connections

Every DDE connection is declared using a declaration element. Multiple DDE fields can refer to one DDE connection by using the same name. The declaration element has no content.

XML Code:	`<text:dde-connection-decl>`
Rules:	
DTD:	`<!ELEMENT text:dde-connections-decl  EMPTY>`

The attributes that you can associate with the `<text:dde-connection-decl>` element are:

- Connection name
- DDE target application
- DDE target file name
- DDE command
- Automatic update flag

Connection Name

The `text:name` attribute specifies the name by which the connection will be referred.

XML Code:	`text:name`
Rules:	
DTD:	`<!ATTLIST text:dde-connection-decl text:name %string; #REQUIRED>`

Target Application

The `text:dde-application` attribute specifies the name of the target application to use for the DDE connection.

XML Code:	`text:dde-application`
Rules:	
DTD:	`<!ATTLIST text:dde-connection-decl text:dde-application %string; #REQUIRED>`
Note:	The target name for OpenOffice.org is `soffice`. Therefore, internal DDE links have the attribute `text:dde-application="soffice"`.

Target Topic

The `text:dde-topic` attribute specifies the name of the topic to use for the DDE connection.

XML Code:	`text:dde-topic`
Rules:	
DTD:	`<!ATTLIST text:dde-connection-decl text:dde-topic %string; #REQUIRED>`
Note:	If the target application is OpenOffice.org, it interprets the DDE topic as the name of the file.

Target Item

The `text:dde-item` attribute specifies which information the target application should deliver.

XML Code:	`text:dde-item`
Rules:	

DTD:	`<!ATTLIST text:dde-connection-decl text:dde-item %string; #REQUIRED>`
Note:	If the target application for the DDE connection is OpenOffice.org Writer, the item represents the name of a bookmark. OpenOffice.org delivers the current text content to the requesting application.

Automatic Update

OpenOffice.org Writer can automatically update DDE links. If preferred, you can use the `text:automatic-update` attribute to specify that the DDE connection links should only be updated at the request of the user.

XML Code:	`text:automatic-update`
Rules:	If the value of this attribute is `true`, the DDE links are automatically updated. If this value of this attribute is `false`, the DDE links are updated on user request only.
DTD:	`<!ATTLIST text:dde-connection-decl text:automatic-update %boolean; "false">`

3.7.26 DDE Connection Fields

A DDE field allows you to display information from a DDE connection. The only parameter required for the DDE field is the name of the DDE connection that supplies the data to this field. This DDE connection element specifies the actual DDE field that appears in the text body.

XML Code:	`<text:dde-connection>`
Rules:	
DTD:	`<!ELEMENT text:dde-connection (#PCDATA)>`

The attribute that you can associate with the `<text:dde-connection>` element is:

- DDE connection name

DDE Connection Name

The `text:name` attribute specifies the name of the DDE connection to which the field refers.

XML Code:	`text:name`
Rules:	
DTD:	`<!ATTLIST text:dde-connection text:name %string; #REQUIRED>`

3.7.27 Reference Fields

OpenOffice.org Writer uses five types of reference field and each type is represented by its own element. The reference field types are based on the type of element they refer to; footnotes, endnotes, bookmarks, references, and sequences. Every reference contains a reference format which determines what information about the referenced target is displayed. For example, references can display:

- The page number of the referenced target
- The chapter number of the referenced target
- Wording indicating whether the referenced target is above or below the reference field

In addition, each reference field must identify its target which is usually done using a name attribute. Bookmarks and references are identified by the name of the respective bookmark or reference. Footnotes, endnotes, and sequences are identified by a name that is usually generated automatically when a document is exported.

XML Code:	`<text:reference-ref>`
	`<text:sequence-ref>`
	`<text:bookmark-ref>`
	`<text:footnote-ref>`
	`<text:endnote-ref>`

Rules:	

DTD:	`<!ELEMENT text:reference-ref (#PCDATA)>`
	`<!ELEMENT text:sequence-ref (#PCDATA)>`
	`<!ELEMENT text:bookmark-ref (#PCDATA)>`
	`<!ELEMENT text:footnote-ref (#PCDATA)>`
	`<!ELEMENT text:endnote-ref (#PCDATA)>`

The attributes that you can associate with the reference field elements are:

- Reference name
- Reference format

Reference Name

The `text:ref-name` attribute identifies the referenced element. Since bookmarks and references have a name, this name is used by the respective reference fields. Footnotes, endnotes, and sequences are assigned names by the application used to create the OpenOffice.org XML file format when the document is exported.

XML Code:	`text:ref-name`
Rules:	
DTD:	`<!ATTLIST text:reference-ref text:ref-name %` `string; #REQUIRED>`

Reference Format

The `text:reference-format` attribute determines what information about the reference is displayed.

XML Code:	`text:reference-format`

Rules: All types of reference fields support the following values for this attribute formats:

- `page`, which displays the number of the page on which the referenced item appears.
- `chapter`, which displays the number of the chapter in which the referenced item appears.
- `direction`, which displays whether the referenced item is above or below the reference field.
- `text`, which displays the text of the referenced item.

If the reference format is not specified, the page format is used as the default.

References to sequence fields support the following three additional values:

- `category-and-value`, which displays the name and value of the sequence.
- `caption`, which displays the caption in which the sequence is used.
- `value`, which displays the value of the sequence.

DTD:

```
<!ATTLIST text:reference-ref text:reference-format
(page|chapter|text|direction) #IMPLIED>
<!ATTLIST text:footnote-ref text:reference-format
(page|chapter|text|direction) #IMPLIED>
<!ATTLIST text:endnote-ref text:reference-format
(page|chapter|text|direction) #IMPLIED>
<!ATTLIST text:bookmark-ref text:reference-format
(page|chapter|text|direction) #IMPLIED>
<!ATTLIST text:sequence-ref text:reference-format
(page|chapter|text|direction|category-and-value|
caption|value) #IMPLIED>
```

Example: Different reference formats and displays

The following table shows all possible reference formats and the resulting reference display that can be used to refer to the table itself. The left column lists the value of the `text:reference-format` attribute and the right column

Table 2: Examples of reference formats

Reference format	Reference display
`page`	182
`chapter`	3.7.27
`text`	Table 2: Examples of reference formats
`direction`	above
`category-and-value`	Table 1
`caption`	Examples of reference formats
`value`	1

3.7.28 Variable Fields

OpenOffice.org Writer documents can contain variables, which are processed or displayed using variable fields. A variable is a name/value pair. The variable name is used throughout the document to identify a particular variable, and therefore variable names cannot be reused for different types of variables. Most variable fields support different value types, such as numbers, dates, strings, and so on. In the OpenOffice.org XML file format, a variable must be declared at the beginning of a document.

There are three types of variables in OpenOffice.org Writer:

- **Simple variables**

  Simple variables, usually called variables, can take different values at different positions throughout a document. You set simple variables using either setter or input fields. Setter fields contain an expression, which is used to compute the new value of the variable. Input fields prompt the user for the new value. You can use simple variables to display different text in recurring elements, such as headers or footers.

- **User variables**

  User variables have the same value throughout a document. If a user variable is set anywhere within the document, all fields in the document that display the user variable have the same value. In the OpenOffice.org user interface, you can set a user variable at any occurrence of a user field, or using user variable input fields. In the OpenOffice.org XML file format, you can only set the value of the user variable after the variable is declared.

- **Sequence variables**

  Sequence variables are used to number certain items in an OpenOffice.org Writer document, for example, images or tables.

Expression and text input fields are also variable fields, but they are not associated with any particular variables. Since their functionality is closely related to that of the variable fields, they are also described in this section of the manual.

XML Code:	`%variable-fields;`
Rules:	
DTD:	`<!ENTITY % variable-fields "text:variable-set | text:variable-get | text:variable-input | text:user-field-get | text:user-field-input | text:sequence | text:expression | text:text-input" >`

You must declare variables before you can use them. The variable declarations are collected in container elements for the particular variable type. The OpenOffice.org XML code for declaring variables is described in the following table.

XML Code:	`%variable-declarations;`
Rules:	
DTD:	`<!ELEMENT text:variable-decls "text:variable-decl*"> <!ELEMENT text:user-field-decls "text:user-field-decl*"> <!ELEMENT text:sequence-decls "text:sequence-decl*"> <!ENTITY % variable-declarations "text:variable-decl?, text:user-field-decl?, text:sequence-decl?>`

3.7.29 Declaring Simple Variables

You declare simple variables using `<text:variable-decl>` elements. The declaration specifies the name

and the value type of the variable.

XML Code:	`<text:variable-decl>`
Rules:	This element does not have any content.
DTD:	`<!ELEMENT text:variable-decl EMPTY>`

To specify the name and value type of the simple variable, you attach the following attributes to the `<text:variable-decl>` element:

- `text:name`

  The name of the variable must be unique. The name cannot already be used for any other type of variable. See Section 3.7.43 for information on using this attribute.

  DTD: `<!ATTLIST text:variable-decl text:name %variable-name; #REQUIRED>`

- `text:value-type`

  See Section 3.7.43 for information on using this attribute.

  DTD: `<!ATTLIST text:variable-decl %value-type-attlist;>`

3.7.30 Setting Simple Variables

You can set simple variables using variable setter elements.

XML Code:	`<text:variable-set>`
Rules:	This element contains the presentation of the value of the variable, which can be empty if the `text:display` attribute is set to `none`.
DTD:	`<!ELEMENT text:variable-set (#PCDATA)>`

The attributes that you can attach to the `<text:variable-set>` element are:

- `text:name`

  This attribute specifies the name of the variable to set. It must match the name of a variable that has already been declared. See Section 3.7.43 for information on using this attribute.

  DTD: `<!ATTLIST text:variable-set text:name %variable-name;>`

- `text:formula`

  This attribute contains the formula to compute the value of the variable field. If the formula equals the content of the field element, you can omit this attribute. See Section 3.7.43 for information on using this attribute.

  DTD: `<!ATTLIST text:variable-set text:formula %formula;>`

- `text:value-type` and the appropriate value attribute

  See Section 3.7.43 for information on using these attributes.

  Note: A simple variable should not contain different value types at different places in a document. However, the current OpenOffice.org software implementation allows the use of different value types for different instances of the same variable. In the case of the numeric value types `float`, `percentage`, and `currency`, the value is automatically converted to the different value type. For value types that are stored

internally as numbers, such as `date`, `time`, and `boolean` types, the values are reinterpreted as numbers of the respective types. If a variable is used for both string and non-string types, the behavior is undefined, therefore this practice is not recommended.

DTD:	`<!ATTLIST text:variable-set %value-attlist;>`

- `text:display`

  You can use this attribute to specify whether or not to display the value of the `<text:variable-set>` element. If the `text:display` attribute is set to `value`, the value of the variable is displayed. If the attribute is set to `none`, the value is not displayed. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-set text:display ( value | none ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-set style:data-style-name %style-name; #IMPLIED>`

3.7.31 Displaying Simple Variables

The `<text:variable-get>` element reads and displays the value of a simple variable.

XML Code:	`<text:variable-get>`
Rules:	The value of this element is the value of the last preceding `<text:variable-set>` element with an identical `text:name` attribute. The element determines how the value of the variable is presented, in accordance with the chosen formatting style.
DTD:	`<!ELEMENT text:variable-get (#PCDATA)>`

The attributes that you can attach to the `<text:variable-get>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. The name must match the name of a preceding `<text:variable-decl>` element. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-get text:name %variable-name; #REQUIRED>`

- `text:display`

  You can use this attribute to specify whether to display the formula for a simple variable or the computed value of the variable. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-get text:display ( value | formula ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-get style:data-style-name %style-name; #IMPLIED>`

3.7.32 Simple Variable Input Fields

As an alternative to setting simple variables using formulas in variable setter fields, the user can be prompted for variable values. To do this, you use the `<text:variable-input>` element.

XML Code:	`<text:variable-input>`
Rules:	This element contains the presentation of the variable's value according to the chosen formatting style.
Note:	The presentation can be empty if the `text:display` attribute is set to `none`.
DTD:	`<!ELEMENT text:variable-input (#PCDATA)>`

The attributes that you can attach to the `<text:variable-input>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. It must match the name of a variable that was already declared. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input text:name %variable-name; #REQUIRED>`

- `text:description`

  This optional attribute contains a brief message that is presented to users when they are prompted for input. The message should give users enough information about the variable or the use of the value within the document to enable them to choose an appropriate value. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input text:description %string: #IMPLIED>`

- `text:value-type` and the appropriate value attribute

  See Section 3.7.43 for information on using these attributes.

DTD:	`<!ATTLIST text:variable-input %value-attlist;>`

- `text:display`

  You can use this attribute to specify whether to display or hide the value of the variable through the variable input field. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input text:display ( value | none ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:variable-input style:data-style-name %style-name; #IMPLIED>`

3.7.33 Declaring User Variables

User variables contain values that are displayed using appropriate fields. Unlike simple variables, user variables have the same value throughout a document because the value of the user variable is specified in the variable declaration.

XML Code:	`<text:user-field-decl>`
Rules:	This element does not have any content.
DTD:	`<!ELEMENT text:user-field-decl EMPTY>`

The attributes that you can associate with the `<text:user-field-decl>` element are:

- `text:name`

  This attribute specifies the name of the variable that you want to declare. The name must be unique. It cannot already be used for any other type of variable including simple and sequence variables. See Section 3.7.43 for information on using this attribute.

  DTD: `<!ATTLIST text:user-field-decl text:name %variable-name; #REQUIRED>`

- `text:formula`

  This attribute contains the formula to compute the value of the user variable field. If the formula is the same as the content of the field element, you can omit this attribute. See Section 3.7.43 for information on using this attribute.

  DTD: `<!ATTLIST text:user-field-decl text:formula %formula; #IMPLIED>`

- `text:value-type` and the appropriate value attribute

  See Section 3.7.43 for information on using these attributes.

  DTD: `<!ATTLIST text:user-field-decl %value-attlist;>`

3.7.34 Displaying User Variables

You can display the content of user variables using `<text:user-field-get>` elements.

XML Code:	`<text:user-field-get>`
Rules:	
DTD:	`<!ELEMENT text:user-field-get (#PCDATA)>`

The attributes that you can attach to the `<text:user-field-get>` element are:

- `text:name`

  This attribute specifies the name of the variable to display. The name must match the name of a preceding `<text:user-field-decl>` element. See Section 3.7.43 for information on using this attribute.

  DTD: `<!ATTLIST text:user-field-get text:name %variable-name; #REQUIRED>`

- `text:display`

You can use this attribute to specify whether to:

- Display the formula used to compute the value of the user variable.
- Display the value of the user variable.
- Hide the user variable fields.

See Section 3.7.43 for information on using this attribute.

Note: Since the OpenOffice.org Writer user interface allows users to edit a user field variable by clicking on any user field, a hidden <text:user-field-get> element can be used as an anchor to allow easy access to a particular user field variable.

DTD:	<!ATTLIST text:user-field-get text:display (value | formula | none) "value">

- style:data-style-name

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.7.43 for information on using this attribute.

DTD:	<!ATTLIST text:user-field-get style:data-style-name %style-name;>

3.7.35 User Variable Input Fields

An alternative method of setting user variables is to use input fields, similar to the input fields for simple variables. You can set a user variable in this way using the <text:user-field-input> element. Since the value of a user field variable is stored in the <text:user-field-decl> element, the <text:user-field-input> element does not contain the value and value type attributes from the <text:variable-input> field.

XML Code:	<text:user-field-input>
Rules:	This element determines how the value of the user variable is displayed, in accordance with the chosen formatting style.
Note:	The presentation can be empty if the text:display attribute is set to none.
DTD:	<!ELEMENT text:user-field-input (#PCDATA)>

The attributes that you can attach to the <text:user-field-input> element are:

- text:name

  This attribute specifies the name of the variable to set. It must match the name of a variable that has already been declared. See Section 3.7.43 for information on using this attribute.

DTD:	<!ATTLIST text:user-field-input text:name %variable-name; #REQUIRED>

- text:description

  This optional attribute contains a brief message that is presented to users when they are prompted for input. The message should give users enough information about the variable or the use of the value within the document, to enable them to choose an appropriate value. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-input text:description %string; #IMPLIED>`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:user-field-input style:data-style-name % style-name; #IMPLIED>`

3.7.36 Declaring Sequence Variables

Sequence variables are used to number items within an OpenOffice.org Writer document. Sequence variables are most commonly used for sequential numbering. However, you can include expression formulas in sequence fields to support more advanced sequences. See Section 3.7.37 for more information on using sequence fields and their uses.

You declare sequence variables using the `<text:sequence-decl>` element.

XML Code:	`<text:sequence-decl>`
Rules:	This element does not have any content.
DTD:	`<!ELEMENT text:sequence-decl EMPTY>`

To facilitate chapter-specific numbering, you can attach attributes for the chapter level and a separation character to a sequence variable. The attributes that you can attach to the `<text:sequence-decl>` element are:

- `text:name`

  This attribute specifies the name of the variable that you want to declare. The name must be unique. It cannot already be used for any other type of variable including simple and user variables. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:sequence-decl text:name %variable-name; #REQUIRED>`

- `text:display-outline-level`

  See the section *Outline Level* for information about this attribute.

- `text:separation-character`

  See the section *Separation Character* for information about this attribute.

Outline Level

You can number sequences by chapter. To use this feature, use the `text:display-outline-level` attribute to specify an outline level that determines which chapters to reference for the chapter-specific numbering. All chapters that are at or below the specified outline level reset the value of the sequence to zero, the default value. Also, the chapter number of the last chapter at or below the specified outline level is prepended to the sequence number. Choosing an outline level of zero results in a straight sequence of all sequence elements for that sequence variable.

XML Code:	`text:display-outline-level`

Rules:	The value of this attribute must be an integer greater than or equal to zero. OpenOffice.org currently supports ten outline levels.
DTD:	`<!ATTLIST text:sequence-decl text:display-outline-level % integer; "0">`

Separation Character

If you number sequences by chapter, use this attribute to choose a character to separate the chapter number from the sequence number.

XML Code:	`text:separation-character`
Rules:	If the value of the `text:display-outline-level` attribute is a non-zero value, you must specify a separation character. Otherwise, if the value of `text:display-outline-level` is zero, you must omit this attribute.
DTD:	`<!ATTLIST text:sequence-decl text:separation-character % character; ".">`

Example: Sequence variable

The sequence variable 3.7.36#5 with a value of `5` is declared using:

Attribute	Value
`text:display-outline-level`	`3`
`text:separation-character`	`#`

3.7.37 Using Sequence Fields

Once a sequence variable is declared, you can use it in sequence fields throughout the document. Most sequence fields simply increment and display the sequence variable. However, sequence fields can also assume a new start value at any given position in a document. This start value is computed using a formula which is contained in the sequence field. If a sequence field without a start value is added to the OpenOffice.org user interface, the OpenOffice.org software automatically inserts an expression of the type `variable+1`.

Sequence fields are most commonly used for simple counting sequences. However, the ability to provide arbitrary expressions supports more complex sequences. To form a sequence of even numbers, all sequence elements for that particular variable need to contain a formula incrementing the value by two, for example, `variable+2`. A sequence with a starting value of `1` and all subsequent elements using the formula `variable*2` yields all powers of two. Since different sequence elements for the same sequence variable may contain different formulas, complex sequences may be constructed.

XML Code:	`<text:sequence>`
Rules:	
DTD:	`<!ELEMENT text:sequence (#PCDATA)>`

The attributes that you can attach to the `<text:sequence>` element are:

- `text:name`

  This attribute specifies the name of the variable that the field is to display. It must match the name of a sequence variable that was already declared. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:sequence text:name %variable-name; #REQUIRED>`

- `text:formula`

  This optional attribute contains a formula to compute the value of the sequence field. If this attribute is omitted, an expression containing the content of the element is used. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:sequence text:formula %formula; #IMPLIED>`

- `style:num-format` and `style:num-letter-sync`

  These attributes specify the numbering style to use. If a numbering style is not specified, the numbering style is inherited from the page style. See Section 3.7.43 for information on these attributes.

DTD:	`<!ATTLIST text:page-number %num-format;>`

- `text:ref-name`

See the following section *Reference Name* for more information about this attribute.

Reference Name

Sequence fields can be the target of references, as implemented using reference fields. See Section 3.7.27 for more information about reference fields. To enable a reference field to identify a particular sequence field, the sequence field must contain an additional attribute containing a name. No two sequence fields can have the same reference name.

XML Code:	`text:ref-name`
Rules:	If the sequence field is not the target of a reference, this attribute can be omitted.
DTD:	`<!ATTLIST text:ref-name %string; #IMPLIED>`

3.7.38 Expression Fields

Expression fields contain expressions that are evaluated and the resulting value is displayed. The value of the expression is formatted according to the chosen formatting style.

XML Code:	`<text:expression>`
Rules:	
DTD:	`<!ELEMENT text:expression (#PCDATA)>`

The attributes that you can attach to the `<text:expression>` element are:

- `text:formula`

  This attribute contains the actual expression used to compute the value of the expression field. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:expression text:formula %formula; #IMPLIED>`

- `text:value-type` and the appropriate value attribute

  See Section 3.7.43 for information on using these attributes.

DTD:	`<!ATTLIST text:expression %value-type;>`

- `text:display`

  Use this attribute to specify one of the following:

  - To display the value of the field.
  - To display the formula used to compute the value.

  See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:expression text:display ( value | formula ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:expression style:data-style-name %style-name; #IMPLIED>`

3.7.39 Text Input Fields

A text input field is a variable field. From the point of view of the OpenOffice.org user interface, a text input field is similar to the `<text:variable-input>` and `<text:user-field-input>` fields. However, the text input field does not change the value of any variables.

XML Code:	`<text:text-input>`
Rules:	
DTD:	`<!ELEMENT text:text-input (#PCDATA)>`

The attribute that you can attach to the `<text:text-input>` element is:

- `text:description`

  This attribute contains a brief message that is presented to users when they are prompted for input. The message should give users enough information about the purpose of the field and how it is used within the document, to enable them to choose an appropriate value. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:text-input text:description %string; #IMPLIED>`

3.7.40 Script Fields

A script field stores scripts or sections of scripts.You can use the field to store and edit scripts that are attached to the document. The primary purpose of this field is to provide an equivalent to the `<script>` element in HTML, so that the content of a `<script>` element in HTML can be imported, edited, and exported using the OpenOffice.org software.

The source code for the script can be stored in one of the following ways:

- The `<text:script>` element contains the source code.

- The source code is stored in an external file. Use the `text:href` attribute to specify the location of the source file.

XML Code:	`<text:script>`
Rules:	The element should have either a `text:href` attribute or content, but not both.
DTD:	`<!ELEMENT text:script (#PCDATA)>`

Script URL

The `text:href` attribute specifies the location of the file that contains the script source code.

XML Code:	`text:href`
Rules:	The script field should have either an URL attribute or content, but not both.
DTD:	`<!ATTLIST text:script text:href CDATA>`

Script Language

The `script:language` attribute specifies the language in which the script source code is written, for example, JavaScript.

XML Code:	`script:language`
Rules:	
DTD:	`<!ATTLIST text:script script:language CDATA>`

3.7.41 Measure Fields

Information to be supplied.

3.7.42 Table Formula Field

The table formula field is a legacy from previous version of StarWriter. It should not be used in new documents. It stores a formula to eb used in tables, a function that is better performed by the table:formula attribute of the table cell.

XML Code:	`<text:table-formula>`
Rules:	This element should not be used in new documents
DTD:	`<!ELEMENT text:table-formula (#PCDATA)>`

The table formula field can take the following attributes:

- `text:formula`

  This attribute contains the actual expression used to compute the value of the table formula field. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:table-formula text:formula %formula; #IMPLIED>`

- `text:display`

  Use this attribute to specify one of the following:

  - To display the value of the field.
  - To display the formula used to compute the value.

  See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:table-formula text:display ( value | formula ) "value">`

- `style:data-style-name`

  This attribute specifies the data style to use to format a numeric, boolean, or date/time variable. If a data style is not specified, a standard data style is used. See Section 3.7.43 for information on using this attribute.

DTD:	`<!ATTLIST text:table-formula style:data-style-name %style-name; #IMPLIED>`

3.7.43 Common Field Attributes

You can use the attributes described in this section with several field elements.

Variable Value Types and Values

Variables and most variable fields have a current value. Every variable has a value type that must be specified when the field supports multiple value types. The value type is specified using the `text:value-type` attribute.

XML Code:	`text:value-type`
Rules:	This attribute must specify one of the following value types for the variable: `float`, `percentage`, `currency`, `date`, `time`, `boolean`, or `string`.
DTD:	`<!ENTITY % value-type-attlist "text:value-type ( float | time | date | percentage | currency | boolean | string ) #REQUIRED">`
Note:	This entity should be used within any `<!ATTLIST>` definitions for `text:value-type` attributes.

Example:

```
<!ELEMENT some-element (#PCDATA)>
<!ATTLIST some-element %value-type-attlist>
```

Depending on the value type, the value itself is written to different value attributes. The supported value types, their respective value attributes, and how the values are encoded are described in the following table:

Value of `text:value-type`	**Value Attribute**	**Encoded as...**
`float`, `percentage`	`text:value`	Numeric value
`currency`	`text:value` and `text:currency`	Numeric value and currency symbol

Value of `text:value-type`	Value Attribute	Encoded as...
`date`	`text:date-value`	Described in ISO 8601, §5.2.1.1, extended format
`time`	`text:time-value`	Described in ISO 8601, §5.4.1 a), extended format
`boolean`	`text:boolean-value`	`true` or `false` strings
`string`	`text:string-value`	Strings

The OpenOffice.org Writer concept of field values and value types and their encoding in XML is modeled on the corresponding XML for table cell attributes. See Section 4.8.1 for more detailed information on these attributes.

The definition of the entity `%value-attlist;` is as follows:

DTD:
```
<!ENTITY % value-attlist
  "%value-type-attlist;
   text:value %float; #IMPLIED
   text:date-value %date; #IMPLIED
   text:time-value %time; #IMPLIED
   text:boolean-value %boolean; #IMPLIED
   text:string-value %string; #IMPLIED
   text:currency CDATA #IMPLIED" >
```

Example:
```
<!ELEMENT some-element (#PCDATA)>
<!ATTLIST some-element %value-attlist;>
```

This entity is useful for defining all value and value type related attributes for any element.

Fixed

The `text:fixed` attribute specifies whether or not the value of a field element is fixed. If the value of a field is fixed, the original value of the field is preserved. If the value of the field is not fixed, the value of the field is replaced by a new value when the document is edited.

This attribute can be used with:

- Date fields
- Time fields
- Page number fields
- All sender fields
- All author fields

XML Code:	`text:fixed`
Rules:	If the value of this attribute is set to `true`, the value of the field element to which this attribute is attached is preserved in all future edits of the document. If the value of this attribute is set to `false`, the value of the field element is not preserved and will be replaced with new values as appropriate.
Sample DTD:	`<!ATTLIST text:author-name text:fixed %boolean; "true">`

Variable Name

Use the `text:name` attribute to specify the name of a variable when you are declaring, setting, or displaying a variable. You can use this attribute with any of the following elements:

- `<text:variable-decl>`
- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-decl>`
- `<text:user-field-get>`
- `<text:user-field-input>`
- `<text:sequence-decl>`
- `<text:sequence>`

XML Code:	`text:name`
Rules:	When you are using this attribute to specify the name of a variable to display, a variable of the appropriate type with the same name must already have been declared.
Sample DTD:	`<!ATTLIST text:sequence text:name %variable-name; #REQUIRED>`

Description

The `text:description` attribute contains a brief message that is displayed when users are prompted for input. You can use this attribute with any of the following elements:

- `<text:placeholder>`
- `<text:variable-input>`
- `<text:user-field-input>`
- `<text:text-input>`

XML Code:	`text:description`
Rules:	The optional `text:description` attribute may contain a brief description.
Sample DTD:	`<!ATTLIST text:text-input text:description %string; #IMPLIED>`

Display

The `text:display` attribute supports up to three values as follows:

- `value`
  This value displays the value of the field. Some fields do not support this value. In these cases, the `text:display` attribute only takes the values `value` or `none`, and `value` or `formula`, respectively.
- `formula`
  This value allows you to display the formula rather than the value of the field. Some fields do not support this

value. In these cases, the `text:display` attribute only takes the values `value` or `none`, and `value` or `formula`, respectively.

- `none`
  Several variable fields support this value, which hides the field content. This allows you to set variables in one part of the document and display them in another part of the document.

You can use this attribute with any of the following elements:

- `<text:variable-set>`
- `<text:variable-get>`
- `<text:variable-input>`
- `<text:user-field-get>`
- `<text:expression>`

XML Code:	`text:display`
Rules:	If the value of this attribute is `value`, the value of the field is displayed. If the value is `formula`, the formula expression used to compute the value is displayed. Otherwise, the field is not be displayed.
Sample DTD:	`<!ATTLIST text:user-field-get text:display ( value | formula | none ) "value">`

Formula

The `text:formula` attribute contains the formula or expression used to compute the value of the field. You can use this attribute with any of the following elements:

- `<text:variable-set>`
- `<text:user-field-decl>`
- `<text:sequence>`
- `<text:expression>`

XML Code:	`text:formula`
Rules:	
Sample DTD:	`<!ATTLIST text:expression text:formula %formula; #REQUIRED>`

Formatting Style

The `style:data-style-name` attribute refers to the data style used to format the numeric value. For general information about styles, see Chapter 1. For more information about data styles, see Chapter 2.

You can use this attribute with any of the following elements:

- `<text:date>`
- `<text:time>`
- `<text:page-number>`

- <text:variable-set>
- <text:variable-get>
- <text:variable-input>
- <text:user-field-get>
- <text:user-field-input>
- <text:expression>

XML Code:	style:data-style-name
Rules:	For string variables you must omit this attribute. Otherwise, this attribute is required. The name must match the name of a data style.
Sample DTD:	<!ATTLIST text:expression style:data-style-name %style-name; #IMPLIED>

Number Formatting Style

You can format numbers that are used for number sequences such as page numbers or sequence fields according to the number styles described in Chapter 2. The number styles supported are as follows:

- Numeric: 1, 2, 3, ...
- Alphabetic: a, b, c, ... or A, B, C, ...
- Roman: i, ii, iii, iv, ... or I, II, III, IV,...

XML Code:	text:num-format
Rules:	The value of this attribute can be any of the XSL number format keys 1, i, I, a, or A.
Sample DTD:	<!ATTLIST some-element text:num-format CDATA #IMPLIED>

Alphabetic number styles need an additional attribute to determine how to display numbers that cannot be represented by a single letter. The OpenOffice.org software supports:

- Synchronized letter numbering, where letters are used multiple times, for example aa, bb, cc, and so on.
- Non-synchronized letter numbering, for example aa, ab, ac, and so on.

See Chapter 2 for more information.

XML Code:	text:num-letter-sync
Rules:	
Sample DTD:	<!ATTLIST some-element text:num-letter-sync %boolean; "false">

The following entity aids the definition of elements that use number formats:

XML Code:	%num-format;
Sample DTD:	<!ENTITY % num-format 'text:num-format CDATA #IMPLIED text:num-letter-sync %boolean; "false"'>
Example:	<!ELEMENT some-element %num-format;>

3.8 Frames in Text Documents

A frame anchor consists of the following two parts:

- Anchor type
  The anchor type specifies how a frame is bound to the text document.
- Anchor position
  The anchor position is the point at which a frame is bound to a text document. For example, if a frame is bound to a page, the anchor position is the page number.

3.8.1 Anchor Type

The anchor type attribute specifies how a frame is bound to the text document.

XML Code:	`text:anchor-type`
Rules:	This attribute has to be attached to every frame element, for example, every `<text:text-box>` that is contained in a text document. It can also be attached to graphic styles, in this case it specifies the default anchor type for every frame that is inserted into a document using the graphic style.
DTD:	`<!ATTLIST style:properties text:anchor-type (page|frame|paragraph|char|as-char) #IMPLIED>`

3.8.2 Anchor Position

The anchor position is the point at which a frame is bound to a text document. The anchor position depends on the anchor type as explained in the following table.

If the value of the `text:anchor-type` attribute is ...	**The anchor position is...**	**The frame element appears ...**	**Notes**
`page`	The page that has the same physical page number as the value of the `text:anchor-page-number` attribute that is attached to the frame element. If no `text:anchor-page-number` attribute is given, the anchor position is the page at which the character behind the frame element appears.	Either • At the start of the document body, outside any paragraph or frame, provided a `text:anchor-page-number` attribute is given. Or • Inside any paragraph element that is not contained in a header, footer, footnote, or text box, if a `text:anchor-page-number` attribute is not given.	The physical page number is the number assigned to the page if all pages in the document are counted starting with page 1.
`frame`	The parent frame that the current frame element is contained in.	In the element representing the frame to which the frame is bound. For example, if an image is bound to a text box, the frame element is located in the text box element.	Currently, frames can only be bound to text boxes.

If the value of the text:anchor-type attribute is ...	The anchor position is...	The frame element appears ...	Notes
paragraph	The paragraph that the current frame element is contained in.	At the start of the paragraph element.	
char	The character after the frame element.	Just before the character.	
as-char	There is no anchor position. The frame behaves like a character.	At the position where the character appears in the document.	

Horizontal and Vertical Alignment

The following tables display the possible values of the attributes style:horizontal-pos, style:horizontal-rel, style:vertical-pos, and style:vertical-rel, depending on the anchor type of the frame. The possible values of these alignment attributes are listed in the first column on the left, and an alignment attribute value/anchor type value match is indicated by an X.

Value of style:horizontal-pos	Value of text:anchor-type				
	page	frame	paragraph	char	as-char
any	X	X	X	X	

Value of style:horizontal-rel	Value of text:anchor-type				
	page	frame	paragraph	char	as-char
page	X		X	X	
page-content	X		X	X	
page-start-margin	X		X	X	
page-end-margin	X		X	X	
frame		X			
frame-content		X			
frame-start-margin		X			
frame-end-margin		X			
paragraph			X	X	
paragraph-content			X	X	
paragraph-start-margin			X	X	
paragraph-end-margin			X	X	
char				X	

Value of style:vertical-pos	Value of text:anchor-type				
	page	frame	paragraph	char	as-char
any	X	X	X	X	X

Value of `style:vertical-rel`	Value of `text:anchor-type`				
	page	**frame**	**paragraph**	**char**	**as-char**
page	X				
page-content	X				
frame		X			
frame-content		X			
paragraph			X	X	
paragraph-content			X	X	
char				X	X
line					X
baseline					X
text					X

Note: XSL and HTML support very few of the combinations of anchor type, vertical/horizontal alignment, and wrap mode that OpenOffice.org supports.

3.8.3 Anchor Page Number

The `text:anchor-page-number` attribute specifies the physical page number of an anchor if the frame is bound to a page.

XML Code:	`text:anchor-page-number`
Rules:	
DTD:	`<!ATTLIST style:properties text:anchor-page-number %number; #IMPLIED>`

3.9 Ruby

Ruby is additional text that is displayed above or below some base text. The purpose of ruby is to annotate the base text or provide information about its pronunciation.

XML Code:	`<text:ruby>`
Rules:	This element can be contained anywhere within a paragraph. It contains to sub elements, one for the base and one for the ruby text.
DTD:	`<!ELEMENT text:ruby (text:ruby-base,text:ruby-text)>`

The attributes that you can associate with the `<text:ruby>` element are:

- Ruby style
- Ruby text formatting properties

There are two elements that can be contained in the `<text:ruby>` element:

- Ruby base
- Ruby text

Ruby Style

A ruby style specifies how the ruby text is displayed relative to the base text. It is represented by a <style:style> element those family is ruby. The ruby style is assigned to the ruby element using a text:style-name attribute.

XML Code:	text:style-name
Rules:	
DTD:	<!ATTLIST text:ruby text:style-name %styleName; #IMPLIED>

Ruby Text Formatting Properties

All of the ruby text is displayed using the same formatting properties. The style:style-name attribute is used to specify these properties.

XML Code:	style:style-name
Rules:	This attribute references a text style that is used to display the ruby text.
DTD:	<!ATTLIST text:ruby-text %styleName;>

Ruby Position

This property specifies the position of the ruby text relative to the ruby base.

XML Code:	style:ruby-position
Rules:	The value of this property can be above or below.
DTD:	<!ATTLIST style:properties style:ruby-position (above|below) #IMPLIED>

Ruby Alignment

This property specifies the alignment of the ruby text relative to the ruby base.

XML Code:	style:ruby-align
Rules:	The value of this property can be left, center, right, distribute-letter, or distribute-space.
DTD:	<!ATTLIST style:properties style:ruby-align (left|center|right|distribute-letter|distribute-space) #IMPLIED>

3.9.1 Ruby Base

The <text:ruby-base> element contains the text that is to be annotated.

XML Code:	<text:ruby-base >
Rules:	This element can contain any content.
DTD:	<!ELEMENT text:ruby-base (%inline-text;|#PCDATA)>

3.9.2 Ruby Text

The <text:ruby-text> element contains the annotation.

XML Code:	<text:ruby-text>
Rules:	
DTD:	<!ELEMENT text:ruby-text (#PCDATA)>

3.10 Text Formatting Properties

You can apply text formatting properties to text portions, paragraphs, and paragraph styles.

3.10.1 Font Variant

Use this property to switch the option to display text as small capitalized letters on or off.

XML Code:	fo:font-variant (XSL property)
Rules:	
DTD:	<!ATTLIST style:properties fo:font-variant (normal|small-caps) #IMPLIED>
Implementation limitation:	At present, the fo:font-variant and fo:text-transform properties are mutually exclusive. If both properties are attached to an item set element simultaneously, the result is undefined except that the fo:text-transform value is none and the fo:font-variant value is normal.

3.10.2 Text Transformations

Use this property to describe text transformations to uppercase, lowercase, and capitalization.

XML Code:	fo:text-transform (XSL property)
Rules:	
DTD:	<!ATTLIST style:properties fo:text-transform (none|lowercase|uppercase|capitalize) #IMPLIED>
Implementation limitation:	At present, the fo:font-variant and fo:text-transform properties are mutually exclusive. If both properties are attached to an item set element simultaneously, the result is undefined except that the fo:text-transform value is none and the fo:font-variant value is normal.

3.10.3 Color

Use this property to specify the foreground color of text.

XML Code:	`fo:color` (XSL property)
Rules:	
DTD:	`<!ENTITY % color "CDATA">` `<!ATTLIST style:properties fo:color %color; #IMPLIED>`

3.10.4 Window Font Color

Use this property to specify whether or not the window foreground color should be as used as the foreground color for a light background color and white for a dark background color.

XML Code:	`style:use-window-font-color`
Rules:	
DTD:	`<!ATTLIST style:properties style:use-window-font-color % boolean; #IMPLIED>`

3.10.5 Text Outline

Use this property to specify whether to display an outline of text or the text itself.

XML Code:	`style:text-outline`
Rules:	This attribute can have a value of `true` or `false`.
DTD:	`<!ATTLIST style:properties style:text-outline %boolean; #IMPLIED>`
Note:	XSL does not have a corresponding property.

3.10.6 Crossing Out

Use this property to specify the style to use when crossing out text.

XML Code:	`style:text-crossing-out`
Rules:	The value of this attribute is the crossing out style for the text.
DTD:	`<!ATTLIST style:properties style:text-crossing-out (none|single-line|double-line|thick-line|slash|X) #IMPLIED>`
Note:	XSL does not support this property, but the values `none` and `single-line` correspond to the values `none` and `underline` for the XSL `fo:text-decoration` property.

3.10.7 Text Position

Use this formatting property to specify whether text is positioned above or below the baseline and to specify the relative font height that is used for this text.

XML Code:	`style:text-position`
Rules:	This attribute can have one or two values. The first value must be present and specifies the vertical text position as a percentage that relates to the current font height or it takes one of the values `sub` or `super`. Negative percentages or the `sub` value place the text below the baseline. Positive percentages or the `super` value place the text above the baseline. If `sub` or `super` is specified, the application can choose an appropriate text position. The second value is optional and specifies the font height as a percentage that relates to the current font-height. If this value is not specified, an appropriate font height is used. Although this value may change the font height that is displayed, it never changes the current font height that is used for additional calculations.
DTD:	`<!ATTLIST style:properties style:text-position CDATA #IMPLIED>`
Note:	The effect of using this property is the same as the effect achieved by using the XSL properties `fo:vertical-align` and `fo:font-size`. This representation is not appropriate because the `fo:font-size` property is used to change the font height without changing its position.

3.10.8 Font Name

Use these properties to assign a font to the text.

XML Code:	`style:font-name`
	`style:font-name-asian`
	`style:font-name-complex`

Rules:	The values of these attributes form the name of a font that is declared by a `<style:font-decl>` element within the `<office:font-decls>` element. The `style:font-name-asian` attribute is evaluated for UNICODE characters that are CJK characters. The `style:font-name-complex` attribute is evaluated for UNICODE characters that are complex text layout (CTL) characters. The `style:font-name` attribute is evaluated for any other UNICODE character.

DTD:	`<!ATTLIST style:properties fo:font-name %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-name-asian %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-name-complex %string; #IMPLIED>`

3.10.9 Font Family

Use these properties to specify the font family for the text.

XML Code:	`fo:font-family` (XSL property)
	`style:font-family-asian`
	`style:font-family-complex`

Rules:	You can use these properties instead of the `style:font-name` attributes to specify the properties of a font individually. However, it is advisable touse the `style:font-name` attributes instead. See Section 3.10.8 for information about when asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:font-family %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-family-asian %string; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-family-complex %string; #IMPLIED>`

3.10.10 Font Family Generic

Use these properties to specify a generic font family name.

XML Code:	`style:font-family-generic`
	`style:font-family-generic-asian`
	`style:font-family-generic-complex`

Rules:	These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % fontFamilyGeneric "(roman|swiss|modern|decorative|script|system)">`
	`<!ATTLIST style:properties style:font-family-generic %fontFamilyGeneric; #IMPLIED>`
	`<!ATTLIST style:properties style:font-family-generic-asian %fontFamilyGeneric; #IMPLIED>`
	`<!ATTLIST style:properties style:font-family-generic-complex %fontFamilyGeneric; #IMPLIED>`

3.10.11 Font Style

Use these properties to specify a font style name.

XML Code:	`style:font-style-name`
	`style:font-style-name-asian`
	`style:font-style-name-complex`

Rules:	These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties style:font-style-name %string; #IMPLIED>`
	
	`<!ATTLIST style:properties style:font-style-name-asian % string; #IMPLIED>`
	
	`<!ATTLIST style:properties style:font-style-name-complex % string; #IMPLIED>`

Note:	XSL does not support this property.

3.10.12 Font Pitch

Use these properties to specify whether a font has a fixed or variable width.

XML Code:	`style:font-pitch`
	`style:font-pitchgv`
	`style:font-pitch-complex`

Rules:	These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % fontPitch "(fixed|variable)">`
	`<!ATTLIST style:properties style:font-pitch %fontPitch; #IMPLIED>`
	`<!ATTLIST style:properties style:font-pitch-asian %fontPitch; #IMPLIED>`
	`<!ATTLIST style:properties style:font-pitch-complex % fontPitch; #IMPLIED>`

Note:	XSL does not support this property.

3.10.13 Font Character Set

Use these properties to specify the character set of a font.

XML Code:	`style:font-charset`
	`style:font-charset-asian`
	`style:font-charset-complex`

Rules:	The value of these attributes can be `x-symbol` or the character encoding in the notation described in the XML recommendation (Chapter 4.3.3, Character Encoding and Entities, http://www.w3.org/TR/REC-xml#charencoding). If the value is `x-symbol`, all characters that are displayed using this font must be contained in the UNICODE character range 0xf000 to 0xf0ff. These properties are ignored if there is no corresponding `fo:font-family` property attached to the same properties element. You can use these properties instead of the `style:font-name` attributes to specify the properties of a font. However, it is advisable to use the `style:font-name` attribute instead. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties style:font-charset CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:font-charset-asian CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:font-charset-complex CDATA #IMPLIED>`

3.10.14 Font Size

Use these properties to specify the size of font.

XML Code:	`fo:font-size` (XSL property)
	`fo:font-size-asian`
	`fo:font-size-complex`

Rules:	The value of these property is either an absolute length or a percentage. In contrast to XSL, percentage values can be used within styles only and relate to the font height of the parent style rather than to the font height of the attributes neighborhood. Absolute font heights such as `medium`, `large`, `x-large`, and so on, and relative font heights such as `smaller`, and `larger` are not supported. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % length_or_percentage "CDATA">`
	`<!ATTLIST style:properties fo:font-size %length_or_percentage; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-asian % length_or_percentage; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-complex % length_or_percentage; #IMPLIED>`

3.10.15 Relative Font Size

Use these properties to specify a relative font size change.

XML Code:	`style:font-size-rel`
	`style:font-size-rel-asian`
	`style:font-size-rel-complex`

Rules:	These properties specify a relative font size change as a length such as `+1pt`, `-3pt`. It cannot be used within automatic styles. The size changes relates to the font size setting that applies to the parent style of the style. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:font-size-rel %length; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-rel-asian %length; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-size-rel-complex %length; #IMPLIED>`

3.10.16 Letter Spacing

Use this property to specify the amount of space between letters.

XML Code:	`fo:letter-spacing` (XSL property)
Rules:	The value of this property can be `normal` or it can specify a length.
DTD:	`<!ATTLIST style:properties fo:letter-spacing CDATA #IMPLIED>`

3.10.17 Language

Use this property to specify the language of the text.

XML Code:	`fo:language` (XSL property)
	`fo:language-asian`
	`fo:language-complex`

Rules:	The value of these properties can be any of the ISO 639 language codes (see http://www.oasis-open.org/cover/iso639a.html). At present, these properties are ignored if they are not specified together with the corresponding `fo:country` property. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:language CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:language-asian CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:language-complex CDATA #IMPLIED>`

3.10.18 Country

Use this property with the `fo:language` properties to specify the country where language of the text is used.

XML Code:	`fo:country` (XSL property)
	`fo:country-asian`
	`fo:country-complex`

Rules:	The value of these properties can be any of the ISO 3166 country codes (see http://www.sil.org/acpub/catalog/country.html). At present, these properties are ignored if they are not specified together with the corresponding `fo:language` property. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:country CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:country-asian CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:country-complex CDATA #IMPLIED>`

3.10.19 Font Style

Use these properties to specify whether to use normal or italic font face.

XML Code:	`fo:font-style` (XSL property)
	`fo:font-style-asian`
	`fo:font-style-complex`

Rules:	See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ENTITY % fontStyle "(normal|italic|oblique)">`
	`<!ATTLIST style:properties fo:font-style %fontStyle; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-style-asian %fontStyle; #IMPLIED>`
	`<!ATTLIST style:properties fo:font-style-complex %fontStyle; #IMPLIED>`

3.10.20 Font Relief

Use this property to specify whether the font should be embossed, engraved, or neither.

XML Code:	`style:font-relief`
Rules:	
DTD:	`<!ENTITY % fontRelief "(none|embossed|engraved)">` `<!ATTLIST style:properties style:font-relief %fontRelief; #IMPLIED>`

3.10.21 Text Shadow

Use this property to specify the text shadow style to use.

XML Code:	`fo:text-shadow` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:text-shadow CDATA #IMPLIED>`
Implementation limitation:	At present, OpenOffice.org only supports a default text shadow style. Therefore, any value other than `none` switches on this default shadow style.

3.10.22 Underlining

Use this property to specify if and how text is underlined.

XML Code:	`style:text-underline`
Rules:	The value of this property is the underlining style for the text, for example, `single`, `dotted`, `dash`.
DTD:	`<!ATTLIST style:properties fo:text-underline (none|single|double|dotted|dash|long-dash|dot-dash|dot-dot-dash|wave|bold|bold-dotted|bold-dash|bold-long-dash|bold-dot-dash|bold-dot-dot-dash|bold-wave|double-wave|small-wave) #IMPLIED>`
Note:	XSL does not support this property but the values `none` and `single` correspond to the values `none` and `underline` in the XSL `fo:text-decoration` property. The `fo:text-decoration` property is also used for crossing out and blinking text.

3.10.23 Underline Color

Use this property to specify the color that is used to underline text.

XML Code:	`style:text-underline-color`
Rules:	The value of this property is either `font-color` or a color. If the value is `font-color`, the current text color is used for underlining.
DTD:	`<!ATTLIST style:properties fo:text-underline-color CDATA #IMPLIED>`
Note:	If you set this property within OpenOffice.org without specifying a `style:text-underline` property for the same style, underlining is switched off.

3.10.24 Font Weight

Use these properties to specify the weight of the font.

XML Code:	`fo:font-weight` (XSL property)
	`fo:font-weight-asian`
	`fo:font-weight-complex`

Rules:	The relative values `lighter` or `bolder` are not supported and only a few distinct numerical values are supported. Unsupported numerical values are rounded off to the next supported value. See Section 3.10.8 for information about when the asian and complex variants of the attribute are evaluated.

DTD:	`<!ATTLIST style:properties fo:font-weight CDATA #IMPLIED>`
	
	`<!ATTLIST style:properties fo:font-weight-asian CDATA #IMPLIED>`
	
	`<!ATTLIST style:properties fo:font-weight-complex CDATA #IMPLIED>`

3.10.25 Text Decoration Word Mode

Use this property to specify whether crossing out and underlining is applied to words only or to portions of text. If crossing out and underlining is applied to text portions, the spaces between words and the words are underlined or crossed out.

XML Code:	`fo:score-spaces`
Rules:	
DTD:	`<!ATTLIST style:properties fo:score-spaces %boolean; #IMPLIED>`
Notes:	XSL does not support this property. In StarOffice 5.2, this property was called `style:decorate-words-only`.

3.10.26 Letter Kerning

Use this property to enable or disable kerning between characters.

XML Code:	`style:letter-kerning`
Rules:	
DTD:	`<!ATTLIST style:properties style:letter-kerning %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.10.27 Text Blinking

Use this property to specify whether or not text blinks.

XML Code:	`style:text-blinking`
Rules:	
DTD:	`<!ATTLIST style:properties style:text-blinking %boolean; #IMPLIED>`
Note:	XSL does not support this property but the values `false` and `true` correspond to the values `none` and `blink` of the XSL `fo:text-decoration` property. The XSL `fo:text-decoration` can be used to represent this property but it is also used for underlined an crossing out text.

3.10.28 Text Background Color

Use this property to specify the background color to apply to characters.

XML Code:	`style:text-background-color`
Rules:	The value of this property can be `transparent` or a color. The property has the same values as the `fo:background-color` property.
DTD:	`<!ENTITY % transparent_or_color "CDATA">` `<!ATTLIST style:properties style:text-background-color %transparent_or_color; #IMPLIED>`
Note:	Unlike OpenOffice.org, XSL does not distinguish between character and paragraph backgrounds. In OpenOffice.org, if a background is applied to a block element (paragraph), it behaves like a paragraph background and if it is applied to an inline element (a piece of text within a paragraph), it behaves like a character background. Therefore, this property can be transformed to an XSL `fo:background-color` but an additional inline element is required.

3.10.29 Text Combine

Use this property to combine characters so that they are displayed within two lines.

XML Code:	`style:text-combine`
Rules:	The value of this attribute can be `none`, `letters` or `lines`. If the value is `lines`, all characters with this attribute value that immediately follow each other are displayed within two lines of approximately the same length. There can be a line break between any two characters to meet this constraint. If the value of the attribute is `letters`, up to 5 characters are combined within two lines. Any additional character is displayed as normal text.
DTD:	`<!ATTLIST style:properties style:text-combine (none|letters|lines)>`
Note:	See http://www.w3.org/TR/WD-i18n-format/".

3.10.30 Text Combine Start and End Characters

Use these two properties to specify a start and end character that is displayed before and after a portion of text whose `style:text-combine` property has a value of `lines`.

XML Code:	`style:text-combine-start-char` `style:text-combine-end-char`
Rules:	
DTD:	`<!ATTLIST style:properties style:text-combine-start-char % character;>` `<!ATTLIST style:properties style:text-combine-end-char % character;>`

3.10.31 Text Emphasis

Use this property to emphasize text in Asian documents.

XML Code:	`style:text-emphasize`
Rules:	The value of this attribute consists of two space-separated values. The first value represents the style to use for emphasis and it can be `none`, `accent`, `dot`, `circle`, or `disc`. The second value represents the position of the emphasis and it can be `above` or `below`. If the first value is `none`, this value can be omitted.
DTD:	`<!ATTLIST style:properties style:text-emphasize CDATA #IMPLIED>`

Note:	OpenOffice.org supports the following combined values:
	
	`none`
	`dot above`
	`dot below`
	`accent above`
	`circle above`
	
	When the document is imported, `disc above` is changed to `dot above`, while any `below` value is changed to a `dot below` value.

3.10.32 Text Autospace

Use this property to specify whether to add space between asian, western, and complex text.

XML Code:	`style:text-autospace`
Rules:	The OpenOffice.org software only supports the values `none` and `ideograph-alpha`.
DTD:	`<!ATTLIST style:properties style:text-autospace (none | ideograph-alpha) #IMPLIED>`

3.10.33 Text Scale

Use this property to decrease or increase the width of the text. To achieve scaling, change the font width.

XML Code:	`style:text-scale`
Rules:	
DTD:	`<!ATTLIST style:properties style:text-scale %percentage #IMPLIED>`

3.10.34 Text Rotation Angle

Use this property to specify an angle to which to rotate the text. You can rotate text to an angle of 90 or 270 degrees. If this attribute is specified for more than one character, all text containing these characters is rotated.

XML Code:	`style:text-rotation-angle`
Rules:	The value of this attribute can be `0`, `90`, or `270`. For any angle greater than 359 the remainder of a division by 360 is used. Any angle other than 0, 90 or 270 is rounded to the nearest possible value.
DTD:	`<!ATTLIST style:properties style:text-rotation-angle %integer; #IMPLIED>`

3.10.35 Text Rotation Scale

If text is rotated, this property specifies whether the width of the text should be scaled to fit into the current line height or the width of the text should remain fixed, therefore changing the current line height.

XML Code:	`style:text-rotation-scale`
Rules:	
DTD:	`<!ATTLIST style:properties style:text-rotationscale (fixed| line-height) #IMPLIED>`

3.10.36 Punctuation Wrap

Use this property to determine whether or not a punctuation mark, if one is present, can be placed in the margin area at the end of a full line of text. This is a common setting in East Asian typography.

XML Code:	`style:punctuation-wrap`
Rules:	
DTD:	`<!ATTLIST style:properties style:punctuation-wrap (simple | hanging) #IMPLIED>`

3.10.37 Line Break

Use this property to select the set of line breaking rules to use for text.

XML Code:	`style:line-break`
Rules:	
DTD:	`<!ATTLIST style:properties style:line-break (normal | strict) #IMPLIED>`

3.11 Paragraph Formatting Properties

You can apply paragraph formatting properties to paragraphs and paragraph styles.

3.11.1 Fixed Line Height

Use this property to specify a fixed line height either as a length or a percentage that relates to the highest character in a line. A special value of `normal` activates the default line height calculation. It is also used to deactivate the effects of the `style:line-height-at-least` and `style:line-spacing` properties.

XML Code:	`fo:line-height` (XSL property)
Rules:	The value of this property can be a length, a percentage, or a value of `normal`.
DTD:	`<!ATTLIST style:properties fo:line-height (normal|%length;|%percentage;) #IMPLIED>`
Note:	The `fo:line-height`, `style:line-height-at-least` and `style:line-spacing` properties are mutually exclusive and cancel each other out. The result of specifying two or more of these properties within one item set element is undefined. XSL supports this property but it does not support a number property value.

3.11.2 Minimum Line Height

Use this property to specify a minimum line height.

XML Code:	`style:line-height-at-least`
Rules:	The value of this property is a length. There is no `normal` value for the property.
DTD:	`<!ATTLIST style:properties style:line-height-at-least %length; #IMPLIED>`
Note:	XSL does not support this property. The `fo:line-height`, `style:line-height-at-least` and `style:line-spacing` properties are mutually exclusive and cancel each other out. The result of specifying two or more of these properties within one item set elements is undefined.

3.11.3 Line Distance

Use this property to specify a fixed distance between two lines

XML Code:	`style:line-spacing`
Rules:	There is no `normal` value for this property.
DTD:	`<!ATTLIST style:properties style:line-spacing %length #IMPLIED>`
Note:	XSL does not support this property. The `fo:line-height`, `style:line-height-at-least` and `style:line-spacing` properties are mutually exclusive and cancel each other out. The result of specifying two or more of these properties within one item set element is undefined.

3.11.4 Text Align

Use this property to specify how to align text in paragraphs.

XML Code:	`fo:text-align` (XSL property)
Rules:	The value of this property can be `start`, `end`, `center`, or `justify`. If there are no values specified for the `style:text-align-last` and `style:justify-single-word` properties within the same item set element, the values of these properties are set to `left` and `false` respectively.
DTD:	`<!ATTLIST style:properties fo:text-align (start|end|center|justify) #IMPLIED>`
Notes:	At present, the values `page-inside` and `page-outside` are not supported. In StarOffice 5.2, the attribute value `justify` was called `justified`.

3.11.5 Text Align of Last Line

Use this property to specify how to align the last line of a justified paragraph.

XML Code:	`style:text-align-last`
Rules:	The value of this property can be `start`, `center`, or `justify`. This property is ignored if it not accompanied by an `fo:text-align` property. If there are no values specified for the `fo:text-align` and `style:justify-single-word` properties, these values of these properties is set to `left` and `false` respectively.
DTD:	`<!ATTLIST style:properties style:text-align-last (start|center|justify) #IMPLIED>`
Note:	In StarOffice 5.2, the attribute value `justify` was called `justified`.

3.11.6 Justify Single Word

If the last line in a paragraph is justified, use this property to specify whether or not a single word should be justified.

XML Code:	`style:justify-single-word`
Rules:	If there are no values specified for the `fo:text-align` and `style:text-align-last` properties, the values of these properties are set to `left`. This means that specifying a `style:justify-single-word` property without specifying a `style:text-align` and `style:text-align-last` property has no effect.
DTD:	`<!ATTLIST style:properties style:justify-single-word %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.11.7 Break Inside

Use this property to control whether page or column breaks are allowed within a paragraph.

XML Code:	`style:break-inside`
Rules:	
DTD:	`<!ATTLIST style:properties style:break-inside (auto|avoid) #IMPLIED>`
Note:	XSL does not support this property.

3.11.8 Widows

Use this property to specify the minimum number of lines allowed at the top of a page to avoid paragraph **widows**.

XML Code:	`fo:widows` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:widows %number; #IMPLIED>`
Note:	Unlike XSL, this property affects column breaks and page breaks.

3.11.9 Orphans

Use this property to specify the minimum number of lines required at the bottom of a page to avoid paragraph **orphans**.

XML Code:	`fo:orphans` (XSL property)
Rules:	
DTD:	`<!ATTLIST style:properties fo:orphans %number; #IMPLIED>`
Note:	Unlike XSL, this property affects column breaks and page breaks.

3.11.10 Tab Stops

Use the tab stop elements to specify tab stop definitions.

XML Code:	`<style:tab-stops>` `<style:tab-stop>`
Rules:	Every tab stop position is represented by a single `<style:tab-stop>` element that is contained in the `<style:tab-stops>` element.
DTD:	`<!ELEMENT style:tab-stops (style:tab-stop)*>` `<!ELEMENT style:tab-stop EMPTY>`
Note:	XSL does not support tab stops.

The attributes that you can associate with the `<style:tab-stops>` and `<style:tab-stop>` elements are:

- Tab position
- Tab type
- Delimiter character
- Leader character

Tab Position

The `style:position` attribute specifies the position of a tab stop.

XML Code:	`style:position`
Rules:	This attribute is associated with the `<style:tab-stop>` element and its value is a length.
DTD:	`<!ATTLIST style:tab-stop style:position %length; #REQUIRED>`

Tab Type

The `style:type` attribute specifies the type of tab stop.

XML Code:	`style:type`
Rules:	This attribute is associated with the `<style:tab-stop>` element and its value can be `left`, `center`, `right`, or `char`.
DTD:	`<!ATTLIST style:tabtype style:type (left|center|right|char) "left">`

Delimiter Character

The `style:char` attribute specifies the delimiter character for tab stops of type `char`.

XML Code:	`style:char`
Rules:	This attribute is associated with the `<style:tab-stop>` element and it *must* be present if the value of the `style:type` attribute is `char`. If the value of `style:type` attribute is not `char`, it is ignored. The value of the attribute must be a single UNICODE character.
DTD:	`<!ATTLIST style:tab-stop style:char %char; #IMPLIED>`

Leader Character

The `style:leader-char` attribute specifies the leader character to use for tab stops.

XML Code:	`style:leader-char`
Rules:	This attribute is associated with the `<style:tab-stop>` element and its value must be a single UNICODE character.
DTD:	`<!ATTLIST style:tab-stop style:leader-char %char; " ">`

3.11.11 Hyphenation

Use this property to enable or disable automatic hyphenation.

XML Code:	`fo:hyphenate`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenate %boolean; #IMPLIED>`
Implementation limitation:	At present, if you enable hyphenation, OpenOffice.org XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenation-keep` to `none` • `fo:hyphenation-remain-char-count` and `fo:hyphenation-push-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit`

3.11.12 Hyphenation Keep

Use this property to enable or disable the hyphenation of the last word on a page.

XML Code:	`fo:hyphenation-keep`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-keep (none|page|column|spread) #IMPLIED>`
Implementation limitation:	At present, this property is not supported. If you enable this property, OpenOffice.org XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate` to `false` • `fo:hyphenation-remain-char-count` and `fo:hyphenation-push-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit` The values `none` and `page` can be set,but they will never be evaluated.

3.11.13 Hyphenation Remain Char Count

Use this property to specify the number of characters that must be present before a hyphenation character.

XML Code:	`fo:hyphenation-remain-char-count`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-remain-char-count % number; #IMPLIED>`
Implementation limitation:	At present, if you enable this property, OpenOffice.org XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate` to `false` • `fo:hyphenation-keep` to `none` • `fo:hyphenation-push-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit`

3.11.14 Hyphenation Push Char Count

Use this property to specify the minimum number of characters that are moved to the next line.

XML Code:	`fo:hyphenation-push-char-count`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-push-char-count % number; #IMPLIED>`
Implementation limitation:	At present, if you enable this property, OpenOffice.org XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate property` to `false` • `fo:hyphenation-keep` to `none` • `fo:hyphenation-remain-char-count` to `0` • `fo:hyphenation-ladder-count` to `no-limit`

3.11.15 Maximum Hyphens

Use this property to specify the maximum number of successive lines that can contain a hyphenated word.

XML Code:	`fo:hyphenation-ladder-count`
Rules:	
DTD:	`<!ATTLIST style:properties fo:hyphenation-ladder-count (no-limit|%number;) #IMPLIED>`
Note:	In StarOffice 5.2, there was a value `none` instead of `no-limit`.
Implementation limitation:	At present, if you enable this property, OpenOffice.org XML sets the following property values unless they are specified within the same item set element: • `fo:hyphenate property` to `false` • `fo:hyphenation-keep` to `none` • `fo:hyphenation-remain-char-count` and `fo:hyphenation-push-char-count` to `0`

3.11.16 Drop Caps

Use the <style:drop-cap> element to specify if the first character or more of a paragraph is displayed in a larger font.

XML Code:	<style:drop-cap>
Rules:	This element can be contained in a <style:properties> element.
DTD:	<!ELEMENT style:drop-cap EMPTY>

The attributes that you can associate with the <style:drop-cap> element are:

- Length
- Lines
- Distance
- Text style

Length

The style:length attribute specifies the number of characters that are dropped.

XML Code:	style:length
Rules:	The value of this attribute can be a number or word, which indicates that the first word should be dropped.
DTD:	<!ATTLIST style:drop-cap style:length (%number;|word) "1">
Note:	XSL does not support drop caps but a conversion to XSL may create a formatting object that contains the dropped characters.

Lines

The style:lines attribute specifies the number of lines that the dropped characters should encircle.

XML Code:	style:lines
Rules:	If the value of this attribute is 1 or 0, drop caps is disabled.
DTD:	<!ATTLIST style:drop-cap style:lines %number; "1">

Distance

The style:distance attribute specifies the distance between the last dropped character and the first of the remaining characters of each line.

XML Code:	style:distance
Rules:	The value of this attribute is a length.
DTD:	<!ATTLIST style:drop-cap style:distance %length; "0cm">

Text Style

The `style:style-name` attribute specifies the text style to apply to the dropped characters.

XML Code:	`style:style-name`
Rules:	
DTD:	`<!ATTLIST style:drop-cap style:style-name CDATA #IMPLIED>`

3.11.17 Register True

Use the `style:register-true` property to ensure that when you are using two-sided printing, the printed lines on both sides of a page match. It also ensures that the text in page columns or text box columns is arranged in such a way that the text baselines seem to run from one column to another.

XML Code:	`style:register-true`
Rules:	
DTD:	`<!ATTLIST style:properties style:register-true %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.11.18 Numbering Style

See Chapter 2 for information on the number style formatting properties.

3.11.19 Left and Right Margins

Use these properties to specify the left and right margins for a paragraph.

XML Code:	`fo:margin-left` and `fo:margin-right` (XSL properties)
Rules:	These two properties must be attached to an item set element together with the `fo:text-indent` property. If any of the properties is missing, its value is assumed to be 0cm. The value `auto` is not supported.
DTD:	`<!ATTLIST style:properties fo:margin-left (%number;|% percantage;) #IMPLIED>` `<!ATTLIST style:properties fo:margin-right (%number;|% percantage;) #IMPLIED>`
Note:	Unlike XSL, percentage values for these attributes can be used within paragraph styles and relate to the margins of the parent paragraph style of the width of the attribute neighborhood.

3.11.20 Text Indent

Use this property to specify an positive or negative indent for the first line of a paragraph.

XML Code:	`fo:text-indent` (XSL property)
Rules:	This property must be attached to an item set element together with the `fo:margin-left` and `fo:margin-right` properties. If any of these properties is missing, its value is assumed to be 0cm.
DTD:	`<!ATTLIST style:properties fo:text-indent %number; #IMPLIED>`
Note:	Unlike XSL, percentage values can be used within paragraph styles. They percentages relate to the parent paragraph style.

3.11.21 Automatic Text Indent

Use this property to specify that the first line of a paragraph is indented by a value that is based on the current font size.

XML Code:	`style:auto-text-indent`
Rules:	This property must be attached to an item set element together with the `fo:margin-left` and `fo:margin-right` properties. If any of these properties is missing, its value is assumed to be 0cm. If this property is attached to an item set element together with a `fo:text-indent` property that has a value of `true`, the `fo:text-indent` property is ignored.
DTD:	`<!ATTLIST style:properties style:auto-text-indent %boolean; #IMPLIED>`
Note:	XSL does not support this property.

3.11.22 Top and Bottom Margins

Use these properties to specify the top and bottom margins for paragraphs.

XML Code:	`fo:margin-top` and `fo:margin-bottom` (XSL properties)
Rules:	These two properties must be attached to an item set element simultaneously. If one of the properties is missing, its value is assumed to be 0cm. The value `auto` is not supported.
DTD:	`<!ATTLIST style:properties fo:margin-top CDATA #IMPLIED>` `<!ATTLIST style:properties fo:margin-bottom CDATA #IMPLIED>`
Note:	Unlike XSL, percentage values can used within paragraph styles. The percentages relate to the parent paragraph style instead of the attribute neighborhood.

3.11.23 Page Sequence Entry Point

See Chapter 4 for detailed information on page sequence entry points.

3.11.24 Break Before and Break After

Use these properties to insert a page or column break before or after a paragraph.

XML Code:	`fo:break-before` and `fo:break-after` (XSL properties)
Rules:	These two properties are mutually exclusive. If they are attached to an item set element simultaneously, the result is undefined. The values `odd-page` and `even-page` behave like a `page` value.
DTD:	`<!ATTLIST style:properties fo:break-before (auto|column|page) #IMPLIED>` `<!ATTLIST style:properties fo:break-after (auto|column|page) #IMPLIED>`

3.11.25 Paragraph Background Color

Use this property to specify the background color of a paragraph.

XML Code:	`fo:background-color` (XSL property)
Rules:	The value of this attribute can be either `transparent` or it can be a color. If the value is `transparent`, it switches off any background image that is specified by a `<style:background-image>` element within the same item set element.
DTD:	`<!ATTLIST style:properties fo:background-color % transparent_or_color #IMPLIED>`

3.11.26 Paragraph Background Image

Use this property to specify a background image for a paragraph.

The background image can be stored in one of the following ways:

- The image data is stored in an external file. Use the Xlink attributes to specify the location of the image.
- The image data is contained in an `<office:binary-data>` subelement in BASE64 encoding.

XML Code:	`<style:background-image>`
Rules:	This element must be contained within a properties element. The background image is only displayed if one of the following conditions exist: • There is an `xlink:href` attribute attached to the element. • There is an `<office:binary-data>` element contained in the element. If the `<style:background-image>` element is empty and if there is no color specified by an `fo:background-color` element in the same properties element, OpenOffice.org XML sets the background color to transparent.

DTD:	`<!ELEMENT style:background-image (office:binary-data?)>`
	`<!ATTLIST style:background-image xlink:type (simple) #IMPLIED>`
	`<!ATTLIST style:background-image xlink:show (embed) #IMPLIED>`
	`<!ATTLIST style:background-image xlink:actuate (onLoad) #IMPLIED>`

Note:	XSL is not suitable for displaying background images because it is not based on XLink.

The attributes that you can associate with the `<style:background-image>` element are:

- Repetition

- Position
- Filter

Repetition

The `style:repeat` attribute specifies whether a background image is repeated or stretched in a paragraph.

XML Code:	`style:repeat`
Rules:	This attribute is attached to the `<style:background-image>` element and its value can be `no-repeat`, `repeat`, or `stretch`.
DTD:	`<!ATTLIST style:background-image style:repeat (no-repeat|repeat|stretch) "repeat">`
Note:	This attribute is similar to the XSL `fo:background-repeat` property, except that XSL does not support the `stretch` value but supports the values `repeat-x` and `repeat-y` instead.

Position

The `style:position` attribute specifies where to position a background image in a paragraph.

XML Code:	`style:position`
Rules:	This attribute is attached to the `<style:background-image>` element and its value can be a space separated combination of `top`, `center`, or `bottom` for the vertical position and `left`, `center`, or `right` for the horizontal position. The vertical and horizontal positions can be specified in any order and if you wish, you can specify just one position in which case the other position defaults to `center`.
DTD:	`<!ATTLIST style:background-image style:repeat CDATA "center">`
Note:	This attribute is similar to the XSL fo:background-position property except that XSL supports a wider range of values.

Filter

The `style:filter-name` attribute specifies the internal OpenOffice.org filter name that is used to load the image into the document.

XML Code:	`style:filter-name`
Rules:	This attribute is attached to the `<style:background-image>` element.
DTD:	`<!ATTLIST style:background-image style:filter CDATA #IMPLIED>`

3.11.27 Border

Use the border properties to specify the border properties for paragraphs.

XML Code:	`fo:border`
	`fo:border-top`
	`fo:border-bottom`
	`fo:border-left`
	`fo:border-right`

Rules:	The `fo:border` property applies to all four sides of a paragraph while the other properties apply to one side only.

DTD:	`<!ATTLIST style:properties fo:border CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-top CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-bottom CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-left CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:border-right CDATA #IMPLIED>`

Implementation limitations:	At present, all four borders must be set simultaneously by using either the `fo:border` property or by attaching all four of the other border properties to an item set element. In the latter case, if one or more of the properties is missing their values are assumed to be `none`. The only border styles supported are `none` or `hidden`, `solid`, and `double`. Any other border style specified is displayed as `solid`. Transparent borders are not supported and the border widths `thin`, `medium`, and `thick` are mapped to lengths. In addition, only some distinct border widths are supported. Unsupported widths are rounded up to the next supported width. If there are no padding properties specified within the same item set element, a default padding is used for sides that have a border. A value of 0cm is used for sides without a border.

3.11.28 Border Line Width

If the line style for a border is `double`, use the border line properties to individually specify the width of the inner and outer lines and the distance between them.

XML Code:	`style:border-line-width`
	`style:border-line-width-top`
	`style:border-line-width-bottom`
	`style:border-line-width-left`
	`style:border-line-width-right`

Rules:	The `style:border-line-width` specifies the line widths of all four sides, while the other attributes specify the line widths of one side only. The value of the attributes can be a list of three space-separated lengths, as follows: • The first value specifies the width of the inner line • The second value specified the distance between the two lines • The third value specifies the width of the outer line

DTD:	`<!ATTLIST style:properties style:border-line-width CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-top CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-bottom CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-left CDATA #IMPLIED>`
	`<!ATTLIST style:properties style:border-line-width-right CDATA #IMPLIED>`

Implementation limitations:	Only a few distinct width triples are supported. Unsupported width triples are rounded to a supported width triple. The result of specifying a border line width without specifying a border width style of `double` for the same border is undefined.
Note:	XSL does not support these border line width properties.

3.11.29 Padding

Use the padding properties to specify the spacing around a paragraph.

XML Code:	`fo:padding`
	`fo:padding-top`
	`fo:padding-bottom`
	`fo:padding-left`
	`fo:padding-right`

Rules:	

DTD:	`<!ATTLIST style:properties fo:padding CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-top CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-bottom CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-left CDATA #IMPLIED>`
	`<!ATTLIST style:properties fo:padding-right CDATA #IMPLIED>`

Implementation limitations:	At present, the value of these properties can be a non-zero value if there is a border at the same side and the border is specified within the same item set element. The value can be zero if there is no border at the same side. If an item set element contains a padding specification for one but not all four sides, a zero or a default padding is assigned to these sides depending on whether or not there is a border at that side. If you specify a padding for one or more sides without specifying borders within the same item set element, OpenOffice.org XML switches off all borders that are not set.

3.11.30 Shadow

Use the `style:shadow` property to specify a shadow effect for the paragraph.

XML Code:	`style:shadow`
Rules:	The valid values for this attribute are the same as the values for the `fo:text-shadow` property. See Section 3.10.21 for information.
DTD:	`<!ATTLIST style:properties style:shadow CDATA #IMPLIED>`
Implementation limitations:	At present, only one shadow effect is supported at a time. The absolute values of the vertical and horizontal shadow positions must be the same and there is no blur radius.
Note:	XSL does not support this property.

3.11.31 Keep with Next

Use the `style:keep-with-next` property to specify whether or not to keep the current paragraph and the

next paragraph together on a page or in a column after a break is inserted.

XML Code:	`style:keep-with-next`
Rules:	
DTD:	`<!ATTLIST style:properties style:keep-with-next %boolean; #IMPLIED>`
Note:	In StarOffice 5.2, this attribute was called `fo:keep-with-next`.

3.11.32 Line Numbering

See Section 3.5 for detailed information on line numbering formatting properties.

3.11.33 Text Autospace, Punctuation Wrap, Line Break

See Sections 3.10.32, 3.10.36 and 3.10.37 for information about these properties.

3.11.34 Vertical Alignment

Use the `style:vertical-align` property to determine the vertical position of a character. In this manual, characters are aligned according to their baseline, which is the default for most European languages. Alternatively, you can vertically align characters as follows:

- To the bottom of the line.
- To the top of the line.
- To the center of the line.
- Automatically, which sets the vertical alignment to suit the text rotation. Text that is rotated 0 or 90 degrees is aligned to the baseline, while text that is rotated 270 degrees is aligned to the center of the line.

The following graphic illustrates the effect of the vertical alignment property when it is set to baseline, top, bottom, and centerrespectively.

XML Code:	`style:vertical-align`
Rules:	
DTD:	`<!ATTLIST style:properties style:vertical-align (auto|baseline|top|bottom|middle) "auto">`
Note:	This property is similar to the vertical-align property in XSL. The property does not support all of the XSL values, and additionally supports automatic vertical alignment.

3.12 Section Formatting Properties

You can apply section formatting properties to section descriptions.

3.12.1 Section Background

The background formatting properties for sections are the same as the background properties for paragraphs. See Section 3.11.25 and 3.11.26 for information on background formatting properties for paragraphs.

3.12.2 Columns

The `<style:columns>` element contains the column elements for a section.

XML Code:	`<style:columns>`
Rules:	This element can contain `<style:column>` elements that specify each column individually (see Section 3.12.3). If these elements are not present, all columns are assigned the same width. The `<style:columns>` can contain a `<style:column-sep>` element that describes the separator line between columns. See Section 3.12.4 for information on this element.
DTD:	`<!ELEMENT style:columns (style:column-sep?,(style:column, style:column+)?>`

The attributes that you can associate with the `<style:columns>` element are:

- Column count
- Column gap

Column Count

The `fo:columns-count` attribute specifies the number of columns in a section.

XML Code:	`fo:columns-count`
Rules:	This attribute is essential.
DTD:	`<!ATTLIST style:columns fo:column-count %number #REQUIRED>`
Note:	This attribute has the same name as an XSL property but it is attached to a different element.

Column Gap

If the `<style:columns>` element does not contain individual `<style:column>` elements, you can specify the gap between columns using the `fo:column-gap` attribute.

XML Code:	`fo:column-gap`
Rules:	If there are individual column elements, this attribute is ignored.
DTD:	`<!ATTLIST style:columns fo:column-gap %length #IMPLIED>`
Note:	This attribute has the same name as an XSL property but it is attached to a different element.

3.12.3 Column Specification

The <style:column> element can be contained in a <style:columns> element, to specify details of an individual column.

XML Code:	<style:column>
Rules:	This element is contained in the <styles:columns> element. There can be either no column elements or there can be the same number of column elements as specified by the fo:column-count attribute.
DTD:	<!ELEMENT style:column EMPTY>
Note:	In XSL, it is not possible to specify columns individually.

The attributes that you can associate with the <style:column> element are:

- Column width
- Column left, right, upper, and lower space

Column Width

Use the style:rel-width attribute to specify the width of a column.

The column width is not specified in a percentage a length, but rather in terms of relative weights. The total space available for the entire table is distributed among its columns according to its relative widths. For example, if three columns are assigned the relative widths 1, 2 and 3, then the first column will take up 1/6 of the available width, the second will take up 1/3, and the last column will take up 1/2 of the available space. To achieve these figures, all given relative widths must be summed up (six in the example), and then each column will get as much space as the proportion of its own relative width to the sum of all relative widths indicates (3/6 = 1/2 for the last column in the example).

XML Code:	style:rel-width
Rules:	The column widths are specified as numbers instead of lengths. To get the absolute column width, the space that is available for a columned area is distributed among the columns proportional to these numbers.
DTD:	<!ATTLIST style:column style:rel-width %number; #REQUIRED>

Column Left, Right, Upper, and Lower Space

For each column, you can specify the left, right, upper, and lower space. The right space of a column together with the left space of the next column corresponds to the gap between two columns. If a columned area contains a separator line between columns, the space that is occupied by the line is contained within the left and right spaces and therefore is not added to them.

XML Code:	For left and right spaces:
	`fo:start-indent`
	`fo:end-indent`
	For upper and lower spaces:
	`fo:space-before`
	`fo:space-after`

Rules:	

DTD:	`<!ATTLIST style:column fo:start-indent %length: "0cm">`
	`<!ATTLIST style:column fo:end-indent %length: "0cm">`
	`<!ATTLIST style:column fo:space-before %length: "0cm">`
	`<!ATTLIST style:column fo:space-after %length: "0cm">`

3.12.4 Column Separator

The `<style:column-sep>` element specifies the separator line to use between columns.

XML Code:	`<style:column-sep>`
Rules:	This element can be contained in a `<style:columns>` element to specify the type of separator line to use between columns.
DTD:	`<!ELEMENT style:column-sep EMPTY>`
Note:	XSL does not support column separators.

The attributes that you can associate with the `<style:column-sep>` element are:

- Line style
- Line width
- Line height
- Vertical line alignment
- Line color

Line Style

Use the `style:style` attribute to specify the line style of the column separator line.

XML Code:	`style:style`
Rules:	
DTD:	`<!ATTLIST style:column-sep style:style` `(none|solid|dotted|dashed|dot-dashed) "solid">`

Line Width

Use the `style:width` attribute to specify the width of the column separator line.

XML Code:	`style:width`
Rules:	
DTD:	`<!ATTLIST style:column-sep style:width %length; #REQUIRED>`

Line Height

Use the `style:height` to specify the height of the column separator line.

XML Code:	`style:height`
Rules:	The value of this attribute is a percentage that relates to the height of the columned area.
DTD:	`<!ATTLIST style:column-sep style:height %percentage; "100%">`

Vertical Line Alignment

Use the `style:vertical-align` attribute to specify how to vertically align a line that is less than 100% of its height within the columned area.

XML Code:	`style:vertical-align`
Rules:	The value of this attribute can be either `top`, `middle`, or `bottom`.
DTD:	`<!ATTLIST style:column-sep style:vertical-align (top|middle|bottom) "top">`

Line Color

Use the `style:color` attribute to specify the color of the column separator line.

XML Code:	`style:color`
Rules:	
DTD:	`<!ATTLIST style:column-sep style:color %color; "#000000">`

3.12.5 Protect

Sections marked with the `style:protect` attribute should not be changed. The user interface should prevent the user from manually making any changes.

XML Code:	`style:protect`
Rules:	The `style:protect` attribute is set by default for linked sections or indexes. Removing the protection makes these sections accessible to the user, but updating the links or the index will not preserve the changes.
DTD:	`<!ATTLIST style:properties style:protect %boolean; "false">`

3.13 Change Tracking in Text Documents

This section describes how the OpenOffice.org software tracks changes to text content.

3.13.1 Tracked Changes

All tracked changes to text documents are stored in a list. The list contains an element for each change made to the document. If the `<text:tracked-changes>` element is absent, change tracking is not enabled.

XML Code:	`<text:tracked-changes>`
Rules:	
DTD:	`<!ELEMENT text:tracked-changes (text:changed-region)+>`

Track Changes

This attribute determines whether or not change tracking is enabled.

XML Code:	`text:track-changes`
Rules:	
DTD:	`<!ATTLIST text:tracked-changes text:track-changes %boolean; "true">`

Protection

The OpenOffice.org software supports change tracking protection. If protection is enabled, a user can not enable or disable the tracking of changes, and also can not accept or reject any changes. Protection is enabled by supplying a protection key.

XML Code:	`text:protection-key`
Rules:	
DTD:	`<!ATTLIST text:tracked-changes text:protection-key CDATA #IMPLIED>`

3.13.2 Changed Regions

For every changed region of a document, there is one entry in the list of tracked changes. This entry contains a list of all changes that were applied to the region. The start and end of this region are marked by the start and end elements that are described in the next section.

XML Code:	`<text:changed-region>`
Rules:	Every element has an ID. The elements that mark the start and end of a region use this ID to identify the region to which they belong.
DTD:	`<!ELEMENT text:changed-region (text:insertion|text:deletion|text:format-change)+>` `<ATTLIST text:changed-region text:id ID #REQUIRED>`

The <text:changed-region> element has an additional attribute that modifies the default behavior when the application shows deleted redlines inline in a document. By default, the redline is merged into the text so that the starting and ending paragraphs are combined with the surrounding text to form merged paragraphs. To change this behavior, you can surpress the last paragraph by setting the text:merge-last-paragraph attribute to false.

XML Code:	text:merge-last-paragraph
Rules:	
DTD:	<ATTLIST text:changed-region text:merge-last-paragraph % boolean; "true">

3.13.3 Region Start and End

There are three elements that mark the start and the end of a changed region, as follows:

- Change start element – <text:change-start>
  This element marks the start of a region with content where text has been inserted or the format has been changed.
- Change end element – <text:change-end>
  This element marks the end of a region with content where text has been inserted or the format has been changed.
- Change position element – <text:change>
  This element marks a position in an empty region where text has been deleted.

XML Code:	<text:change-start> <text:change-end> <text:change>
Rules:	All three elements have an attribute that specifies the ID of the region to which they belong.
DTD:	<!ELEMENT text:change-start EMPTY> <!ELEMENT text:change-end EMPTY> <!ELEMENT text:change EMPTY> <!ATTLIST text:change-start text:region-id IDREF #REQUIRED> <!ATTLIST text:change-end text:region-id IDREF #REQUIRED> <!ATTLIST text:chang text:region-id IDREF #REQUIRED>

3.13.4 Insertion

The <text:insertion> element contains the information that is required to identify any insertion of content. This content can be a piece of text within a paragraph, a whole paragraph, or a whole table. The inserted content is part of the text document itself and is marked by a change start and a change end element.

XML Code:	<text:insertion>
Rules:	
DTD:	<!ELEMENT text:insertion (office:change-info)>

Example: Insertion of text

```
<text:tracked-changes>
  <text:changed-region text:id="c001">
    <text:insertion>
      <office:change-info office:chg-author="Michael Brauer"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04"/>
    </text:insertion>
  </text:changed-region>
</text:tracked-changes>
...
<text:p>
  This is the original text<text:change-start text:region-id="c001"/>,
  but this has been added</text:change-end text:region-id="c001"/>.
</text:p>
```

3.13.5 Deletion

A <text:deletion> element contains content that was deleted while change tracking was enabled. The position where the text was deleted is marked by the change position element (<text:change>).

XML Code:	<text:deletion>
Rules:	If part of a paragraph was deleted, the text that was deleted is contained in this element as a paragraph element. If the deleted text is reinserted into the document, the paragraph is joined with the paragraph where the deletion took place.
DTD:	<!ELEMENT text:deletion (office:change-info,%text;)>

Example: Deletion of text

```
<text:tracked-changes>
  <text:changed-region text:id="c002">
    <text:deletion>
      <office:change-info office:chg-author="Michael Brauer"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04">
      <text:p>
        , but this has been deleted
      </text:p>
    </text:deletion>
  </text:changed-region>
</text:tracked-changes>
...
<text:p>
  This is the original text<text:change text:region-id="c002"/>.
</text:p>
```

This example shows:

- Deleted text = `but this has been deleted`
  This text is contained in the <text:p> element within the <text:deletion> element.
- Current text = `This is the original text.`
  This text is contained in the <text:p> element at the end of the example.
- Original text before deletion took place = `This is the original text, but this has been deleted.`

3.13.6 Format Change

A format change element represents any change in formatting attributes. The region where the change took place is marked by a change start and a change end element.

XML Code:	`<text:format-change>`
Rules:	
DTD:	`<!ELEMENT text:format-change (office:change-info)>`
Note:	A format change element does not contain the actual changes that took place.

3.14 Optional Information

You can include the information described in this section in an XML document to improve its performance. However, it is not essential.

3.14.1 Auto Mark File for Alphabetical Indices

To ease the creation of alphabetical indices, OpenOffice.org allows you to assign a so-called auto mark file. This file contains a list of keywords, and upon update of an alphabetical index every occurrence of such a keyword in the document is marked as an alphabetical index entry.The auto mark file element is an XLink, and the URL is given by the `xlink:href` attribute.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST text:alphabetical-index-auto-mark-file xlink:href CDATA #REQUIRED>` `<!ATTLIST text:alphabetical-index-auto-mark-file xlink:type (simple) fixed "simple">`

CHAPTER 4

Table Content

This chapter describes the OpenOffice.org XML representation of table and spreadsheet content. It contains the following sections:

- General Introduction to OpenOffice.org Tables
- Calculation Settings
- Change Tracking
- Tables
- Columns
- Rows
- Cells
- Table Cell Content Validations
- Subtables
- Label Ranges
- Named Expressions
- Filters
- Database Ranges
- Data Pilot Tables
- Consolidation
- DDE Links
- Table Formatting Properties
- Column Formatting Properties
- Table Row Formatting Properties
- Table Cell Formatting Properties

4.1 General Introduction to OpenOffice.org Tables

OpenOffice.org Writer and OpenOffice.org Calc documents can include tables, but the internal structure of the tables created using these applications is different. The differences between the table structures in both applications are as follows:

- The structure of OpenOffice.org Calc tables is similar to the structure of un-nested HTML and XSL tables. An OpenOffice.org Calc XML document does not contain any **subtables**, sometimes called nested tables.
- The structure of OpenOffice.org Writer tables is similar to the structure of nested HTML or XSL tables that do not have any vertically merged cells. An OpenOffice.org Writer XML document does not contain vertically merged cells.

If a document that contains either a subtable or vertically merged cells, or both, is converted to XML, the structure of the table may change. This does not affect how the table appears when the document is displayed.

There are several reasons why you need to preserve the internal OpenOffice.org Writer table structure:

1. The OpenOffice.org API and formulas access table cells using names that are derived from the internal table structure.
2. Within an OpenOffice.org Writer table, rows may have a fixed height or background. If the internal OpenOffice.org Writer table structure is not preserved, there could be rows that do not have a corresponding row in the HTML or XSL representation of the table. This could lead to a loss of information.
3. The internal column widths of an OpenOffice.org Writer table do not have to be the same as the displayed column widths.

The representation of tables is based on a grid of rows and columns. Rows take precedence over columns. The table is divided into rows and the rows are divided into cells. Each column includes a column description, but this description does not contain any cells.

Rows and columns appear in **row groups** and **column groups**. These groups specify whether or not to repeat a row or column on the next page.

4.2 Document Protection

You can protect the structure of a spreadsheet document so that users can not insert, delete, move or rename the tables in the document. You can use an optional password to prevent users from resetting the table protection flag to allow editing.

XML Code:	`table:structure-protected` `table:protection-key`
Rules:	

DTD:	`<!ATTLIST office:body table:structure-protected %boolean; "false">`
	
	`<!ATTLIST office:body table:protection-key CDATA #IMPLIED>`

4.3 Calculation Settings

Spreadsheet documents contain settings that affect the calculation of formulas, for example the null date or iteration settings. These settings must be saved in the document in the `<table:calculation-settings>` element.

XML Code:	`<table:calculation-settings>`
Rules:	
DTD:	`<!ELEMENT table:calculation-settings (table:null-date?, table:iteration?)>`

The attributes that you can associate with the <table:calculation-settings> element are:

- Case sensitive
- Precision as shown
- Search criteria must apply to whole cell
- Automatic find labels
- Null year

Case Sensitive

This attribute specifies whether or not to distinguish between upper and lower case in text when comparing cell contents for calculations.

XML Code:	table:case-sensitive
Rules:	
DTD:	<!ATTLIST table:calculation-settings table:case-sensitive %boolean; "true">

Precision as Shown

This attribute specifies whether to perform a calculation using the rounded values displayed in the spreadsheet or using all of the digits in a number.

XML Code:	table:precision-as-shown
Rules:	If the value of this attribute is true, OpenOffice.org Calc performs the calculation using the rounded values displayed in the spreadsheet. If the value of this attribute is false, OpenOffice.org Calc performs the calculation using all of the digits in the number but displays a rounded number as the result.
DTD:	<!ATTLIST table:calculation-settings table:precision-as-shown %boolean; "false">

Search Criteria Must Apply to Whole Cell

This attribute specifies whether or not the specified search criteria, according to the regular expression used, must apply to the entire cell contents.

XML Code:	table:search-criteria-must-apply-to-whole-cell
Rules:	
DTD:	<!ATTLIST table:calculation-settings table:search-criteria-must-apply-to-whole-cell %boolean; "true">

Automatic Find Labels

This attribute specifies whether or not to automatically find the labels of rows and columns.

XML Code:	`table:automatic-find-labels`
Rules:	
DTD:	`<!ATTLIST table:calculation-settings table:automatic-find-labels %boolean; "true">`

Null Year

This attribute specifies the start year for year values that contain only two digits.

XML Code:	`table:null-year`
Rules:	
DTD:	`<!ATTLIST table:calculation-settings table:null-year % positiveInteger; "1930">`

4.3.1 Null Date

This element specifies the null date.

XML Code:	`<table:null-date>`
Rules:	
DTD:	`<!ELEMENT table:null-date EMPTY>`
Implementation limitation:	In Star Office, there are only three possible null dates as follows; `12/30/1899`, `01/01/1900`, and `01/01/1904`.

The attributes that you can associate with the `<table:null-date>` element are:

- Value Type and Date Value

Value Type and Date Value

XML Code:	`table:value-type` `table:date-value`
Rules:	
DTD:	`<!ATTLIST table:null-date table:value-type %valueType; #FIXED "date"` `table:date-value %date; "1899-12-30">`

4.3.2 Iteration

The `<table:iteration>` element enables formulas with iterative references to be calculated after a specific number of iterations. Formulas with iterative references are repeated until the problem is solved. If this element is not enabled, a formula with an iterative reference in a table causes an error message.

XML Code:	`<table:iteration>`
Rules:	
DTD:	`<!ELEMENT table:iteration EMPTY>`

The attributes that you can associate with the `<table:iteration>` element are:

- Status
- Steps
- Maximum difference

Status

This attribute specifies whether or not the iteration is enabled.

XML Code:	`table:status`
Rules:	
DTD:	`<!ATTLIST table:iteration table:status (enable|disable) "disable">`

Steps

This attribute specifies the maximum number of iterations allowed.

XML Code:	`table:steps`
Rules:	
DTD:	`<!ATTLIST table:iteration table:steps %positiveInteger; "100">`

Maximum Difference

This attribute specifies the maximum difference allowed between two calculation results. The iteration is stopped if the result is less than the value of this attribute.

XML Code:	`table:maximum-difference`
Rules:	
DTD:	`<!ATTLIST table:iteration table:maximum-difference %float; "0.001">`

4.4 Change Tracking in Spreadsheets

In OpenOffice.org Writer documents, you can not track changes to tables. In OpenOffice.org Calc documents, you can track changes to tables. This section describes how OpenOffice.org Calc tracks changes to table content.

4.4.1 Tracked Changes

OpenOffice.org Calc stores all of the changes to a spreadsheet document in a list. The list contains an element for each change made to the document. To track the changes to a spreadsheet document, the <table:tracked-changes> element must be present.

XML Code:	<table:tracked-changes>
Rules:	
DTD:	<!ELEMENT table:tracked-changes (table:cell-content-change | table:insertion | table:deletion | table:movement | table:rejection)>

Track Changes

This attribute specifies whether or not the change tracking is enabled.

XML Code:	table:track-changes
Rules:	
DTD:	<!ATTLIST table:tracked-changes table:track-changes %boolean; "true">

Protection

This attribute specifies that the change tracking data is protected, which prevents users from accepting or rejecting changes or disabling the change tracking. If the change tracking is protected, a user can still make changes to the content of the document. The protection is assigned by the application.

XML Code:	table:protection-key
Rules:	
DTD:	<!ATTLIST table:tracked-changes table:protection-key CDATA #IMPLIED>

4.4.2 Dependences

The <table:dependences> element contains the information which change to this change depends. Every element of the tracked-changes can contain a <table:dependences> element.

XML Code:	<table:dependences>
Rules:	
DTD:	<!ELEMENT table:dependences (table:dependence)+>

4.4.3 Dependence

The <table:dependence> element contains the information of one change action which the parent element depends to.

XML Code:	`<table:dependence>`
Rules:	
DTD:	`<!ELEMENT table:dependence EMPTY>`

The attributes that you can associate with this element are:

- ID

4.4.4 Deletions

The `<table:deletions>` element contains all deletions which are effected with this change.

XML Code:	`<table:deletions>`
Rules:	
DTD:	`<!ELEMENT table:deletions (table:cell-content-deletion | table:change-deletion)+>`

4.4.5 Cell Content Deletion

This element contains the content of a deleted cell.

XML Code:	`<table:cell-content-deletion>`
Rules:	
DTD:	`<!ELEMENT table:cell-content-deletion (table:cell-address?, table:change-track-table-cell?)>`

The attributes that you can associate with this element are:

- ID

4.4.6 Change Deletion

This element contains the information of deleted changes.

XML Code:	`<table:change-deletion>`
Rules:	
DTD:	`<!ELEMENT table:change-deletion EMPTY>`

The attributes that you can associate with this element are:

- ID

4.4.7 Insertion

The <table:insertion> element contains the information that is required to identify any insertion of content. This content can be one or more rows, one or more columns, or a table.

XML Code:	<table:insertion>
Rules:	
DTD:	<!ELEMENT table:insertion (office:change-info, table:dependences?, table:deletions?)>

The attributes that you can associate with this element are:

- ID
- Acceptance State
- Rejecting Change ID
- Type
- Position
- Count
- Table

Type

This attribute specifies the type of the insertion.

XML Code:	table:type
Rules:	
DTD:	<!ATTLIST table:insertion table:type (row | column | table) #REQUIRED>

Position

This attribute specifies the position where the insertion was made in the table.

XML Code:	Table:position
Rules:	
DTD:	<!ATTLIST table:insertion table:position %integer; #REQUIRED>

Count

This attribute specifies the count of inserted rows | columns | tables.

XML Code:	table:count
Rules:	
DTD:	<!ATTLIST table:insertion table:count %positiveInteger; "1">

Table

This attribute specifies the table where the insertion is. This attribute only exists, if it is a column or row insertion.

XML Code:	`table:table`
Rules:	
DTD:	`<!ATTLIST table:insertion table:table %Integer; #IMPLIED>`

Example: Insertion of text in a cell

```
<table:tracked-changes>
  <table:insertion table:id="c001" table:acceptance-state="pending" table:
type="column" table:position="5">
      <office:change-info office:chg-author=”Sascha Ballach”
                          office:chg-date=”05/18/99”
                          office:chg-time=”12:56:04”/>
  </table:insertion>
</table:tracked-changes>
```

4.4.8 Deletion

A `<table:deletion>` element contains content that was deleted while change tracking was enabled.

XML Code:	`<table:deletion>`
Rules:	If the content of a cell was deleted, the deleted cell is contained in the `<table:dependences>` element. If the cell content was given while change tracking was enabled the previous cell is contained in the `<table:dependence>` element. If the content was given while change tracking was disabled the previous cell is contained in the `<table:cell-content-deletion>` element.
DTD:	`<!ELEMENT table:deletion (office:change-info, table:dependences?, table:deletions?, table:cut-offs?) >`

The attributes that you can associate with this element are:

- ID
- Acceptance State
- Rejecting Change ID
- Type
- Position
- Table
- Multi Deletion Spanned

Type

This attribute specifies the type of the insertion.

XML Code:	`table:type`
Rules:	
DTD:	`<!ATTLIST table:deletion table:type (row | column | table) #REQUIRED>`

Position

This attribute specifies the position where the deletion was made in the table.

XML Code:	`table:position`
Rules:	
DTD:	`<!ATTLIST table:deletion table:position %integer; #REQUIRED>`

Table

This attribute specifies the table where the deletion is. This attribute only exists, if it is a column or row deletion.

XML Code:	`table:table`
Rules:	
DTD:	`<!ATTLIST table:deletion table:table %Integer; #IMPLIED>`

Multi Deletion Spanned

This attribute specifies the count of deletions which were done with this deletion. Only the first deletion contains this attribute.

XML Code:	`table:base-change-position`
Rules:	
DTD:	`<!ATTLIST table:deletion table:multi-deletion-spanned % integer; #IMPLIED>`

4.4.9 Cut Offs

A `<table:cut-offs>` element contains the information about cut offs. This means e.g. If I delete a column in a range which was moved while change tracking was enable this have to store. This element contains such informations.

XML Code:	`<table:cut-offs>`
Rules:	
DTD:	`<!ELEMENT table:cut-offs (table:insertion-cut-off | table: movement-cut-off+ | (table:insertion-cut-off, table:movement-cut-off+)) >`

4.4.10 Insertion Cut Off

This element contains the information where a insertion was deleted and which.

XML Code:	`<table:insertion-cut-off>`
Rules:	
DTD:	`<!ELEMENT table:insertion-cut-off EMPTY >`

The attributes that you can associate with this element are:

- ID
- position

Position

This attribute specifies the position where the insertion was deleted.

XML Code:	`table:position`
Rules:	
DTD:	`<!ATTLIST table:insertion-cut-off table:position %integer; #REQUIRED>`

4.4.11 Movement Cut Off

This element contains the information where a movement was deleted and which.

XML Code:	`<table:movement-cut-off>`
Rules:	
DTD:	`<!ELEMENT table:movement-cut-off EMPTY >`

The attributes that you can associate with this element are:

- ID
- start position, end position, position

Start Position, End Position, Position

This attribute specifies the position where in the movement was deleted.

XML Code:	`table:start-position; table:end-position; table:position`
Rules:	

DTD:	`<!ATTLIST table:movement-cut-off table:start-position % integer; #IMPLIED>`
	`<!ATTLIST table:movement-cut-off table:end-position %integer; #IMPLIED>`
	`<!ATTLIST table:movement-cut-off table:position %integer; #IMPLIED>`

Example: Deletion of a column which do not contain content

```
<table:tracked-changes>
    <table:deletion table:id=”c002” table:acceptance-state="pending" table:
type="column" table:position="9">
      <office:change-info office:chg-author=”Sascha Ballach”
                          office:chg-date=”05/18/99”
                          office:chg-time=”12:56:04”>
    </table:deletion>
</table:tracked-changes>
```

4.4.12 Movement

A <table:movement> element contains the information that is required to identify any movement of content. This content can be a cell content or a cell range content.

XML Code:	<table:movement>
Rules:	
DTD:	<!ELEMENT table:movement (table:source-range-address, table: target-range-address, office:change-info, table:dependences?, table:deletions?)>

The attributes that you can associate with this element are:

- ID
- Acceptance State
- Rejecting Change ID

4.4.13 Target Range Address, Source Range Address

These elements contains a cell address or a cell range address.

XML Code:	table:target-range-address table:source-range-address
Rules:	
DTD:	<!ELEMENT table:target-range-address EMPTY> <!ELEMENT table:source-range-address EMPTY>

The attributes that you can associate with these elements are:

- Column, Row, and Table
- Start column, End column, Start row, End row, Start table, and End table

Column, Row, and Table

If the range address is a cell address these three attributes are necessary.

XML Code:	table:column table:row table:table
Rules:	
DTD:	<!ATTLIST table:target-range-address table:column %Integer; #IMPLIED> <!ATTLIST table:target-range-address table:row %Integer; #IMPLIED> <!ATTLIST table:target-range-address table:table %Integer; #IMPLIED> <!ATTLIST table:source-range-address table:column %Integer; #IMPLIED> <!ATTLIST table:source-range-address table:row %Integer; #IMPLIED> <!ATTLIST table:source-range-address table:table %Integer; #IMPLIED>

Start Column, End Column, Start Row, End Row, Start Table, and End Table

If the range address is a range address instead of a cell address, these attributes are necessary.

XML Code:	table:start-column table:end-column table:start-row table:end-row table:start-table table:end-table
Rules:	
DTD:	<!ATTLIST table:target-range-address table:start-column % Integer; #IMPLIED> <!ATTLIST table:target-range-address table:start-row %Integer; #IMPLIED> <!ATTLIST table:target-range-address table:start-table % Integer; #IMPLIED> <!ATTLIST table:target-range-address table:end-column % Integer; #IMPLIED> <!ATTLIST table:target-range-address table:end-row %Integer; #IMPLIED> <!ATTLIST table:target-range-address table:end-table %Integer; #IMPLIED> <!ATTLIST table:source-range-address table:start-column % Integer; #IMPLIED> <!ATTLIST table:source-range-address table:start-row %Integer; #IMPLIED> <!ATTLIST table:source-range-address table:start-table % Integer; #IMPLIED> <!ATTLIST table:source-range-address table:end-column % Integer; #IMPLIED> <!ATTLIST table:source-range-address table:end-row %Integer; #IMPLIED> <!ATTLIST table:source-range-address table:end-table %Integer; #IMPLIED>

Example: Moving a cell

```
<table:tracked-changes>
  <table:changed-region table:id="c003">
    <table:moving>
      <office:change-info office:chg-author="Sascha Ballach"
                          office:chg-date="05/18/99"
                          office:chg-time="12:56:04"/>
    </table:moving>
  </table:changed-region>
</table:tracked-changes>
...
<table:table-cell ...>
  <table:change-start table:region-id="c003"/>
  <text:p>
    This is the original text, but this has been added.
  </text:p>
  <table:change-end table:region-id="c003"/>
</table:table-cell>
...
<table:table-cell ...>
  <table:change table:region-id="c003"/>
</table:table-cell>
```

4.4.14 Change Track Cell

A `<table:change-track-table-cell>` element contains all information of a `<table:table-cell>` which are needed in the change track. It contains some additional informations which are only in the change track needed.

XML Code:	`<table:change-track-table-cell>`
Rules:	
DTD:	`<!ELEMENT table:change-track-table-cell (text:p*)>`

Cell Address

If the cell is a formula cell a cell address attribute is necessary.

XML Code:	`table:cell-address`
Rules:	This attribute is only evaluated for cells which are in the tracked changes and a formula cell.
DTD:	`<!ATTLIST table:change-track-table-cell table:cell-address %cell-address; #IMPLIED>`

Matrix Covered

If the cell is a matrix cell and not the base of the matrix this attribute is necessary.

XML Code:	`table:matrix-covered`
Rules:	This attribute is only evaluated for cells which are in the tracked changes and a formula cell.
DTD:	`<!ATTLIST table:change-track-table-cell table:matrix-covered (true | false) "false">`

4.4.15 Cell Content Change

A <table:cell-content-change> element contains the information that is required to identify any change of the cell content.

XML Code:	<table:cell-content-change>
Rules:	
DTD:	<!ELEMENT table:cell-content-change (table:cell-address, office:change-info, table:dependences?, table:deletions?, table:previous)>

The attributes that you can associate with this element are:

- ID
- Acceptance State
- Rejecting Change ID

4.4.16 Cell Address

This element contains a cell address. There is a attribute specified which contains the same information. But I need this element because in a change track the values can be negative or greater than the table range.

XML Code:	table:cell-address
Rules:	
DTD:	<!ELEMENT table:cell-address EMPTY>

The attributes that you can associate with this element are:

- Column, Row, and Table

Column, Row, and Table

XML Code:	table:column, table:row, table:table
Rules:	
DTD:	<!ATTLIST table:cell-address table:column %Integer; #IMPLIED> <!ATTLIST table:cell-address table:row %Integer; #IMPLIED> <!ATTLIST table:cell-address table:table %Integer; #IMPLIED>

4.4.17 Previous

This element contains the previous cell which is overwritten with this change.

XML Code:	table:previous
Rules:	
DTD:	<!ELEMENT table:previous table:change-track-table-cell>

The attributes that you can associate with this element are:

- ID

4.4.18 Rejection

A <table:cell-content-change> element contains the information that is required to identify any change of the cell content.

XML Code:	<table:cell-content-change>
Rules:	
DTD:	<!ELEMENT table:rejection (office:change-info, table: dependences?, table:deletions?)>

The attributes that you can associate with this element are:

- ID
- Acceptance State
- Rejecting Change ID

4.4.19 Common Change Tracking Attributes

You can use the attributes described in this section with the following change tracking elements:

ID

This attribute specifies the ID that OpenOffice.org Calc assigns to the change action.

XML Code:	table:id
Rules:	The value of this attribute must be unique.
DTD:	<!ATTLIST table:rejection table:id CDATA #REQUIRED>

Acceptance State

This attribute specifies whether the change is accepted, rejected, or pending.

XML Code:	table:acceptance-state
Rules:	
DTD:	<!ATTLIST table:rejection table:acceptance-state (accepted | rejected | pending) "pending">

Rejecting Change ID

If a change is rejected, this attribute specifies the ID that OpenOffice.org Calc assigns to the rejected change action.

XML Code:	`table:rejecting-change-id`
Rules:	
DTD:	`<!ATTLIST table:rejection table:rejecting-change-id % positiveInteger; #IMPLIED>`

4.5 Tables

4.5.1 Table

The table element describes a table.

XML Code:	`<table:table>`
Rules:	The content of a table element is one or more groups of columns and rows.

DTD:	`<!ENTITY % table-columns "( table:table-columns | ( table:table-column | table:table-column-group )+ )">`
	`<!ENTITY % table-header-columns "table:table—header-columns">`
	
	`<!ENTITY % table-rows "( table:table-rows | ( table:table-row+ | table:table-row-group ) )">`
	`<!ENTITY % table-header-rows "table:table-header-rows">`
	
	`<!ENTITY % table-column-groups "( (%table-header-columns;?, % table-columns;) |(%table-columns;, %table-header-columns;, % table-columns;?) )">`
	
	`<!ENTITY % table-row-groups "( (%table-header-rows;?, %table-rows;) | (%table-rows;, %table-header-rows;, %table-rows;?) )">`
	
	`<!ELEMENT table:table (table:view-settings, table:table-source?, table:scenario?, table:shapes?, %table-column-groups;,%table-row-groups;)>`

Table Name

The `table:name` attribute specifies the name of a table.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:table table:name CDATA #REQUIRED>`

Table Style

The `table:style-name` attribute describes the formatting properties of a table, such as width and background color. The table style can be either an automatic or common style.

XML Code:	`table:style-name`
Rules:	You define a table style using a `<style:style>` element and a family attribute value of `table`. The `<table:table>` element includes a `table:style-name` attribute that references the style by the `style:name` attribute.
DTD:	`<!ATTLIST table:table table:style-name %style-name #REQUIRED>`

Example: Table Style

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm"
   fo:background-color="light-grey"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  ...
</table:table>
```

DDE Connection

Information to be supplied.

XML Code:	`office:dde-source`
	`office:dde-command`
	`office:dde-mode`

Rules:	

DTD:	`<!ATTLIST table:table office:dde-source CDATA #IMPLIED>`
	`<!ATTLIST table:table office:dde-command CDATA #IMPLIED>`
	`<!ATTLIST table:table office:dde-mode CDATA #IMPLIED>`

Protected

The `table:protected` attribute specifies whether or not a table is protected from editing. If the table is protected, the `table:protection-key` attribute can specify a password to prevent a user from resetting the protection flag to enable editing. If a table is protected, all of the table elements and the cell elements with a `style:cell-protect` attribute set to `true` are protected.

XML Code:	`table:protected` and `table:protection-key`
Rules:	These attributes can be attached to the `<table:table>` element.
DTD:	`<!ATTLIST table:table table:protected %boolean; "false">` `<!ATTLIST table:table table:protection-key CDATA #IMPLIED>`

Print Ranges

The `table:print-ranges` attribute specifies the print ranges of the table. It contains a list of cell addresses or cell range addresses.

XML Code:	`table:print-ranges`
Rules:	This attribute can be associated with the `<table:table>` element.
DTD:	`<!ATTLIST table:table table:print-ranges %cell-range-address-list; #IMPLIED>`

4.5.2 Table Source

If a table is linked to an original table, the original table is represented by a table source element.

XML Code:	`<table:table-source>`
Rules:	
DTD:	`<!ELEMENT table:table-source EMPTY>`

The attributes that you can associate with the `<table:table-source>` element are:

- Mode
- URL
- Filter name
- Table name
- Filter options
- Refresh delay

Mode

The `table:mode` attribute specifies how data should be copied between the linked table and the original table.

XML Code:	`table:mode`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST table:table-source table:mode ( "copy-all" | "copy-results-only" ) "copy-all">`

URL

The XLink attributes specify the URL of the document containing the original table.

XML Code:	`xlink:type`, `xlink:actuate`, and `xlink:href`
Rules:	

DTD:	`<!ATTLIST table:table-source xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST table:table-source xlink:actuate (onRequest) "onRequest">`
	`<!ATTLIST table:table-source xlink:href %url; #REQUIRED>`

Filter Name

The `table:filter-name` attribute specifies the file type of the document containing the original table.

XML Code:	`table:filter-name`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:table-source table:filter-name CDATA #IMPLIED>`

Filter Options

The `table:filter-options` attribute specifies optional settings about the file type.

XML Code:	`table:filter-options`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:table-source table:filter-options CDATA #IMPLIED>`

Table Name

The `table:table-name` attribute specifies the name of the table in the original table document. If the table name is not specified, the application uses the first table in the document.

XML Code:	`table:table-name`
Rules:	
DTD:	`<!ATTLIST table:table-source table:table-name CDATA #IMPLIED>`

Refresh Delay

The `table:refresh-delay` attribute specifies the time delay between refresh actions.

XML Code:	`table:refresh-delay`
Rules:	This attribute can be associated with the `<table:database-range>` element.
DTD:	`<!ATTLIST table:table-source table:refresh-delay % timeDuration; #IMPLIED >`

4.5.3 Scenario Table

The `<table:scenario>` element represents a scenario table. The name of the table and the name of the scenario are the same. The scenario is displayed in the regular table preceeding the scenario table. Only one scenario table can be active at one time.

XML Code:	`<table:scenario>`
Rules:	
DTD:	`<!ELEMENT table:scenario EMPTY>`

The attributes that you can associate with this element are:

- Display Border
- Border Color
- Copy Back
- Copy Styles
- Copy Formulas
- Is Active

- Scenario Ranges
- Comment

Display Border

The `table:display-border` attribute specifies whether or not to display the border of the scenario.

XML Code:	`table:display-border`
Rules:	
DTD:	`<!ATTLIST table:scenario table:display-border %boolean; "true">`

Border Color

The `table:border-color` attribute specifies the color of the border.

XML Code:	`table:border-color`
Rules:	
DTD:	`<!ATTLIST table:scenario table:border-color %color; #IMPLIED>`

Copy Back

The `table:copy-back` attribute specifies whether or not data is copied back into the active scenario table if another scenario is activated.

XML Code:	`table:copy-back`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, you can directly edit the data for each scenario in the scenario table.
DTD:	`<!ATTLIST table:scenario table:copy-back %boolean; "true">`

Copy Styles

The `table:copy-styles` attribute specifies whether or not to copy the cell styles with the data.

XML Code:	`table:copy-styles`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST table:scenario table:copy-styles %boolean; "true">`

Copy Formulas

The `table:copy-formulas` attribute specifies whether or not to copy the formulas.

XML Code:	`table:copy-formulas`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the formulas are copied. If the value is `false`, only the values resulting from the formulas are copied.
DTD:	`<!ATTLIST table:scenario table:copy-formulas %boolean; "true">`

Is Active

The `table:is-active` attribute specifies whether or not the current scenario is active.

XML Code:	`table:is-active`
Rules:	
DTD:	`<!ATTLIST table:scenario table:is-active %boolean; #REQUIRED>`

Scenario Ranges

The `table:scenario-ranges` attribute specifies the range of this scenario.

XML Code:	`table:scenario-ranges`
Rules:	The value of this attribute is a list of cell range addresses.
DTD:	`<!ATTLIST table:scenario table:scenario-ranges %cell-range-address-list; #REQUIRED>`

Comment

The `table:comment` attribute contains a comment about the scenario.

XML Code:	`table:comment`
Rules:	
DTD:	`<!ATTLIST table:scenario table:comment CDATA #IMPLIED>`

4.5.4 Shapes

This element contains all shapes with an anchor on a table. This is a container element and does not have any associated attributes.

XML Code:	`<table:shapes>`
Rules:	
DTD:	`<!ELEMENT table:shapes ANY>`

4.6 Columns

4.6.1 Grouping

Columns can be grouped. Every group can contain a new group, columns, and column headers. Every group can be visible or hidden.

The <table:table-column-headers> should only be separated by <table:table-column-group> elements, so that if the table:table-column-group does not exist there is only one <table:table-header-columns> element.

XML Code:	<table:table-column-group>
Rules:	There can only be one <table:table-header-columns> element in this element.
DTD:	<!ELEMENT table:table-column-group "(table:table-header-columns | table:table-column | table:table-column-group)+" >

The attributes that you can associate with this element are:

- Display

Display

This attribute specifies whether or not the group is visible.

XML Code:	table:display
Rules:	
DTD:	<!ATTLIST table:table-column-group table:display %boolean; "true">

4.6.2 Column Groups

There are two types of column groups, as follows:

- **Header groups**

  A header group is a group of columns that repeat on each page if the table extends over several pages.

- **Body groups**

  A body group is a group of columns that do not repeat across pages. Typically, a body group contains the content of the table that is not part of the header.

XML Code:	`<table:table-header-columns>` `<table:table-columns>`
Rules:	The table header column element represents a header column. The table column element represents a body column. You can omit the `<table:table-columns>` element, in the same way that you can omit the`<TBODY>` tag in HTML. A table must contain at least one column group, but only one header group. A body group must not follow another body group.
DTD:	`<!ELEMENT table:table-header-columns ( table:table-column | table:table-column-group )+>` `<!ELEMENT table:table-columns ( table:table-column | table:table-column-group )+>`
Notes:	Applications may support column header groups, but this is not essential. If a user agent does not support header groups, it must process header groups as body groups. There are no column groups in XSL.

4.6.3 Column Description

Every column in a table has a column description element. If two or more columns are adjoining and have the same properties, you can describe them using a single `<table:table-column>` element.

XML Code:	`<table:table-column>`
Rules:	
DTD:	`<!ELEMENT table:table-column EMPTY>`
Notes:	The `<table:table-column>` element is similar to the XSL `<fo:table-column>` element.

Number of Columns Repeated

The `table:number-columns-repeated` attribute specifies the number of columns to which a column description applies.

XML Code:	`table:number-columns-repeated`
Rules:	If two or more columns are adjoining, and have the same properties, you can use a single `<table:table-column>` element to describe them. In this case, you use a `table:number-columns-repeated` attribute to specify the number of successive columns to which the description applies. You specify this attribute with the `<table:table-column>` element.
DTD:	`<!ATTLIST table:table-column` `table:number-columns-repeated %number; "1">`

Column Style

A table style stores the formatting properties of a table column, such as width and background color. The table style can be either an automatic or a common style. You specify the style of a column using a table style.

XML Code:	`table:style-name`
Rules:	To define the style of the column, you use a `<style:style>` element and a family attribute value of `table`. The `<table:table-column>` element includes a `table:style-name` attribute that references the style by the `style:name` attribute.
DTD:	`<!ATTLIST table:style-name %style-name; #REQUIRED;>`

Visibility

The `table:visibility` attribute specifies whether the column is visible, filtered, or collapsed.

XML Code:	`table:visibility`
Rules:	This attribute is associated with the `<table:table-column>` element. If the value of this attribute is `filter`, the column is also collapsed.
DTD:	`<!ATTLIST table:table-column table:visibility ( visible | collapse | filter ) "visible">`

Default Cell Style

The `table:default-cell-style-name` attribute specifies the default cell style. Cells without a style use this style when there is no default cell style specified for the current column.

XML Code:	`table:default-cell-style-name`
Rules:	This attribute is associated with the `<table:table-column>` element.
DTD:	`<!ATTLIST table:table-column table:default-cell-style-name % style-name; #IMPLIED>`

Example: Table with three columns

This example shows the OpenOffice.org XML code for a table with three columns.

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm"
   fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  ...
</table:table>
```

4.7 Rows

4.7.1 Grouping

Rows can be grouped. Every group can contain a new group, rows, and row-headers. Every group can be visible or hidden.

The `table:table-row-headers` element can only be separated by `table:table-row-group` elements, so that if `table:table-row-group` does not exist there is only one `table:table-header-rows` element.

XML Code:	`<table:table-row-group>`
Rules:	There can only be one `table:table-header-rows` element in this element.
DTD:	`<!ELEMENT table:table-row-group "( table:table-header-rows | table:table-row | table:table-row-group)+" >`
Notes:	

The attributes that you can associate with this element are:

- Display

Display

This attribute specifies whether or not the group is visible.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST table:table-row-group table:display %boolean; "true">`

4.7.2 Row Groups

There are two types of row groups, as follows:

- **Header group**

  A header group is a group of rows that repeat on each page if the table extends over several pages.

- **Body group**

  A body group is a group of rows that do not repeat across pages. Typically, a body group contains the content of the table that is not part of the header.

XML Code:	`<table:table-header-rows>` `<table:table-rows>`
Rules:	The table header rows element represents a header row. The table row element represents a body row. You can omit the `<table:table-rows>` element, in the same way that you can omit the`<TBODY>` tag in HTML. A table must contain at least one row group, but only one header group. A body group must not follow another body group.
DTD:	`<!ELEMENT table:table-header-rows ( table:table-row | table:table-row-group )+>` `<!ELEMENT table:table-rows ( table:table-row | table:table-row-group )+>`
Notes:	Applications may support row header groups, but this is not essential. If a user agent does not support header groups, it must process header groups as body groups.

4.7.3 Row

The `<table:table-row>` element contsins other elements that specify the content of the table row.

XML Code:	`<table:table-row>`
Rules:	

DTD:	`<!ENTITY % table-cell`
	`"(table:table-cell|table:covered-table-cell)">`
	
	`<!ELEMENT table:table-row %table-cell;+>`

Notes:	The `<table:table-row>` element is similar to the XSL `<fo:table-row>` element.

Number of Rows Repeated

The `table:number-rows-repeated` attribute specifies the number of rows to which a row element applies. If two or more rows are adjoining, and have the same content and properties, you can use a single`<table:table-row>` element to describe them.

XML Code:	`table:number-rows-repeated`
Rules:	You specify this attribute with the `<table:table-row>` element.
DTD:	`<!ATTLIST table:table-row` `table:number-rows-repeated %number; "1">`
Notes:	You can not repeat a row that contains a vertically merged cell.

Row Style

A table style stores the formatting properties of a table row, such as height and background color. The table style can be either an automatic or a common style. You specify the style of a row using a table style.

XML Code:	`table:style-name`
Rules:	To define the style of the row, you use a `<style:style>` element and a family attribute value of `table`. The `<table:table-row>` element includes a `table:style-name` attribute that references the style by the `style:name` attribute.
DTD:	`<!ATTLIST table:table-row table:style-name %style-name; #IMPLIED;>`
Notes:	The table row style attribute is similar to the XSL `<fo:table-row>` element.

Visibility

The `table:visibility` attribute specifies whether the row is visible, filtered, or collapsed.

XML Code:	`table:visibility`
Rules:	This attribute is associated with the `<table:table-row>` element. If the value of this attribute is `filter`, the row also collapsed.
DTD:	`<!ATTLIST table:table-row table:visibility ( visible | collapse | filter ) "visible">`

Default Cell Style

The `table:default-cell-style-name` attribute specifies the default cell style. Cells without an individual style use the default style.

XML Code:	`table:default-cell-style-name`
Rules:	This attribute is associated with the `<table:table-row>` element.
DTD:	`<!ATTLIST table:table-row table:default-cell-style-name % style-name; #IMPLIED>`

Example: Table with three rows and three columns

This example shows the OpenOffice.org XML code for a table with three rows and three columns. The first two rows of the table have a blue background.

```
<style:style style:name="Table 1" style:family="table">

  <style:properties fo:width="12cm"
   fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
  <style:properties fo:background-color="blue"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  <table:table-rows>
    <table:table-row table:style-name="Row1">
      ...
    </table:table-row>
    <table:table-row table:style-name="Row1">
      ...
    </table:table-row>
    <table:table-row>
      ...
    </table:table-row>
  <table:table-rows>
</table:table>
```

4.8 Cells

4.8.1 Table Cell

The `<table:table-cell>` element specifies the content of a table cell. The `<table:table-cell>` elements are contained in table row elements.

XML Code:	`<table:table-cell>` `<table:covered-table-cell>`
Rules:	The `<table:table-cell>` element can contain: • Paragraphs and other text content, or • A subtable When table cells merge horizontally and vertically, the `<table:covered-table-cell>` element represents cells that are covered by other cells. The `<table:table-cell>` element represents all other cells.

DTD:	`<!ENTITY % cell-content "(table:cell-range-source?, office:annotation?,(table:subtable|%text-wo-table;))">`
	`<!ELEMENT table:table-cell %cell-content;>`
	`<!ELEMENT table:covered-table-cell %cell-content;>`

Notes:	There is a difference between the representation of cells that span several rows or columns in OpenOffice.org XML, and their representation in HTML and XSL. When a cell merges with other cells, the cells that the first cell covers do not appear in the HTML or XSL representation of the table. OpenOffice.org XML represents these covered cells as `<table:covered-table-cell>` elements. The reasons for this are as follows: • In spreadsheets, there can be some content in the covered cells. • If a row does not include covered cells, it is very difficult to identify the column that contains a particular cell. It means that you must process all preceding cells to identify the column. Identifying the column that contains a particular cell is very important for transformations to other XML languages, because this information is essential to calculate the width of a cell. Apart from this, the table cell element is similar to the XSL `<fo:table-cell>` element and the HTML `<td>` and `<th>` tags.

Number of Cells Repeated

The `table:number-columns-repeated` attribute specifies the number of successive columns in which a cell is repeated.

XML Code:	`table:number-columns-repeated`
Rules:	You can use a single `<table:table-cell>` element to describe two or more adjoining cells, if they meet the following conditions: • The cells contain the same content and properties. • The cells are not merged horizontally or vertically. In this case, you use a `table:number-columns-repeated` attribute to specify the number of successive columns in which the cell is repeated. You specify this attribute with either the `<table:table-cell>` element or the `<table:covered-table-cell>` element.

DTD:	`<!ATTLIST table:table-cell`
	`table:number-columns-repeated %number; "1">`
	`<!ATTLIST table:covered-table-cell`
	`table:number-columns-repeated %number; "1">`

Number of Rows and Columns Spanned

These attributes specify the number of rows and columns that a cell spans. You specify these attributes with the `<table:table-cell>` element.

XML Code:	`table:number-rows-spanned` `table:number-columns-spanned`
Rules:	

DTD:	`<!ATTLIST table:table-cell`
	`table:number-rows-spanned %number; "1">`
	
	`<!ATTLIST table:table-cell`
	`table:number-columns-spanned %number; "1">`

Notes:	When a cell covers another cell, a `<table:covered-table-cell>` element must appear in the table to represent the covered cell.

Cell Style

A table style stores the formatting properties of a cell, such as the following:

- Background color
- Number format
- Vertical alignment
- Borders

The table style can be either an automatic or a common style. You specify the style of a cell using a table style. If a cell does not have a style, the application checks if a the current row has a default cell style. If the current row does not have a default cell style, the application checks if the current column has a default cell style.

XML Code:	`table:style-name`
Rules:	The value of this attribute must be the name of a `<style:style>` element that belongs to the `table-cell` style family.

DTD:	`<!ATTLIST table:table-cell table:style-name %style-name; #IMPLIED>`
	`<!ATTLIST table:covered-table-cell table:style-name %style-name;`
	`#IMPLIED>`

Cell Content Validation

The `table:content-validation-name` attribute specifies if a cell contains a validity check.

XML Code:	`table:content-validation-name`
Rules:	The value of this attribute is the name of a `<table:cell-content-validation>`element.

DTD:	`<!ATTLIST table:table-cell table:content-validation-name CDATA`
	`#IMPLIED>`
	`<!ATTLIST table:covered-table-cell table:content-validation-name`
	`CDATA #IMPLIED>`

See Section 4.9 for more information on cell content validation and the `<table:cell-content-validation>` element.

Formula

Formulas allow you to perform calculations within table cells. Every formula begins with an equal (=) sign. Formulas can include the following components:

- Numbers.
- Text.
- Named ranges.
- Operators.
- Logical operators.
- Function calls.
- Addresses of cells that contain numbers. The addresses can be relative or absolute, see Section 4.8.7. Addresses in formulas start with a "[" and end with a "]". See Sections 4.8.7and 4.8.8 for information about how to address a cell or cell range.

The following is an example of a simple formula:

```
=sum([.A1:.A5])
```

This formula calculates the sum of the values of all cells in the range "`.A1:.A5`". The function is "`sum`". The parameters are marked by a "`(`" at the start and a "`)`" at the end. If a function contains more than one parameter, the parameters are separated by a "`;`".

The following is a variation of the formula shown above:

```
=sum([.A1];[.A2];[.A3];[.A4];[.A5])
```

The result of this formula is the same. The components that you use in the formula depend on the application that you are using.

The `table:formula` attribute contains a formula for a table cell.

XML Code:	`table:formula`
Rules:	See above.
DTD:	`<!ATTLIST table:table-cell table:formula CDATA #IMPLIED>` `<!ATTLIST table:covered-table-cell table:formula CDATA #IMPLIED>`
Notes:	One of the following attributes represents the current value of the cell: • `table:value` • `table:date-value` • `table:time-value` • `table:boolean-value` • `table:string-value`

Matrix

When an application is performing spreadsheet calculations, a connected range of cells that contains values is called a matrix. If the cell range contains *m* rows and *n* columns, the matrix is called an *m x n* matrix. The smallest possible matrix is a *1 x 2* or *2 x 1* matrix with two adjacent cells. If you want to use a matrix in a formula, you must include the cell range address of the matrix in the formula. In a matrix formula, only special matrix operations are possible.

The number of rows and columns that a matrix spans are represented by the `table:number-matrix-rows-`

spanned and table:number-matrix-columns-spanned attributes, which are attached to the cell elements.

XML Code:	table:number-matrix-rows-spanned table:number-matrix-columns-spanned
Rules:	These attributes are attached to the cell element in the first row and the first column of the matrix.
DTD:	<!ATTLIST table:covered-table-cell table:number-matrix-rows-spanned %number; #IMPLIED> <!ATTLIST table:table-cell table:number-matrix-rows-spanned %number; #IMPLIED> <!ATTLIST table:covered-table-cell table:number-matrix-columns-spanned %number; #IMPLIED> <!ATTLIST table:table-cell table:number-matrix-columns-spanned %number; #IMPLIED>

Value Type

The table:value-type attribute specifies the type of value that can appear in a cell.

XML Code:	table:value-type
Rules:	The table:cell-type may contain one of the following values: • float • time • date • percentage • currency • boolean • string
DTD:	<!ATTLIST table:table-cell table:value-type ("float"|"time"|"date"|"percentage"|"currency"|"boolean"| "string") "string"> <!ATTLIST table:covered-table-cell table:value-type ("float"|"time"|"date"|"percentage"|"currency"|"boolean"| "string") "string">

Cell Current Numeric Value

The table:value attribute specifies the current numeric value of a cell.

XML Code:	table:value
Rules:	This attribute is only evaluated for cells that contain the following data types: • float • percentage • currency
DTD:	<!ENTITY % float CDATA> <!ATTLIST table:table-cell table:value %float; #IMPLIED> <!ATTLIST table:covered-table-cell table:value %float; #IMPLIED>

Cell Current Date Value

The `table:date-value` attribute specifies the current date value of a cell.

XML Code:	`table:date-value`
Rules:	This attribute is only evaluated for cells whose data type is `date`.
DTD:	`<!ATTLIST table:table-cell table:date-value %date; #IMPLIED>` `<!ATTLIST table:covered-table-cell table:date-value %date; #IMPLIED>`

Cell Current Time Value

The `table:time-value` attribute specifies the current time value of a cell.

XML Code:	`table:time-value`
Rules:	This attribute is only evaluated for cells whose data type is `time`.
DTD:	`<!ATTLIST table:table-cell table:time-value %time; #IMPLIED>` `<!ATTLIST table:covered-table-cell table:time-value %time; #IMPLIED>`

Cell Current Boolean Value

The `table:boolean-value` attribute specifies the current Boolean value of a cell.

XML Code:	`table:boolean-value`
Rules:	This attribute is only evaluated for cells whose data type is `boolean`.
DTD:	`<!ATTLIST table:table-cell table:boolean-value %boolean; #IMPLIED>` `<!ATTLIST table:covered-table-cell table:boolean-value %boolean; #IMPLIED>`

Cell Current String Value

The `table:string-value` attribute specifies the current string value of a cell.

XML Code:	`table:string-value`
Rules:	This attribute is only evaluated for cells whose data type is `string`.
DTD:	`<!ATTLIST table:table-cell table:string-value CDATA #IMPLIED>` `<!ATTLIST table:covered-table-cell table:string-value CDATA #IMPLIED>`
Notes:	If the `table:string-value` attribute does not exist, the value of the cell is the text content of the cell.

Cell Current Currency Value

The `table:currency` attribute specifies the current currency value of a cell. The value of this attribute is usually currency information such as DEM or EUR.

XML Code:	`table:currency`
Rules:	This attribute is only evaluated for cells whose data type is `currency`.
DTD:	`<!ATTLIST table:table-cell table:currency CDATA #IMPLIED>` `<!ATTLIST table:covered-table-cell table:currency CDATA #IMPLIED>`

Table Cell Protection (OpenOffice.org Writer only)

The `table:protected` attribute protects the table cells contained in OpenOffice.org Writer documents. Users can not edit the content of a cell that is marked as protected.

XML Code:	`table:protected`
Rules:	This attribute should only be used in OpenOffice.org Writer tables.
DTD:	`<!ATTLIST table:table-cell table:protected %boolean; "false">`

OpenOffice.org Calc supports more sophisticated protection functionality for spreadsheet documents, as described in Section 4.21.16.

4.8.2 Cell Range Source

If a cell is linked to a database range or named range of another file, the original database range or named range is represented by a `<table:cell-range-source>` element.

XML Code:	`<table:cell-range-source>`
Rules:	
DTD:	`<!ELEMENT table:cell-range-source EMPTY>`

The attributes that you can associate with this element are:

- Name
- URL
- Filter name
- Filter options
- Last size
- Refresh delay

Name

The `table:name` attribute specifies the name of the database range or named range.

XML Code:	`table:name`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST table:cell-range-source table:name CDATA #REQUIRED>`

URL

The XLink attributes specify the URL of the linked table document.

XML Code:	`xlink:type`, `xlink:actuate`, and `xlink:href`
Rules:	

DTD:	`<!ATTLIST table:cell-range-source xlink:type (simple) #FIXED "simple">`
	`<!ATTLIST table:cell-range-source xlink:actuate (onRequest) #FIXED "onRequest">`
	`<!ATTLIST table:cell-range-source xlink:href %url; #REQUIRED>`

Filter Name

The `table:filter-name` attribute specifies the file type of the linked table document.

XML Code:	`table:filter-name`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:cell-range-source table:filter-name CDATA #REQUIRED>`

Filter Options

The `table:filter-options` attribute specifies optional settings about the file type.

XML Code:	`table:filter-options`
Rules:	The value of this attribute is application-specific.
DTD:	`<!ATTLIST table:cell-range-source table:filter-options CDATA #IMPLIED>`

Last Size

The `table:last-column-spanned` and `table:last-row-spanned` attributes specify the last known size of the range. If the size of the range is changed since the last operation, the values of these attributes are incorrect.

XML Code:	`table:last-column-spanned` `table:last-row-spanned`
Rules:	These attributes are mandatory.
DTD:	`<!ATTLIST table:cell-range-source table:last-column-spanned % positiveInteger; #REQUIRED>` `<!ATTLIST table:cell-range-source table:last-row-spanned % positiveInteger; #REQUIRED>`

Refresh Delay

The `table:refresh-delay` attribute specifies the time delay between refresh actions.

XML Code:	`table:refresh-delay`
Rules:	
DTD:	`<!ATTLIST table:cell-range-source table:refresh-delay % timeDuration; #IMPLIED>`

4.8.3 Annotation

The `<office:annotation>` element specifies an OpenOffice.org annotation.

XML Code:	`<office:annotation>`
Rules:	
DTD:	`<!ELEMENT office:annotation (text:p)*>`

The attributes associated with the `<office:annotation>` element are:

- Author
- Creation date
- Display

Author

The `office:author` attribute specifies the author of the annotation.

XML Code:	`office:author`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST office:author CDATA; #IMPLIED>`

Creation Date

The `office:create-date` attribute specifies the creation date and time of the annotation. If the application only has a date string and cannot parse this string, it must write the string to the `office:create-date-string` attribute.

XML Code:	`office:create-date`
Rules:	This attribute is mandatory.
DTD:	`<!ATTLIST office:create-date %date-time; #IMPLIED>`
	`<!ATTLIST office:create-date-string %string; #IMPLIED>`

Display

The `office:display` attribute specifies whether or not the annotation is visible.

XML Code:	`office:display`
Rules:	This attribute is optional and can have a `true` or `false` value.
DTD:	`<!ATTLIST office:display %boolean; "false">`

4.8.4 Detective

The `<table:detective>` element allows you to display links between the current formula cells and the cells in the spreadsheet document. This makes is easy to check formula definitions and make any necessary corrections.

XML Code:	`<table:detective>`
Rules:	
DTD:	`<!ELEMENT table:detective (table:highlighted-range*, table:operation*)>`

The elements that can be contained in the `<table:detective>` element are:

- Highlighted range
- Operation

4.8.5 Highlighted Range

The `<table:highlighted-range>` element specifies a range cell address for a range which contains dependents or precendents and whether or not the range contains an error. Also it can specify that the containing cell is marked invalid. The `table:marked-invalid` attribute should be always alone and should not be there if another attribute is there.

It represents the visible marks created by detective. It is not given, that the representation is really true. So it is possible, that the cell is not invalid, but at the time the operation was done the cell was invalid. So it is also possible, that the dependents/precendents are no longer in the given ranges.

XML Code:	`<table:highlighted-range>`
Rules:	
DTD:	`<!ELEMENT table:highlighted-range EMPTY>`

The attributes that you can associate with the `<table:highlighted-range>` element are:

- Cell Range Address
- Direction
- Contains Error
- Marked Invalid

Cell Range Address

The `table:cell-range-address` attribute specifies the cell range address of the highlighted range.

XML Code:	`table:cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:highlighted-range table:cell-range-address %cell-range-address; #IMPLIED>`

Direction

The `table:direction` attribute specifies the direction of the arrow between this cell and the highlighted range.

XML Code:	`table:direction`
Rules:	
DTD:	`<!ATTLIST table:highlighted-range table:direction (from-another-table | to-another-table | from-same-table) #IMPLIED>`

Contains Error

The `table:contains-error` attribute specifies whether or not the cell range contains an error.

XML Code:	`table:contains-error`
Rules:	
DTD:	`<!ATTLIST table:highlighted-range table:contains-error %boolean; #IMPLIED>`

Marked Invalid

The `table:marked-invalid` attribute specifies whether or not the cell is marked invalid.

XML Code:	`table:marked-invalid`
Rules:	
DTD:	`<!ATTLIST table:highlighted-range table:marked-invalid %boolean; #IMPLIED>`

4.8.6 Operation

The `<table:operation>` element contains the operation and its index. Using the index, the application can restore the order of the operations of all cells.

XML Code:	`<table:operation>`
Rules:	
DTD:	`<!ELEMENT table:operation EMPTY>`

The attributes associated with the `<table:operation>` element are:

- Name
- Index

Name

The `table:name` attribute specifies the name of the operation. Possible names are `trace-dependents`, `remove-dependents`, `trace-precedents`, `remove-precedents`, and `trace-errors`.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:operation table:name (trace-dependents|remove-dependents|trace-precedents|remove-precedents|trace-errors) #REQUIRED>`

Index

The `table:index` attribute specifies the index in the order of the operations of all cells.

XML Code:	`table:index`
Rules:	
DTD:	`<!ATTLIST table:operation table:index %nonNegativeInteger; #REQUIRED>`

4.8.7 Cell Address Entity

A special data type exists for the address of a cell. A cell address entity describes a cell address.

The structure of a cell address is as follows:

1. The name of the table.
2. A dot (.).
3. An alphabetic value representing the column. The letter A represents column 1, B represents column 2, and so on. AA represents column 27, AB represents column 28, and so on.
4. A numeric value representing the row. The number 1 represents the first row, the number 2 represents the second row, and so on.

   A1 represents the cell in column 1 and row 1. B1 represents the cell in column 2 and row 1. A2 represents the cell in column 1 and row 2.

For example, if you have a table with the name `SampleTable` and you want to address the cell in column 34 and row 16, the address is `SampleTable.AH16`. In some cases it is not necessary to provide the name of the table. However, the dot must be present. When the table name is not required, the address in the previous example is `.AH16`.

The structure of the address of a cell in a subtable is as follows:

1. The address of the cell that contains the subtable.
2. A dot (.).
3. The address of the cell in the subtable.

For example, to reference the cell in column 1 and row 1 in a subtable that is called `Subtable`, and that is in column 34 and row 16 of the table `SampleTable`, the address is `SampleTable.AH16.A1`. If the name of the table contains a blank the name should be between two quotation marks. This quotation marks should be

quoted.

Absolute and relative cell addressing

You can reference cells in two ways, using absolute addresses or relative addresses. When you perform an operation on a table cell, for example when you copy a formula, absolute cell references do not change; relative cell references do change. The previous example uses relative addressing.

To create an absolute address, place a dollar sign ($) before each table name, column reference, and row reference. For example, the absolute address of the previous example is `$SampleTable.$AH$16`. You can combine absolute and relative references in a cell address. For example, you can use `SampleTable.AH$16` to refer to a relative table and column, and an absolute row. Absolute addresses must contain a table name. The discrimination between absolute and relative addressing is only necessary in some special cases. Otherwise, a cell reference without the dollar signs is used.

XML Code:	`cell-address`
Rules:	
DTD:	`<!ENTITY % cell-address CDATA>`

4.8.8 Cell Range Address Entity

A cell range is a number of adjacent cells forming a rectangular shape. The rectangle stretches from the cell on the top left to the cell on the bottom right.

The range address of cells has a special data type. To specify a cell range address, you use an entity. To reference a range of adjacent cells, construct the address as follows, in the specified order:

1. The address of the cell at the top left of the range you want to reference.
2. A colon (:).
3. The address of the cell at the bottom right of the range you want to reference.

For example, you use the address `.A1:.B2` to reference the range of cells from column 1 and row 1 to column 2 and row 2. The smallest range you can specify is a single cell. In this case, the range address is the same as the cell address.

XML Code:	`cell-range-address`
Rules:	
DTD:	`<!ENTITY % cell-range-address CDATA>`

4.8.9 Cell Range Address List Entity

A cell range address list is a list of cell ranges or cell addresses , or both. Each item in the list is separated by a space.

XML Code:	`cell-range-address-list`
Rules:	
DTD:	`<!ENTITY % cell-range-address-list CDATA>`

4.9 Table Cell Content Validations

The <table:content-validations> element contains all of the cell content validations.

XML Code:	<table:content-validations>
Rules:	
DTD:	<!ELEMENT table:content-validations (table:content-validation)*>

4.9.1 Table Cell Content Validation

The <table:content-validation> element specifies the validation of the content of the cell with this style.

XML Code:	<table:content-validation>
Rules:	If this element is not present, the cell can contain any content.
DTD:	<!ELEMENT table:content-validation (table:help-message?, (table:error-message | table:error-macro)?)>

The attributes that you can associate with the <table:content-validation> element are:

- Name
- Condition
- Base cell address
- Allow empty cell

Name

The table:name attribute specifies the name of the content validation. The name is created automatically by the application.

XML Code:	table:name
Rules:	
DTD:	<!ATTLIST table:content-validation table:name CDATA #REQUIRED>

Condition

The table:condition attribute specifies the condition which cell content is allowed.

XML Code:	table:condition

Rules:	The value of this attribute is a Boolean expression. The syntax of the expression is similar to the XPath syntax. The following are valid conditions: • `Condition ::= ExtendedTrueCondition | TrueFunction 'and' TrueCondition` • `TrueFunction ::= cell-content-is-whole-number() | cell-content-is-decimal-number() | cell-content-is-date() | cell-content-is-time()` • `ExtendedTrueCondition ::= ExtendedGetFunction | cell-content-text-length() Operator Value` • `TrueCondition ::= GetFunction | cell-content() Operator Value` • `GetFunction ::= cell-content-is-between(Value, Value) | cell-content-is-not-between(Value, Value)` • `ExtendedGetFunction ::= cell-content-text-length-is-between (Value, Value) | cell-content-text-length-is-not-between (Value, Value)` • `Operator ::= '<' | '>' | '<=' | '>=' | '=' | '!='` • `Value ::= NumberValue | String | Formula` A `Formula` is a formula without an equals (=) sign at the beginning. See Section 4.8.1 for more information. A `String` comprises one or more characters surrounded by quotation marks. A `NumberValue` is a whole or decimal number. You must include an `Operator`. The number in a `NumberValue` or `Formula` cannot contain comma separators for numbers of 1000 or greater.
DTD:	`<!ATTLIST table:content-validation table:condition CDATA #IMPLIED>`

Base Cell Address

The `table:base-cell-address` attribute specifies the address of the base cell for relative addresses in formulas.

XML Code:	`table:base-cell-address`
Rules:	This attribute is only necessary when the condition contains a formula. The value of this attribute must be an absolute cell address with a table name.
DTD:	`<!ATTLIST table:content-validation table:base-cell-address % cell-address; #IMPLIED>`

Allow Empty Cell

The `table:allow-empty-cell` attribute specifies whether or not a cell can be empty.

XML Code:	table:allow-empty-cell
Rules:	
DTD:	<!ATTLIST table:content-validation table:allow-empty-cell % boolean; #IMPLIED>

4.9.2 Help Message

The <table:help-message> element specifies a message to display if a user selects the cell.

XML Code:	<table:help-message>
Rules:	
DTD:	<!ELEMENT table:help-message (text:p*)>

The attributes that you can associate with the <table:help-message> element are:

- Title
- Display

Title

The table:title attribute specifies the title of the help message.

XML Code:	table:title
Rules:	
DTD:	<!ATTLIST table:help-message table:title CDATA #IMPLIED>

Display

The table:display attribute specifies whether or not to display the message.

XML Code:	table:display
Rules:	
DTD:	<!ATTLIST table:help-message table:display %boolean; #IMPLIED>

4.9.3 Error Message

The <table:error-message> element specifies a message to display if a user tries to include unacceptable content in a cell.

XML Code:	<table:error-message>
Rules:	
DTD:	<!ELEMENT table:error-message (text:p*)>

The attributes that you can associate with the <table:error-message> element are:

- Title

- Message Type
- Display

Title

The `table:title` attribute specifies the title of the error message.

XML Code:	`table:title`
Rules:	
DTD:	`<!ATTLIST table:error-message table:title CDATA #IMPLIED>`

Message Type

The `table:message-type` attribute specifies the type of error message.

XML Code:	`table:message-type`
Rules:	The value of this attribute can be `stop`, `warning`, or `information`.
DTD:	`<!ATTLIST table:error-message table:message-type ( stop | warning | information ) #IMPLIED>`

Display

The `table:display` attribute specifies whether or not to display the message.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST table:error-message table:display %boolean; #IMPLIED>`

4.9.4 Error Macro

The `<table:error-macro>` element specifies a macro that is executed when a user tries to included unacceptable content in a cell.

XML Code:	`<table:error-macro>`
Rules:	
DTD:	`<!ELEMENT table:error-macro EMPTY>`

The attributes that you can associate with the `<table:error-macro>` element are:

- Name
- Execute

Name

The `table:name` attribute specifies the name of the macro.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:error-macro table:name CDATA #IMPLIED>`

Execute

The `table:execute` attribute specifies whether or not to execute the macro.

XML Code:	`table:execute`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST table:error-macro table:execute %boolean; #IMPLIED>`

4.10 Subtables

A subtable is a table within another table. It occupies one cell and no other content can appear in this cell. If a table cell contains a subtable, it cannot contain any paragraphs.

XML Code:	`<table:sub-table>`
Rules:	The borders of a subtable merge with the borders of the cell that it resides in. A subtable does not contain any formatting properties. A subtable is essentially a container for some additional table rows that integrate seamlessly with the parent table.
DTD:	`<!ELEMENT table:sub-table (%table-column-groups;,%table-row-groups;)>`
Notes:	There is a difference between a subtable and a HTML table that is nested within another HTML table. A nested HTML table appears as a table within a table, that is, it has borders distinct from those of the parent cell and respects the padding of the parent cell.

Example of Representation of Subtable

OpenOffice.org **XML can represent this table in either of the ways detailed in Sample 1 and Sample 2.**

A1	**B1**	**C1**
A2	B2.1.1	B2.2.1
	B2.1.2	

Sample 1

OpenOffice.org XML can describe the preceding table as follows:

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm" fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
  <style:properties fo:background-color="grey"/>
</style:style>
<style:style style:name="Cell1" style:family="table-cell">
  <style:properties fo:background-color="grey"/>
</style:style>
<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  <table:table-header-rows>
    <table:table-row table:style-name="Row1">
      <table:table-cell>
        <text:p text:style="Table Caption">
          A1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          B1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          C1
        </text:p>
      </table:table-cell>
    </table:table-row>
  </table:table-header-rows>
```

```
  <table:table-rows>
    <table:table-row>
      <table:table-cell table:number-rows-spanned="2" table:style-name="Cell1">
        <text:p text:style="Table Body">
          A2
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Body">
          B2.1.1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Body">
          B2.2.1
        </text:p>
      </table:table-cell>
    </table:table-row>
    <table:table-row>
      <table:covered-table-cell/>
      <table:table-cell table:number-columns-spanned="2">
        <text:p text:style="Table Body">
          B2.1.2
        </text:p>
      </table:table-cell>
      <table:covered-table-cell/>
    </table:table-row>
  </table:table-rows>
</table:table>
```

Sample 2

This sample ignores the borders of the table. OpenOffice.org XML can describe the preceding table as follows:

```
<style:style style:name="Table 1" style:family="table">
  <style:properties fo:width="12cm" fo:background-color="light-grey"/>
</style:style>
<style:style style:name="Col1" style:family="table-column">
  <style:properties fo:width="2cm"/>
</style:style>
<style:style style:name="Col2" style:family="table-column">
  <style:properties fo:width="4cm"/>
</style:style>
<style:style style:name="Col3" style:family="table-column">
  <style:properties fo:width="6cm"/>
</style:style>
<style:style style:name="Row1" style:family="table-row">
  <style:properties fo:background-color="grey"/>
</style:style>
<style:style style:name="Cell1" style:family="table-cell">
  <style:properties fo:background-color="grey"/>
</style:style>

<table:table table:name="Table 1" table:style-name="Table 1">
  <table:table-columns>
    <table:table-column table:style-name="Col1"/>
    <table:table-column table:style-name="Col2"/>
    <table:table-column table:style-name="Col3"/>
  </table:table-columns>
  <table:table-header-rows>
    <table:table-row table:style-name="Row1">
      <table:table-cell>
        <text:p text:style="Table Caption">
          A1
        </text:p>
      </table:table.cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          B1
        </text:p>
      </table:table-cell>
      <table:table-cell>
        <text:p text:style="Table Caption">
          C1
        </text:p>
      </table:table-cell>
    </table:table-row>
  </table:table-header-rows>
  <table:table-rows>
    <table:table-row>
      <table:table-cell table:style-name="Cell1">
        <text:p text:style="Table Body">
          A2
        </text:p>
      </table:table-cell>
      <table:table-cell table:number-columns-spanned="2">
```

```
        <table:subtable>
          <table:table-columns>
            <table:table-column table:style-name="Col2"/>
            <table:table-column table:style-name="Col3"/>
          </table:table-columns>
              <table:table-cell>
                <text:p text:style="Table Body">
                  B2.1.1
                </text:p>
              </table:table-cell>
              <table:table-cell>
                <text:p text:style="Table Body">
                  B2.2.1
                </text:p>
              </table:table-cell>
            </table:table-row>
            <table:table-row>
              <table:table-cell table:number-columns-spanned="2">
                <text:p text:style="Table Body">
                  B2.1.2
                </text:p>
              </table:table-cell>
              <table:covered-table-cell/>
            </table:table-row>
          </table:table-rows>
        </table:subtable>
      </table:table-cell>
      <table:covered-table-cell/>
    </table:table-row>
  </table:table-rows>
</table:table>
```

4.11 Label Ranges

The label ranges element contains a collection of label ranges.

XML Code:	`<table:label-ranges>`
Rules:	
DTD:	`<!ELEMENT table:label-ranges (table:label-range)* >`

4.11.1 Label Range

The label range element specifies a cell range which contain the labels and a cell range, which contains the data. There are two types of label ranges.

- One for columns
- One for rows.

The range of the data should have either the same height and vertical position like the label range of rows or the same width and horizontal position like the label range of columns. For information on defining a cell range, see Section 4.8.4.

XML Code:	`<table:label-range>`
Rules:	
DTD:	`<!ELEMENT table:label-range EMPTY>`

The attributes that you can associate with the label range element are:

- Label cell range address
- Data cell range address
- Orientation

Label Cell Range Address

This attribute specifies the cell range address of the labels.

XML Code:	`table:label-cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:label-range table:label-cell-range-address % cell-range-address; #REQUIRED>`

Data Cell Range Address

This attribute specifies the cell range address of the data.

XML Code:	`table:data-cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:label-range table:data-cell-range-address % cell-range-address; #REQUIRED>`

Orientation

This attribute specifies the orientation of the label range.

XML Code:	`table:orientation`
Rules:	This attribute can have a value of `column` or `row`.
DTD:	`<!ATTLIST table:range table:orientation (column|row) #REQUIRED>`

4.12 Named Expressions

The named expressions element contains a collection of expressions with names, which you can use to refer to the expression.

XML Code:	`<table:named-expressions>`
Rules:	
DTD:	`<!ELEMENT table:named-expressions (table:named-range|table:named-expression)* >`

An expression element can be used to represent:

- A named cell range.
- Other expressions, for example, part of a formula.

4.12.1 Named Range

The named range element specifies a cell range with a name. For information on defining a cell range, see Section 4.8.4.

XML Code:	`<table:named-range>`
Rules:	If the cell range address is relative, the `table:base-cell-address` attribute must be attached to this element. If the named expression is a cell range a attribute that contains the cell area is necessary.
DTD:	`<!ELEMENT table:named-range EMPTY>`
Note:	A Named Range is one case where a discrimination between absolute and relative addresses is possible.

The attributes that you can associate with the named range element are:

- Name
- Cell range address
- Base cell address
- Range usable as

Name

XML Code:	`table:name`
Rules:	This attribute can be attached to the `<table:named-range>` and `<table:named-expression>` elements.
DTD:	`<!ATTLIST table:named-range table:name CDATA #REQUIRED>`

Cell Range Address

This attribute specifies the cell range address.

XML Code:	`table:cell-range-address`
Rules:	This attribute can be attached to the `<table:named-range>` element.
DTD:	`<!ATTLIST table:named-range table:cell-range-address %cell-range-address; #REQUIRED>`

Base Cell Address

This attribute specifies the base cell address if the cell address or the cell range address in the named range is relative.

XML Code:	`table:base-cell-address`
Rules:	This attribute can be attached to the `<table:named-range>` element.
DTD:	`<!ATTLIST table:named-range table:base-cell-address %cell-address; #IMPLIED>`
Note:	Discrimination between absolute and relative addressese is not possible. Therefore a table name in the address will needed and dollar signs will ignored.

Range usable as

This attribute specifies the possible usage of the named range. The named range can be used as a Print Range, a Filter, a Repeat Row, or a Repeat Column.

XML Code:	`table:range-usable-as`
Rules:	This attribute can be attached to the `<table:named-range>` element. The value of this attribute can be either: • `none` or • a space-separated list that consists of any of the values `print-range`, `filter`, `repeat-row` or `repeat-column`.
DTD:	`<!ATTLIST table:named-range table:range-usable-as CDATA "none">`

4.12.2 Named Expression

The named expression element contains an expression with a name, for example, part of a formula.

XML Code:	`<table:named-expression>`
Rules:	Expressions do not support the equal (=) sign as the first character. If the element contains a named range or another named expression, the named range or named expression must be specified first, before the containing expression.
DTD:	`<!ELEMENT table:named-expression EMPTY>`

The attributes that you can associate with the named expression element are:

- Name (the usage of this attribute is the same as for the `<table:named-range>` element, see Section

4.12.1.)

- Expression
- Base cell address (the usage of this attribute is the same as for the <table:named-range> element, see Section 4.12.1.)

Expression

XML Code:	table:expression
Rules:	This attribute can be attached to the <table:named-expression> element.
DTD:	<!ATTLIST table:named-expression table:expression CDATA #REQUIRED>

Example: Named expressions element with a named range and a named expression

```
<table:named-expressions>
  <table:named-range table:name="sample1" table:cell-range-address=".C4"
table:base-cell-address="sampletable.F1" table:area-type="none"/>
  <table:named-range table:name="sample2" table:cell-range-address=".$D$3:.
$K$8" table:area-type="print-range filter"/>
  <table:named-expression table:name="sample3" table:expression="sum([.A1:.
B3])"/>
</table:named-expressions>
```

4.13 Filters

4.13.1 Table Filter

The table filter element describes how to filter the data in a database range or datapilot tables.

XML Code:	<table:filter>
Rules:	
DTD:	<!ELEMENT table:filter (table:filter-condition | table:filter-and | table:filter-or) >

Target Range Address

This attribute specifies where the result of the filter is output.

XML Code:	table:target-range-address
Rules:	This attribute can be attached to the <table:filter> and <table:sort> elements. If the attribute is not present, the data is output in the source range.
DTD:	<!ATTLIST table:filter table:target-range-address %cell-range-address; #IMPLIED >
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

Condition Source Range Address

This attribute specifies the cell range from which the filter condition can get its data.

XML Code:	`table:condition-source-range-address`
Rules:	This attribute can be attached to the `<table:filter>` element.
DTD:	`<!ATTLIST table:filter table:condition-source-range-address % cell-range-address; #IMPLIED>`

Condition Source

This attribute specifies the source location from where the application gets the filter condition.

XML Code:	`table:condition-source`
Rules:	This attribute can have one of the following values: • `self` The application gets the filter condition from the `<table:filter>` element. • `cell-range` The application gets the filter condition from the cell range specified by the `table:condition-source-range-address` attribute.
DTD:	`<!ATTLIST table:filter table:condition-source ("self" | "cell-range") "self")`

Display Duplicates

A display duplicates attribute specifies whether or not to display duplicate matches in the result.

XML Code:	`table:display-duplicates`
Rules:	
DTD:	`<!ATTLIST table:filter table:display-duplicates %boolean; "true">`

4.13.2 Filter And

The `filter-and` element specifies whether the logical operator AND is used in a filter.

XML Code:	`<table:filter-and>`
Rules:	
DTD:	`<!ELEMENT table:filter-and ( table:filter-or | table:filter-condition )+ >`

4.13.3 Filter Or

The `filter-or` element specifies whether the logical operator OR is used in a filter.

XML Code:	`<table:filter-or>`
Rules:	
DTD:	`<!ELEMENT table:filter-or ( table:filter-and | table:filter-condition )+ >`

4.13.4 Filter Condition

The table filter condition element describes a condition to apply in a filter operation.

XML Code:	`<table:filter-condition>`
Rules:	
DTD:	`<!ELEMENT table:filter-condition EMPTY >`

Field Number

A field number attribute specifies which field to use for the condition. An example of a field number is a row or column number, the condition being the orientation of the table.

XML Code:	`table:field-number`
Rules:	
DTD:	`<!ATTLIST table:filter-condition table:field-number CDATA #REQUIRED>`

Case Sensitive

A case sensitive attribute determines whether a filter condition is case sensitive.

XML Code:	`table:case-sensitive`
Rules:	
DTD:	`<!ATTLIST table:filter-condition table:case-sensitive %boolean; "false">`

Data Type

A data type attribute specifies what data type to use for the filter condition.

XML Code:	`table:data-type`
Rules:	
DTD:	`<!ATTLIST table:filter-condition table:data-type ("text" | "number") "text">`

Value

A value attribute specifies a value for the filter condition.

XML Code:	`table:value`
Rules:	
DTD:	`<!ATTLIST table:filter-condition table:value CDATA #REQUIRED>`

Operator

An operator attribute specifies what operator to use in the filter condition.

XML Code:	`table:operator`
Rules:	The possible values for the operator attribute depend on whether the condition uses a regular expression. If the condition includes a regular expression there are only two possible values: • `match` (matches) • `!match` (does not match) If the condition does not include a regular expression, the possible values are the following conventional relational operators: • = (Equal to) • != (Not equal to) • < (Less than) • > (Greater than) • <= (Less than or equal to) • >= (Greater than or equal to) In addition, you can use non-conventional operators such as “empty” and “!empty”, “bottom values”, “top values”, “bottom percent”, and “top percent”. For example, you can use the latter two operators to filter the lowest and highest percentage of entries.
DTD:	`<!ATTLIST table:filter-condition table:operator CDATA #REQUIRED>`

Example:Representation of a filter

```
<filter>
 <filter-or>
  <filter-and>
   <filter-condition table:field-number=1 table:operator="=" table:value="Doe"/
>
   <filter-condition table:field-number=2 table:operator="=" table:
value="John"/>
  </filter-and>
  <filter-and>
   <filter-condition table:field-number=1 table:operator="=" table:
value="Burns"/>
   <filter-condition table:field-number=2 table:operator="=" table:
value="Michael"/>
  </filter-and>
 </filter-or>
</filter>
```

4.14 Database Ranges

The Database Ranges element contains a collection of Database Ranges.

XML Code:	`<table:database-ranges>`
Rules:	
DTD:	`<!ELEMENT table:database-ranges (table:database-range)* >`

4.14.1 Database Range

A database range is a named area in a table where you can perform database operations.

XML Code:	`<table:database-range>`
Rules:	
DTD:	`<!ELEMENT table:database-range ((table:database-source-sql| table:database-source-table| table:database-source-query)?, table:filter?, table:sort?, table:subtotal-rules?)>`

Database Range Name

The `table:name` attribute specifies the name of the database range on which to perform operations. Only one database range can be without a name. This database range is usually created by the application and is used to filter or sort data in a cell range without the user creating a database range.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:database-range table:name CDATA #IMPLIED>`

Is Selection

The `table:is-selection` attribute specifies whether the database range includes the complete database, or a selection of records from the database.

XML Code:	`table:is-selection`
Rules:	
DTD:	`<!ATTLIST table:database-range table:is-selection %boolean; "false">`

On Update Keep Styles

The `table:on-update-keep-styles` attribute specifies the behavior of the cell styles if the data in the data source changes.

XML Code:	`table:on-update-keep-styles`
Rules:	If the attribute value is true, the style of the new cells in the database range is the same as the other cells in the column. If the attribute value is false, the style of the new cells in the database range is the standard style of the document.
DTD:	`<!ATTLIST table:database-range table:on-update-keep-styles % boolean; "false">`

On Update Keep Size

The `table:on-update-keep-size` attribute specifies the behavior of the database range if the size of the data in the data source changes.

XML Code:	`table:on-update-keep-size`
Rules:	If the attribute value is true, the range retains its size. If the attribute value is false, the range does not retain its size.
DTD:	`<!ATTLIST table:database-range table:on-update-keep-size %boolean; "true">`

Has Persistent Data

The `table:has-persistent-data` attribute specifies whether or not to store the data in a database range.

XML Code:	`table:has-persistent-data`
Rules:	
DTD:	`<!ATTLIST table:database-range table:has-persistent-data %boolean; "true">`

Orientation

The `table:orientation` attribute specifies the orientation of the database range.

XML Code:	`table:orientation`
Rules:	The orientation of a database range can be by row or by column. The only possible values for this attribute are `row` and `column`.
DTD:	`<!ATTLIST table:database-range table:orientation ( "row" | "column" ) "row">`

Contains Header

The `table:contains-header` attribute specifies whether or not the database range contains a header.

XML Code:	`table:contains-header`
Rules:	
DTD:	`<!ATTLIST table:database-range table:contains-header %boolean; "true">`

Display Filter Buttons

The `table:display-filter-buttons` buttons attribute specifies whether or not to display filter buttons.

XML Code:	`table:display-filter-buttons`
Rules:	
DTD:	`<!ATTLIST table:database-range table:display-filter-buttons % boolean; "false">`

Target Range Address

The `table:target-range-address` attribute specifies the cell range address of the database range.

XML Code:	`table:target-range-address`
Rules:	This attribute can be attached to the `<table:database-range>` element.
DTD:	`<!ATTLIST table:database-range table:target-range-address % cell-range-address; #REQUIERED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, a table name must be specified in the address and dollar signs are ignored.

Refresh Delay

The `table:refresh-delay` attribute specifies the time delay between refresh actions.

XML Code:	`table:refresh-delay`
Rules:	This attribute can be associated with the `<table:database-range>` element.
DTD:	`<!ATTLIST table:database-range table:refresh-delay % timeDuration; #IMPLIED >`

4.14.2 Database Source SQL

This element describes an SQL database that integrates with the table.

XML Code:	`<table:database-source-sql>`
Rules:	
DTD:	`<!ELEMENT table:database-source-sql EMPTY?>`

Database Name

A database name attribute specifies the name of the SQL database that the data is imported from.

XML Code:	`table:database-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-sql table:database-name CDATA #REQUIRED>`

SQL Statement

An SQL statement attribute specifies the SQL statement to use when importing data from an SQL database.

XML Code:	`table:sql-statement`
Rules:	
DTD:	`<!ATTLIST table:database-source-sql table:sql-statement CDATA #REQUIRED>`

Parse SQL Statement

A parse SQL statement attribute specifies whether or not the application will parse SQL statements.

XML Code:	`table:parse-sql-statement`
Rules:	
DTD:	`<!ATTLIST table:database-source-sql table:parse-sql-statement % boolean; "false">`

4.14.3 Database Source Table

The database source table element contains the information about the included database and the table.

XML Code:	`<table:database-source-table>`
Rules:	
DTD:	`<!ELEMENT table:database-source-table EMPTY?>`

Database Name

A database name attribute specifies the name of the database that data is imported from.

XML Code:	`table:database-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-table table:database-name CDATA #REQUIRED>`

Table Name

A table name attribute specifies the table that data is imported from.

XML Code:	`table:table-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-table table:table-name CDATA #REQUIRED>`

4.14.4 Database Source Query

The database source query element contains the information about the included database and the query.

XML Code:	`<table:database-source-query>`
Rules:	
DTD:	`<!ELEMENT table:database-source-query EMPTY?>`

Database Name

A database name attribute specifies a database that data is imported from.

XML Code:	`table:database-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-query table:database-name CDATA #REQUIRED>`

Query Name

A query name attribute specifies the query to perform on the database whose data is being imported.

XML Code:	`table:query-name`
Rules:	
DTD:	`<!ATTLIST table:database-source-query table:query-name CDATA #REQUIRED>`

4.14.5 Sort

The sort element describes the sort keys in a database range.

XML Code:	`<table:sort>`
Rules:	
DTD:	`<!ELEMENT table:sort (table:sort-by)+ >`

Bind Styles to Content

The `table:bind-styles-to-content` attribute specifies whether or not cells retain their style attributes after a sort operation.

XML Code:	`table:bind-styles-to-content`
Rules:	
DTD:	`<!ATTLIST table:sort table:bind-styles-to-content %boolean; "true">`

Target Range Address

This attribute specifies where the result of the sort is put. The attribute is used with the `<table:sort>` element in the same way as it is used with the `<table:filter>` element. See Section 4.13.1 for information about using this attribute.

Case Sensitive

The `table:case-sensitive` attribute specifies whether or not the sort operation is case sensitive.

XML Code:	`table:case-sensitive`
Rules:	
DTD:	`<!ATTLIST table:sort table:case-sensitive %boolean; "false">`

Language

The `table:language` attribute specifies the language of the compare operator.

XML Code:	`table:language`
Rules:	
DTD:	`<!ATTLIST table:sort table:language CDATA #IMPLIED>`

Country

This attribute specifies the country for the compare operator.

XML Code:	`table:country`
Rules:	
DTD:	`<!ATTLIST table:sort table:country CDATA #IMPLIED>`

Algorithm

This attribute specifies the algorithm for the compare operator.

XML Code:	`table:algorithm`
Rules:	
DTD:	`<!ATTLIST table:sort table:algorithm CDATA #IMPLIED>`

4.14.6 Sort By

The sort by element specifies which field to sort, the data type of this field, and how to sort it.

XML Code:	`<table:sort-by>`
Rules:	
DTD:	`<!ELEMENT table:sort-by EMPTY >`

Field Number

A field number attribute specifies the row or column number to sort by.

XML Code:	`table:field-number`
Rules:	
DTD:	`<!ATTLIST table:sort-by table:field-number CDATA #REQUIRED>`

Data Type

A data type attribute specifies the data type of the field to be sorted.

XML Code:	`table:data-type`
Rules:	If the attribute value is `automatic`, the application must determine what type of data is in the field. If the field contains a user-defined data type, the attribute value is the name of this data type.
DTD:	`<!ATTLIST table:sort-by table:data-type ("text" | "number" | "automatic" | qname-but-not-ncname ) "automatic" >`

Order

An order attribute specifies whether to sort the data in ascending or descending order.

XML Code:	`table:order`
Rules:	
DTD:	`<!ATTLIST table:sort-by table:order ( "ascending" | "descending" ) "ascending" >`

4.14.7 Subtotal Rules

The subtotal rules element contains the following information:

- The provisional result of a field in a database range, for example, a column.
- The function used to calculate the provisional result.

The element consists of generated groups of fields in the database range. For example, all cells with the same content in the same field form a group.

XML Code:	`<table:subtotal-rules>`
Rules:	
DTD:	`<!ELEMENT table:subtotal-rules ( table:sort-groups? | table:subtotal-rule*) >`

Bind Styles To Content

A bind style to content attribute specifies whether or not cells retain their style features after a subtotal operation.

XML Code:	`table:bind-styles-to-content`
Rules:	This attribute is only evaluated if the `table:sort-groups` element is present.
DTD:	`<!ATTLIST table:subtotal-rules table:bind-styles-to-content % boolean; "true">`

Case Sensitive

A case sensitive attribute specifies whether or not the case of characters is important when comparing entries, for example, when sorting groups.

XML Code:	`table:case-sensitive`
Rules:	This attribute only evaluates if the `table:sort-groups` element is present.
DTD:	`<!ATTLIST table:subtotal-rules table:case-sensitive %boolean; "false">`

Page Breaks On Group Change

A page breaks on group change attribute specifies whether or not to insert a page break after the subtotal for each group.

XML Code:	`table:page-breaks-on-group-change`
Rules:	This attribute only evaluates if the `table:sort-groups` element is present.
DTD:	`<!ATTLIST table:subtotal-rules table:page-breaks-on-group-change % boolean; "false">`

4.14.8 Sort Groups

The sort groups element specifies whether to sort column groups or row groups, and how to sort them. It belongs to the subtotal rules element, see the previous section.

XML Code:	`<table:sort-groups>`
Rules:	
DTD:	`<!ELEMENT table:sort-groups EMPTY >`

Data Type

A data type attribute specifies the data type of the column group or row group to sort.

XML Code:	`table:data-type`
Rules:	If the attribute value is `automatic`, the application must determine what type of data is in the group. If the group contains a user-defined data type, the attribute value is the name of this data type.
DTD:	`<!ATTLIST table:sort-groups table:data-type ("text" | "number" | "automatic" | qname-but-not-ncname ) "auto-matic" >`

Order

An order attribute specifies whether to sort the group data in ascending or descending order.

XML Code:	`table:order`
Rules:	
DTD:	`<!ATTLIST table:sort-groups table:order ( "ascending" | "descending" ) "ascending" >`

4.14.9 Subtotal Rule

The subtotal rule element contains a rule for one field, for example, a column. The rule contains the group field number, which specifies the column group for which the rule is used, and one or more subtotal fields, which specify a field and the function of the field. In summary, the rule describes how to calculate the subtotal.

XML Code:	`<table:subtotal-rule>`
Rules:	
DTD:	`<!ELEMENT table:subtotal-rule (table:subtotal-field)* >`

Group By Field Number

A group by field number attribute specifies the field, for example, a column, that is to be grouped.

XML Code:	`table:group-by-field-number`
Rules:	This attribute can be used with the `<table:subtotal-rule>` element.
DTD:	`<!ATTLIST table:subtotal-rule table:group-by-field-number CDATA #REQUIRED >`

4.14.10 Subtotal Field

The subtotal field element contains the field number and the function that is used to calculate a provisional result. An example of a field is a column.

XML Code:	`<table:subtotal-field>`
Rules:	
DTD:	`<!ELEMENT table:subtotal-field EMPTY >`

Field Number

A field number attribute specifies the index number of the field.

XML Code:	`table:field-number`
Rules:	
DTD:	`<!ATTLIST table:subtotal-field table:field-number CDATA #REQUIRED>`

Function

A function attribute specifies what kind of subtotals to calculate. The following are possible values for this attribute: `auto`, `average`, `count`, `countnums`, `max`, `min`, `product`, `stdev`, `stdevp`, `sum`, `var` and `varp`.

XML Code:	`table:function`
Rules:	
DTD:	`<!ATTLIST table:subtotal-field table:function CDATA #REQUIRED>`

Example: Subtotal field

```
<table:database-range table:range-position="sampletable.A1:sampletable.G20"
table:name="sample">
  <table:database-source-table table:database-name="sampleDB" table:table-
name="sampleTable"/>
  <table:filter ...>
   •
   •
   •
  </table:filter>
  <table:sort>
    <table:sort-by table:field-number=1>
  </table:sort>
  <table:subtotal-rules>
    <table:sort-groups/>
    <table:subtotal-rule table:column-group "3">
      <table:subtotal-field table:field-number="1" table:function="sum"/>
    </table:subtotal-rule>
  </table:subtotal-rules>
</table:database-range>
```

4.15 Data Pilot Tables

Data pilot tables allow you to analyze and evaluate your data. The data pilot tables element can contain several data pilot tables.

XML Code:	`<table:data-pilot-tables>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-tables (table:data-pilot-table)* >`

4.15.1 Data Pilot Table

XML Code:	`<table:data-pilot-table>`
Rules:	This element can contain *one* of the following elements: • `<table:database-source-sql>` (see Section 4.14.2) • `<table:database-source-table>` (see Section 4.14.3) • `<table:database-source-query>` (see Section 4.14.4) • `<table:source-service>` (see Section 4.15.2) • `<table:source-cell-range>` (see Section 4.15.3)
DTD:	`<!ELEMENT table:data-pilot-table ((table:database-source-sql | table:database-source-table | table:database-source-query | table:source-service | table:source-cell-range), table:data-pilot-field+)>`

The attributes associated with the data pilot table element are:

- Data pilot table name
- Application data
- Grand total
- Ignore empty rows
- Identify categories
- Target range address

Data Pilot Table Name

This attribute specifies the name of the data pilot table.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:name CDATA #REQUIRED>`

Application Data

This attribute specifies extra information about the data pilot table, which can be used by the application.

XML Code:	`table:application-data`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:application-data CDATA #IMPLIED>`

Grand Total

This attribute specifies if a column, row, or both have a grand total or if there is a grand total at all.

XML Code:	`table:grand-total`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:grand-total ( "no" | "row" | "column" | "both" ) "both" >`

Ignore Empty Rows

This attribute specifies whether or not empty rows in the source range should be ignored.

XML Code:	`table:ignore-empty-rows`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:ignore-empty-rows % boolean; "false">`

Identify Categories

This attribute specifies whether or not the application orders rows without labels to the next higher category specified by a row label.

XML Code:	`table:identify-categories`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:identify-categories % boolean; "false">`

Target Range Address

This attribute specifies where the target range of the data pilot table output.

XML Code:	`table:target-range-address`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:target-range-address % cell-range-address; #REQUIRED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

Buttons

This attribute specifies all cells which are a button. This is a list of cell-addresses.

XML Code:	`table:buttons`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-table table:buttons %cell-range-address-list; #REQUIRED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

4.15.2 Source Service

A source service element contains information about the service which is used to create the data pilot table.

XML Code:	`<table:source-service>`
Rules:	
DTD:	`<!ELEMENT table:source-service EMPTY >`

The attributes that you can associate with this element are:

- Service name
- Source name
- Object name
- Source username
- Source password

Service Name

This attribute specifies the name of the service.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:source-service table:name CDATA #REQUIRED >`

Source Name

This attribute specifies the source of the service.

XML Code:	`table:source-name`
Rules:	
DTD:	`<!ATTLIST table:source-service table:source-name CDATA #REQUIRED >`

Object Name

This attribute specifies the name of the object in the source which contains the data.

XML Code:	`table:object-name`
Rules:	
DTD:	`<!ATTLIST table:source-service table:object-name CDATA #REQUIRED >`

Source Username

This attribute specifies the username required to access the source.

XML Code:	`table:username`
Rules:	
DTD:	`<!ATTLIST table:source-service table:username CDATA #IMPLIED >`

Source Password

This attribute specifies the password required to access the source.

XML Code:	`table:password`
Rules:	
DTD:	`<!ATTLIST table:source-service table:password CDATA #IMPLIED >`

4.15.3 Source Cell Range

Asource cell range element contains information about the cell range and how the data pilot table gets the data from the range. You can acquire the data with or without a query.

XML Code:	`<table:source-cell-range>`
Rules:	
DTD:	`<!ELEMENT table:source-cell-range (table:filter)? >`

The attributes that you can associate with the source cell range element is:

- Cell range address

Cell Range Address

This attribute specifies the cell range containing the source data.

XML Code:	`table:cell-range-address`
Rules:	
DTD:	`<!ATTLIST table:source-cell-range table:cell-range-address % cell-range-address; #REQUIRED >`
Note:	Discrimination between absolute and relative addresses is not possible. Therefore, you must specify a table name in the address and dollar signs are ignored.

4.15.4 Filter

See Section 4.13.1 for information on filtering in tables.

4.15.5 Data Pilot Field

XML Code:	`<table:data-pilot-field>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-field ( table:data-pilot-level ) ? >`

The attributes that you can associate with the data pilot field element are:

- Source field name
- Is data layout field
- Function
- Orientation
- Used hierarchy

Source Field Name

This attribute specifies the name of the source field. There can be multiple `<table:data-pilot-field>` elements with the same value for this attribute.

XML Code:	`table:source-field-name`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:source-field-name CDATA #REQUIRED >`

Is Data Layout Field

This attribute specifies whether or not the source field is a data layout field.

XML Code:	`table:is-data-layout-field`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:is-data-layout-field % boolean; "false" >`

Function

This attribute specifies the function which is used for the source field. Possible values for this attribute are: `average`, `count`, `countnums`, `max`, `min`, `product`, `stdev`, `stdevp`, `sum`, `var` and `varp`.

XML Code:	`table:function`
Rules:	This attribute is only evaluated if the value of the `table:orientation` attribute is `data`.
DTD:	`<!ATTLIST table:data-pilot-field table:function CDATA #REQUIRED >`

Orientation

This attribute specifies the orientation of the source field. The orientation can be by row, by column, by data, by

page, or hidden.

XML Code:	`table:orientation`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:orientation ( "row" | "column" | "data" | "page" | "hidden" ) #REQUIRED>`

Used Hierarchy

This attribute specifies the used hierarchy of the source field.

XML Code:	`table:used-hierarchy`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-field table:used-hierarchy CDATA "1" >`

4.15.6 Data Pilot Level

The data pilot level element contains information about the level of a data pilot table.

XML Code:	`<table:data-pilot-level>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-level (table:data-pilot-subtotals?, table:data-pilot-members?) >`

The attribute that you can associate with the data pilot level element is:

- Show empty

Show Empty

This attribute specifies whether or not empty fields should be displayed. If this attribute is not present, the application can determine the default setting with the help of the source.

XML Code:	`table:show-empty`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-level table:show-empty %boolean; #IMPLIED >`

4.15.7 Data Pilot Subtotals

The data pilot subtotals element contains information about the provisional result of a field in a data pilot table and the function used to calculate the result. If the element is not present, the application can determine the subtotals with the help of the source.

XML Code:	`<table:data-pilot-subtotals>`
Rules:	This element can contain the data pilot subtotal elements.
DTD:	`<!ELEMENT table:data-pilot-subtotals ( table:data-pilot-subtotal )* >`

4.15.8 Data Pilot Subtotal

Thedata pilot subtotal element contains the function which is used to calculate the subtotal.

XML Code:	`<table:data-pilot-subtotal>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-subtotal EMPTY >`

The attribute that you can associate with the data pilot subtotal element is:

- Function

Function

This attribute specifies the function used for the subtotal. Possible functions are `auto`, `average`, `count`, `countnums`, `max`, `min`, `product`, `stdev`, `stdevp`, `sum`, `var` and `varp`.

XML Code:	`table:function`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-subtotal table:function CDATA #REQUIRED >`

4.15.9 Data Pilot Members

The data pilot members element contains information about the members of the data pilot source. This element can contain data pilot member elements.

XML Code:	`<table:data-pilot-members>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-members ( table:data-pilot-member )* >`

4.15.10 Data Pilot Member

The data pilot member element contains information about a member of the data pilot table.

XML Code:	`<table:data-pilot-member>`
Rules:	
DTD:	`<!ELEMENT table:data-pilot-member EMPTY >`

The attributes that you can associate with the data pilot member element are:

- Member name
- Display
- Show details

Member Name

This attribute specifies the name of the data pilot member.

XML Code:	`table:name`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-member table:name CDATA #REQUIRED>`

Display

This attribute specifies whether or not a data pilot member is visible. If this attribute is not present, the application can determine the default setting with the help of the source.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-member table:display %boolean; #IMPLIED >`

Show Details

This attribute specifies whether or not the details about a data pilot member are displayed. If this attribute is not present, the application can determine the default setting with the help of the source.

XML Code:	`table:show-details`
Rules:	
DTD:	`<!ATTLIST table:data-pilot-member table:show-details %boolean; #IMPLIED >`

4.16 Consolidation

Use this function to combine data from several independent table areas. A new area is calculated via a selected mathematical function and based on those areas.

XML Code:	`<table:consolidation>`
Rules:	
DTD:	`<!ELEMENT table:consolidation EMPTY >`

The attributes that you can associate with this element are:

- Function
- Source cell range addresses

- Target cell address
- Use label
- Link to source data

Function

This attribute contains the function which is used to consolidate the data. Possible functions are `auto`, `average`, `count`, `countnums`, `max`, `min`, `product`, `stdev`, `stdevp`, `sum`, `var` and `varp`.

XML Code:	`table:function`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:function CDATA #REQUIRED >`

Source Cell Range Addresses

This attribute contains a list of cell range addresses. These are the source cell ranges.

XML Code:	`table:source-cell-range-addresses`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:source-cell-range-addresses %cell-range-address-list; #REQUIRED >`

Target Cell Address

This attribute contains the target cell address.

XML Code:	`table:target-cell-address`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:target-cell-address %cell-address; #REQUIRED >`

Use Label

This attribute specifies whether or not labels should be used by the consolidation and if used, which ones. Possible values are `none`, `column`, `row` and `both`.

XML Code:	`table:use-label`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:use-label (none | column | row | both) "none" >`

Link to Source Data

This attribute specifies whether to link the data in the consolidation area to the source data, and whether or not to automatically update the results of the consolidation if any changes are made to the original data.

XML Code:	`table:link-to-source-data`
Rules:	
DTD:	`<!ATTLIST table:consolidation table:link-to-source-data % boolean; "false" >`

4.17 DDE Links

The `<table:dde-links>` container element stores all DDE links. Every link contains the DDE Source and the data of the last connection.

XML Code:	`<table:dde-links>`
Rules:	
DTD:	`<!ELEMENT table:dde-links (table:dde-link)+ >`

4.17.1 DDE Link

This `<table:dde-link>` element contains the DDE source and a simple table element.

XML Code:	`<table:dde-link>`
Rules:	The table does not need a name. The elements in the table do not have styles or other information. Only the data in the cell attributes is used. The cell does not contain paragraphs. The data is stored in the value attributes. The table must contain at least one cell.
DTD:	`<!ELEMENT table:dde-link (table:dde-source, table:table) >`

4.17.2 DDE Source

The DDE source of the `<table:dde-link>` element only supports the value `true` for the `office:automatic-update` attribute.

Additionally, the DDE source has a new attribute `table:conversion-mode`.

Conversion Mode

This attribute specifies the method by which the DDE server converts its data into numbers. There are three possible values.

XML Code:	`table:conversion-mode`
Rules:	
DTD:	`<!ATTLIST table:dde-link table:conversion-mode (into-default-style-data-style|into-english-number|let-text) "into-default-style-data-style" >`

4.18 Table Formatting Properties

The following sections detail the formatting properties that can be applied to tables.

4.18.1 Table Width

Every table must have a fixed width. You specify this width as a fixed length.

You can also specify the width of a table relative to the width of the area that the table is in. In this case, you specify the width as a percentage. User agents that support specifying the relative width of a table can specify widths in this way, but it is not essential.

XML Code:	`style:width` `style:rel-width`
Rules:	

DTD:	`<!ATTLIST style:properties style:width %length; #IMPLIED>`
	`<!ATTLIST style:properties`
	`style:rel-width %percentage; #IMPLIED>`

Notes:	The reasons why every table must have a fixed width and relative widths are only an option are as follows: • Specifying the width of a table by a percentage is useful for current web browsers and other applications where the percentage is relative to the width of a window. But it may cause problems if the percentage relates to a fixed paper width. • Relative widths can also cause problems for applications such as spreadsheet applications, where there is no requirement for a table to fit on a page. However, if an application supports relative widths, it is relatively easy to program the application to calculate a fixed table width, based on a percentage.

4.18.2 Table Alignment

A table alignment property specifies the horizontal alignment of a table.

XML Code:	`table:align`
Rules:	The options for a table alignment property are as follows: • `left` — The table aligns to the left. • `center` — The table aligns to the center. • `right` — The table aligns to the right. • `margins` — The table fills all the space between the left and right margins.
DTD:	`<!ATTLIST style:properties` `table:align (left|center|right|margins) #REQUIRED>`
Notes:	User agents that do not support the `margins` value, may treat this value as `left`.

4.18.3 Table Left and Right Margin

These properties specify the distance of the table from the left and right margins. See Chapter 3 for a full explanation of left and right margin properties.

XML Code:	`fo:margin-left` `fo:margin-right`
Rules:	An application may recognize table margins, but this is not essential. Tables that align to the left or to the center ignore right margins, and tables align to the right or to the center ignore left margins.

4.18.4 Table Top and Bottom Margin

These properties specify the distance of the table from the top and bottom margins, `fo:margin-top` and `fo:margin-bottom`. See Chapter 3 for a full explanation of top and bottom margin properties.

4.18.5 Page Sequence Entry Point

See ? for information on this attribute `style:page-sequence-name`.

4.18.6 Break Before and Break After

These properties insert a page or column break before or after a table, `fo:break-before` and `fo:break-after`. See Section 3.11.24 for a full explanation of these properties.

4.18.7 Table Background and Background Image

These properties specify the background color and image of a table using the attribute `fo:background-color` and the element `<style:background-image>`. See Section 3.11.25 and 3.11.26 for a full explanation of these properties.

4.18.8 Table Shadow

The table shadow property specifies that a shadow visual effect appears on a table, using the attribute `style:shadow`. See Section 3.11.30 for a full explanation of this property.

4.18.9 Keep with Next

The keep with next property specifies that a table stays with the paragraph that follows it, using the attribute `fo:keep-with-next`. See Section 3.11.31 for a full explanation of this property.

4.18.10 May Break Between Rows

This property specifies that a page break may occur inside a table.

XML Code:	`style:may-break-between-rows`
Rules:	
DTD:	`<!ATTLIST style:properties style:may-break-between-rows %boolean; #IMPLIED>`

4.18.11 Border Model Property

The `table:border-model` property specifies what border model to use when creating a table with a border. There are two types of border model, as follows:

- **Collapsing border model**

  When two adjacent cells have different borders, the wider border appears as the border between the cells. Each cell receives half of the width of the border.

- **Separating border model**

  Borders appear within the cell that specifies the border.

Both border models are very similar to the collapsing and separating border models of XSL and **CSS2**. They differ in how border widths relate to row and column widths.

In OpenOffice.org, a row height or column width includes any space required to display borders or padding. This means that, while the width and height of the content area is less than the column width and row height, the sum of the widths of all columns is equal to the total width of the table.

In XSL and CSS2, a column width or row height specifies the width or height of the content area of a cell. This means that the sum of the widths of all columns is less than the width of the table.

XML Code:	`table:border-model`
Rules:	
DTD:	`<!ATTLIST style:properties table:border-model (collapsing|separating) #IMPLIED>`

4.18.12 Page Style

This attribute specifies the name of the page style.

XML Code:	`table:page-style-name`
Rules:	
DTD:	`<!ATTLIST style:properties table:page-style-name %styleName; #IMPLIED>`

4.18.13 Display

This attribute specifies whether or not a table is displayed.

XML Code:	`table:display`
Rules:	
DTD:	`<!ATTLIST style:properties table:display %boolean; #IMPLIED>`

4.19 Column Formatting Properties

The following sections detail the formatting properties that you can apply to table columns.

4.19.1 Column Width

Every table column must have a fixed width. You specify this width as a fixed length.

You can also specify the width of a column relative to the other column widths. Applications that support specifying the relative width of a column can specify widths in this way, but it is not essential.

New information to be supplied.

XML Code:	`style:column-width` `style:rel-column-width`
Rules:	To specify a fixed width, use the `style:column-width` property. To specify a relative width, use the `style:rel-column-width` property, followed by a number signifying the width you require.

DTD:	`<!ENTITY % rel-number "CDATA">`
	`<!ATTLIST style:properties style:column-width %length; #IMPLIED>`
	`<!ATTLIST style:properties style:rel-column-width %rel-number; #IMPLIED>`

Notes:	The relative width of a table is the sum of all of the relative widths of the columns in the table.
Note:	In XSL and CSS, a column width does not include any border widths or padding.

Break Before and Break After

The break before (`fo:break-before`) and break after (`fo:break-after`) attributes can be used to format tables in a similar way to the way they are used to format paragraphs. For tables, the only values you can set for these attributes are "`auto`" or "`page`". See Section 3.11.24 for more information about using these attributes.

4.20 Table Row Formatting Properties

The following sections detail the formatting properties that you can apply to table rows.

4.20.1 Row Height

This property specifies the height of a table row. By default, row height is the height of the tallest item in the row. You can also specify a minimum height or a fixed height.

XML Code:	`style:row-height` `style:min-row-height`
Rules:	To specify a fixed height, use the `style:row-height` property. To specify a minimum height, use the `style:min-row-height` property.
DTD:	`<!ATTLIST style:properties style:row-height %length; #IMPLIED>` `<!ATTLIST style:properties style:min-row-height %length; #IMPLIED>`
Note:	In XSL and CSS, a row height does not include border widths or padding.

Row Background

To apply a background color or a background image to a table row, use the background and background color paragraph formatting properties. See Sections 3.11.25 and 3.11.26 for a full explanation of the background and background color properties.

4.20.2 Break Before and Break After

The break before (`fo:break-before`) and break after (`fo:break-after`) attributes can be used to format table rows in a similar way to the way they are used to format paragraphs. For table rows, the only values you can set for these attributes are "`auto`" or "`page`". See Section 3.11.24 for more information about using these attributes.

4.21 Table Cell Formatting Properties

The following sections detail the formatting properties that you can apply to table cells.

4.21.1 Vertical Alignment

The vertical alignment property allows you to specify the vertical alignment of text in a table cell.

XML Code:	`fo:vertical-align`
Rules:	
DTD:	`<!ATTLIST style:properties` `fo:vertical-align (top|middle|bottom|automatic) #IMPLIED>`
Notes:	The options for the vertical alignment property are as follows: • `top` — Aligns text vertically with the top of the cell. • `middle` — Aligns text vertically with the middle of the cell. • `bottom` — Aligns text vertically with the bottom of the cell. • Automatic – The application decide how to align the text.
Limitations:	There is no XSL property that specifies the vertical alignment of a table cell. This appears to be an oversight rather than deliberate.

4.21.2 Text Align

See Chapter 3 for information on using this property to format table cells.

4.21.3 Text Align Source

This property specifies the source of the text-align property.

XML Code:	`style:text-align-source`
Rules:	If the value of this attribute is "`fix`", only the `fo:text-align` property is used. If the value is "`value-type`", the text alignment is dependent on the `value-type` of the cell.
DTD:	`<!ATTLIST style:properties style:text-align-source (fix|value-type) #IMPLIED>`

4.21.4 Text Outline

See Chapter 3 for information on using this property to format table cells.

4.21.5 Direction

This property specifies the direction of characters in a cell. The most common direction is left to right (`ltr`). The other direction is top to bottom (`ttb`), where the characters in the cell are stacked but not rotated.

XML Code:	`fo:direction`
Rules:	
DTD:	`<!ATTLIST style:properties fo:direction ( "ltr" | "ttb" ) #IMPLIED>`

4.21.6 Vertical Glyph Orientation

This property specifies the vertical glyph orientation. The property specifies an angle or automatic mode. The only possible angle is 0, which disables this feature.

XML Code:	`fo:glyph-orientation-vertical`
Rules:	
DTD:	`<!ATTLIST style:properties style:glyph-orientation-vertical ( auto | 0 ) #IMPLIED>`

4.21.7 Text Shadow

To specify a text shadow within a table cell, use the `fo:text-shadow` formatting property. See Chapter 3 for a full explanation of the text shadow property.

4.21.8 Cell Shadow

To specify a cell shadow in a table cell, use the `style:shadow` formatting property. See Chapter 3 for a full explanation of the shadow property.

4.21.9 Cell Background

To apply a background color or a background image to a table cell, use the background and background color paragraph formatting properties. See Chapte 3 for a full explanation of the background and background color properties.

4.21.10 Cell Borders and Border Line Width

To apply a border to a table cell and specify the width of the border, use the border and border line width paragraph formatting properties. See Sections 3.11.27 and 3.11.28 for a full explanation of the border and border line width properties.

4.21.11 Padding

To apply padding to a table cell, use the padding paragraph formatting property. See Chapter 3 for a full explanation of the padding property.

4.21.12 Left Margin

To specify a left margin in a table cell, use the left margin paragraph formatting property. See Chapter 3 for a full explanation of the left margin property.

4.21.13 Wrap Option

This property is like specified in XSL. In addition the XSL property `fo:overflow` is necessary. The default is not wrap like in XSL, but the default is no-wrap.

XML Code:	`fo:wrap-option` `fo:overflow`
Rules:	
DTD:	`<!ATTLIST style:properties fo:wrap-option (no-wrap|wrap) #IMPLIED>` `<!ATTLIST style:properties fo:overflow (auto) #FIXED>`

4.21.14 Rotation Angle

This property specifies the value of a rotation angle in degrees.

XML Code:	`style:rotation-angle`
Rules:	
DTD:	`<!ATTLIST style:properties style:rotation-angle %number #IMPLIED>`

4.21.15 Rotation Align

This property specifies how the edge of the text in a cell is aligned after a rotation. There are four alignment options:

Alignment	Text is...	Borders and background are...
None.	Rotated.	Unchanged.
Bottom of the cell.	Rotated and may overlap with other cells if the text is longer than the length of the cell.	Positioned parallel to the text, whereby the upper or lower edge is drawn at the original position of the cell.
Top of the cell.		
Center of the cell.		

The OpenOffice.org XML code for the rotation align attribute is described in the following table.

XML Code:	`style:rotation-align`
Rules:	The value of this attribute can be "`none`", "`bottom`", "`top`", or "`center`".
DTD:	`<!ATTLIST style:properties style:rotation-align ( "none" | "bottom" | "top" | "center") #IMPLIED>`

4.21.16 Cell Protect

This property specifies how a cell is protected.

XML Code:	`style:cell-protect`
Rules:	This attribute is only evaluated if the current table is protected (see 4.5.1). The value of the attribute can be "`none`", "`hidden-and-protected`", or a space-separated list containing the values "`protected`" or "`formula-hidden`".
DTD:	`<!ATTLIST style:properties style:cell-protect CDATA #IMPLIED >`

4.21.17 Print Content

This property specifies whether or not the cell content can be printed.

XML Code:	`style:print-content`
Rules:	
DTD:	`<!ATTLIST style:properties style:print-content %boolean; #IMPLIED >`

4.21.18 Data Style

This property contains the name of a data style to use as the data style for the cell. The data style is represented by one of the style elements described in Chapter 2. The style can be referenced by a name.

XML Code:	`style:data-style-name`
Rules:	
DTD:	`<!ATTLIST style:properties style:data-style-name %style-name; #REQUIRED>`

CHAPTER 5

Graphic Content

This chapter provides the OpenOffice.org **XML specification for the core elements of the OpenOffice.org** graphics **applications, OpenOffice.org Draw and OpenOffice.org Impress. It contains the following sections:**

- Master Pages
- Drawing Pages
- Drawing Shapes
- Presentation Shapes
- 3D Shapes
- Graphic Style Elements
- Stroke Properties
- Fill Properties
- Text Animation Properties
- Text Properties
- Graphic Properties
- Shadow Properties
- Connector Properties
- Measure Properties
- Caption Properties
- 3D Geometry Properties
- 3D Lighting Properties
- 3D Texture Properties
- 3D Material Properties
- 3D Shadow Properties
- Layer Sets
- Glue Points
- Presentation Page Layouts
- Presentation Page Attributes

- Presentation Settings
- Presentation Animations

Important Note

OpenOffice.org **Draw is a subset of OpenOffice.org Impress. The OpenOffice.org Draw application does not support presentation features. In this chapter, these features are indicated using the phrase "For presentations only".**

5.1 Master Pages

You use **master pages** as common backgrounds for **drawing pages**. You assign master pages using the `<styles:master-page>` element.

XML Code:	`<style:master-page>`
Rules:	Master pages are stored in the `<office:styles>` element. You must have one master page element. Each drawing page is linked to one master page, which is you specify using the `style:parent-style-name` attribute of the drawing pages style.
DTD	`<!ELEMENT style:master-page ( (style:header, style:header-left?)?, (style:footer, style:footer-left?)?, office:forms?, style:style*, (%shapes;)*, presentation:notes? )>`

The attributes that you can associate with the `<style:master-page>` element are:

- Page name
- Page master
- Page style
- Next style name

The elements that you can include in the `<style:master-page>` element are:

- Presentation notes
- Shapes
- Frames

Page Name

The `style:name` attribute specifies the name of a master page. Each master page is referenced using the page name.

XML Code:	`style:name`
Rules:	This attribute is required and the name specified must be unique.
DTD:	`<!ATTLIST style:master-page style:name %styleName; #REQUIRED>`

Page Master

The `style:page-master-name` attribute specifies the size, border, and orientation of a master page. These settings are collectively called the page master.

XML Code:	`style:page-master-name`
Rules:	This attribute is required.
DTD:	`<!ATTLIST style:master-page style:page-master-name % styleName; #REQUIRED>`

Page Style

You can assign additional page style attributes to a drawing page using the `draw:style-name` attribute.

XML Code:	`draw:style-name`
Rules:	This attribute is optional. The fixed family for page styles is `drawing-page`.
DTD:	`<!ATTLIST draw:page draw:style-name %styleName; #IMPLIED >`

Next Style Name

If the application supports automatically generated pages, the `draw:next-style-name` attribute identifies the master page that is used as a template for the next page.

XML Code:	`draw:next-style-name`
Rules:	This attribute is optional. The value of this attribute must be the name of another `style:master-page` element.
DTD:	`<!ATTLIST style:master-page style:next-style-name % styleName; #IMPLIED>`

5.1.1 Handout Master

For applications that support printing handout pages, this element is a template for automatically generating the handout pages. This element can contain any type of shape. The most useful shape is the `<draw:page-thumbnail>`, which is replaced by actual drawing pages from the document.

XML Code:	`<style:handout-master>`
Rules:	You can only have one `<style:handout-master>` element and it is stored in the `<office:styles>` element.
DTD	`<!ELEMENT style:handout-master (%shapes;)*>`

The attributes that you can associate with the `<style:handout-master>` element are:

- Presentation Page Layout

Presentation Page Layout

This attribute links to a <style:presentation-page-layout> element. See Section 5.23 for information on the presentation page layout element

XML Code:	presentation:presentation-page-layout-name
Rules:	This attribute is optional.
DTD:	<!ATTLIST style:handout-master presentation:presentation-page-layout-name %styleName; #IMPLIED>

5.1.2 Presentation Notes

Each drawing page in a presentation can have an additional presentation notes page, which contains:

- A preview of the drawing page
- Additional graphic shapes

The <presentation:notes> element contains the presentation notes.

XML Code:	<presentation:notes>
Rules:	
DTD:	<!ELEMENT presentation:notes (%shapes;)*>
Notes:	For presentations only.

Example: Master page

```
<office:styles>
  ...
  <style:master-page style:name="home" style:page-master="default">
    <style:style style:name="title" style:family="presentation">
      <style:properties fo:font-style="italic"/>
    </style:style>
    <style:style style:name="subtitle" style:family="presentation"
                 style:parent-style-name="title">
      <style:properties style:text-outline="true"/>
    </style:style>
    <draw:rectangle .../>
    <presentation:notes>
      <draw:text ...>this is a note</draw:text>
    </presentation:notes>
  </style:master-page>
  ...
</office:styles>
```

5.2 Drawing Pages

A drawing page is a container for content in a drawing or presentation document. Drawing pages are used for the following:

- Drawings
- Slides for presentations

XML Code:	`<draw:page>`
Rules:	You must assign a master page to each drawing page.
DTD:	`<!ELEMENT draw:page (office:forms?,(%shapes;)*,presentation:animations?,presentation:notes?)>`

The attributes that you can associate with the `<draw:page>` element are:

- Page name
- Page style
- Master page
- Presentation page layout
- Presentation page properties

The elements that you can include in the `<draw:page>` element are:

- Shapes
- Frames
- Presentation notes
- Forms
- Animations

Page Name

The `draw:name` attribute specifies the name of a drawing page.

XML Code:	`draw:name`
Rules:	This attribute is optional. If you use this attribute, the name must be unique. If you do not use this attribute, the application generates a unique name.
DTD:	`<!ATTLIST draw:page draw:name %styleName; #IMPLIED >`

Page Style

You can assign additional formatting attributes to a drawing page by assigning a page style.

XML Code:	`draw:style-name`
Rules:	This attribute is optional. The fixed family for page styles is `drawing-page`.
DTD:	`<!ATTLIST draw:page draw:style-name %styleName; #IMPLIED >`

Master Page

Each drawing page must have one master page assigned to it. The master page:

- Defines properties such as the size and borders of the drawing page
- Serves as a container for shapes that are used as a common background

The `draw:master-page-name` attribute specifies the name of the master page assigned to the drawing page.

XML Code:	`draw:master-page-name`
Rules:	This attribute is required.
DTD:	`<!ATTLIST draw:page draw:master-page-name %styleName; #REQUIRED>`

Presentation Page Layout

This attribute links to a `<style:presentation-page-layout>` element. See Section 5.23 for information on the presentation page layout element

XML Code:	`presentation:presentation-page-layout-name`
Rules:	This attribute is optional.
DTD:	`<!ATTLIST draw:page presentation:presentation-page-layout-name %styleName; #IMPLIED>`

Presentation Page Properties

Each drawing page can have optional presentation properties, for example, the duration for which a page is displayed or a fade effect. For information on the attributes used to represent these properties, see Section 5.24.

Note: These attributes are for presentations only.

5.2.1 Background Style Properties

A drawing page can have an optional background that defines the background appearance of the page. If you set an optional background, it overrides the background of the assigned master page, but not the shapes that are on the page. You can alter the background of the assigned master page by using one of the following elements in the style element of the page:

- `<style:background-image>` - see Chapter 3
- `<fo:background-color>` - see Chapter 3
- `<draw:hatch>` - see Section 5.6.2
- `<draw:gradient>` - see Section 5.6.1

5.2.2 Presentation Notes

Each drawing page in a presentation can have an additional presentation notes page, which contains a preview of the corresponding drawing page and additional graphic shapes. You can include the `<presentation:notes>` element in the `<draw:page>` element. See Section 5.1.2 for more information about this element.

Example: Drawing page

```
<office:automatic-styles>
 <style:style style:name="gg3434" style:family="drawing-page">
  <style:properties presentation:page-duration="5s">
 </style:style>
 <style:style style:name="titledia"
              style:family="presentation-page-layout">
  <presentation:placeholder presentation:object="title"
                                    svg:x="20%" svg:y="10%"
                                svg:width="80%" svg:height="10%" />
  <presentation:placeholder presentation:object="subtitle"
                                svg:x="20%" svg:y="30%"
                                svg:width="80%" svg:height="60%" />
 </style:style>
<office:automatic-styles>
<office:body>
 <draw:page office:name="Page 1"
            draw:style-name="gg3434"
            draw:master-page-name="home"
            presentation:page-layout-name="titeldia">
    <draw:rect .../>
    <presentation:notes>
      <draw:text ...>this is a note</draw:text>
    </presentation:notes>
 </draw:page>
<office:body>
```

5.3 Drawing Shapes

5.3.1 Rectangle

The <draw:rect> element represents a rectangular drawing shape.

XML Code:	<draw:rect>
Rules:	
DTD:	<!ELEMENT draw:rect (office:events?, (draw:glue-point)*, % draw-text;)> <!ATTLIST draw:rect %draw-position; %draw-size; %draw-style-name; %draw-transform; %draw-end-position; %table-background; %draw-layer; %draw-z-index; %draw-id; %text-anchor; %draw-layer;>

The attributes that you can associate with the <draw:rect> element are:

- Position, Size, Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Text anchor
- Table background

- Draw end position
- Round corners

Round Corners

This attribute specifies the radius of the circle used to round off the corners of the rectangle.

XML Code:	`draw:corner-radius`
Rules:	
DTD:	`<!ATTLIST draw:rect draw:corner-radius %nonNegativeLength; #IMPLIED>`

Example: Rectangular drawing shape

```
<draw:rect svg:x="2cm" svg:y="3cm" svg:width="10cm" svg:height="20cm" svg:
transform="rotate(45)" draw:style-name="object-with-shadow">
```

5.3.2 Line

The `<draw:line>` element represents a line.

XML Code:	`<draw:line>`
Rules:	
DTD:	`<!ELEMENT draw:line ( office:events?, (draw:glue-point)*, % draw-text; )>`

```
<!ATTLIST draw:line     svg:x1      %coordinate; #IMPLIED
                        svg:y1      %coordinate; #IMPLIED
                        svg:x2      %coordinate; #IMPLIED
                        svg:y2      %coordinate; #IMPLIED
                        %draw-style-name;
                        %draw-transform;
                        %draw-end-position;
                        %table-background;
                        %draw-layer;
                        %draw-z-index;
                        %text-anchor;
                        %draw-id;
```

The attributes that you can associate with the `<draw:line>` element are:

- Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Text anchor
- Table background
- Draw end position
- Start point
- End point

Start Point

The start point attributes specify the start coordinates of the line.

XML Code:	`svg:x1` `svg:y1`
Rules:	
DTD:	See above.

End Point

The end point attributes specify the end coordinates of the line.

XML Code:	`svg:x2` `svg:y2`
Rules:	
DTD:	See above.

5.3.3 Polyline

The `<draw:polyline>` element represents a polyline drawing shape.

XML Code:	`<draw:polyline>`
Rules:	

DTD:	`<!ELEMENT draw:polyline ( office:events?, (draw:glue-point)*, %draw-text; )>`
	`<!ATTLIST draw:polyline %draw-position;`
	`%draw-size;`
	`%draw-viewbox;`
	`%draw-style-name;`
	`%draw-transform;`
	`%draw-layer`
	`%draw-z-index`
	`%draw-end-position;`
	`%table-background;`
	`%text-anchor;`
	`%draw-id>`

Implementation limitation:	The OpenOffice.org drawing core currently ignores the size attribute, but writes out a correct one. The size of a shape is determined exclusively from the shape data (i.e. polygon vertices).

The attributes that you can associate with the `<draw:polyline>` element are:

- Position, Size, ViewBox, Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Text anchor
- Table background
- Draw end position
- Points

Points

The svg:points attribute stores a sequence of points, which are connected by straight lines. Each point consists of two coordinates. The coordinates are separated by a comma and the points are separated by white spaces.

XML Code:	svg:points
Rules:	
DTD:	<!ATTLIST draw:polyline svg:points %points; #REQUIRED>

5.3.4 Polygon

The <draw:polygon> element represents a polygon. A polygon is a closed set of straight lines.

XML Code:	<draw:polygon>
Rules:	
DTD:	<!ELEMENT draw:polygon (office:events?, (draw:glue-point)*, %draw-text;)> <!ATTLIST draw:polygon %draw-position; %draw-size; %draw-viewbox; svg:points %Points; #REQUIRED %draw-style-name; %draw-transform; %draw-layer; %draw-z-index; %draw-end-position; %table-background; %text-anchor; %draw-id;>
Implementation limitation:	The OpenOffice.org drawing core currently ignores the size attribute, but writes out a correct one. The size of a shape is determined exclusively from the shape data (i.e. polygon vertices).

The attributes that you can associate with the <draw:polygon> element are:

- Position, Size, ViewBox, Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Text anchor
- Table background
- Draw end position
- Points – see Section 5.3.3

5.3.5 Path

The <draw:path> element represents a path. A path is a shape with a user-defined outline. The shape is built using multiple drawing actions such as:

- *moveto* − set a new current point
- *lineto* − draw a straight line

- *curveto* − draw a curve using a cubic bezier
- *arc* – draw an elliptical or circular arc
- *closepath* − close the current shape by drawing a line to the last *moveto*

Compound paths are paths with subpaths, each subpath consisting of a single *moveto* followed by one or more line or curve operations. Compound paths can be used for effects such as holes in objects.

XML Code:	`<draw:path>`
Rules:	

DTD:	`<!ELEMENT draw:path ( office:events?, (draw:glue-point)*, %draw-text; )>`
	`<!ATTLIST draw:path %draw-position;`
	`%draw-size;`
	`%draw-viewbox;`
	`%draw-style-name;`
	`%draw-transform;`
	`%draw-layer;`
	`%draw-z-index;`
	`%draw-end-position;`
	`%table-background;`
	`%text-anchor;`
	`%draw-id;>`

Implementation limitation:	The OpenOffice.org drawing core currently ignores the size attribute, but writes out a correct one. The size of a shape is determined exclusively from the shape data (i.e. polygon vertices).

The attributes that you can associate with the `<draw:path>` element are:

- Position, Size, ViewBox, Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14.
- Text anchor
- Table background
- Draw end position
- Path data

Path Data

The syntax for this attribute is documented in Chapter 8 of the *Scalable Vector Graphics (SVG) 1.0 Specification W3C Working Draft.* See the Preface for a pointer to this document.

XML Code:	`svg:d`
Rules:	
DTD:	`<!ATTLIST svg:path d %PathData; #REQUIRED>`
Implementation limitation:	The current OpenOffice.org drawing core only supports path shapes which have either open or closed curves. A mix of open and closed curves for one shape is not supported. The elliptical arc command is also not supported yet.

5.3.6 Circle

The `<draw:circle>` element represents a circular drawing shape.

XML Code:	`<draw:circle>`
Rules:	

DTD:	`<!ELEMENT draw:circle ( office:events?, (draw:glue-point)*, %draw-text; )>`
	`<!ATTLIST draw:circle svg:cx %coordinate; #IMPLIED`
	`svg:cy %coordinate; #IMPLIED`
	`svg:r %length; #IMPLIED`
	`%draw-style-name;`
	`%draw-transform;`
	`%draw-layer;`
	`%draw-z-index;`
	`%draw-end-position;`
	`%table-background;`
	`%text-anchor;`
	`%draw-id;>`

The attributes that you can associate with the `<draw:circle>` element are:

- Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Text anchor
- Table background
- Draw end position
- Center point
- Radius
- Kind
- Start angle
- End angle

Center Point

The center point attributes specify the coordinates of the center point of the circle.

XML Code:	`svg:cx` `svg:cy`
Rules:	
DTD:	`<!ATTLIST draw:circle svg:cx %coordinate; #IMPLIED` `svg:cy %coordinate; #IMPLIED>`

Radius

The radius attribute specifies the radius of the circle.

XML Code:	`svg:r`
Rules:	
DTD:	`<!ATTLIST draw:circle svg:r %length; #IMPLIED>`

Kind

The `draw:kind` attribute specifies the appearance of the circle.

XML Code:	`draw:kind`
Rules:	
DTD:	`<!ATTLIST draw:circle draw:kind (full|section|cut|arc) "full">`

Start Angle

For circles where the `draw:kind` attribute value is `section`, `cut` or `arc`, the `draw:start-angle` attribute specifies the start angle of the section, cut, or arc.

XML Code:	`svg:start-angle`
Rules:	
DTD:	`<!ATTLIST draw:circle draw:start-angle %nonNegativeInteger; #IMPLIED>`

End Angle

For circles where the `draw:kind` attribute value is `section`, `cut` or `arc`, the `draw:end-angle` attribute specifies the end angle of the section, cut, or arc.

XML Code:	`svg:end-angle`
Rules:	
DTD:	`<!ATTLIST draw:circle draw:end-angle %nonNegativeInteger; #IMPLIED>`

5.3.7 Ellipse

The `<draw:ellipse>` element represents an ellipse.

XML Code:	`<draw:ellipse>`
Rules:	

DTD:	`<!ELEMENT draw:ellipse ( office:events?, (draw:glue-point)*, %draw-text; )>`
	`<!ATTLIST draw:ellipse svg:cx %coordinate; #IMPLIED`
	`svg:cy %coordinate; #IMPLIED`
	`svg:rx %length; #IMPLIED`
	`svg:ry %length; #IMPLIED`
	`%draw-style-name;`
	`%draw-transform;`
	`%draw-layer;`
	`%draw-z-index;`
	`%draw-end-position;`
	`%table-background;`
	`%text-anchor;`
	`%draw-id;>`

The attributes that you can associate with the <draw:ellipse> element are:

- Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Center point, Kind, Start angle, End angle, Radius – see Section 5.3.6
- Text anchor
- Table background
- Draw end position

5.3.8 Connector

The <draw:connector> element represents a series of lines that connect to glue points on two other shapes.

XML Code:	<draw:connector>
Rules:	
DTD:	<!ELEMENT draw:connector (office:events?, (draw:glue-point)*, %draw-text;)> <!ATTLIST draw:connector %draw-style-name; %draw-layer; %draw-z-index; %draw-end-position; %table-background; %text-anchor; %draw-id;>

The attributes that you can associate with the <draw:connector> element are:

- Style, Layer, Z-Index and ID – see Section 5.3.14
- Type
- Start position
- Start shape
- Start glue point
- End position
- End shape
- End glue point
- Line show
- Text anchor
- Table background
- Draw end position

Type

The draw:type attribute specifies how the connection between two points is rendered.

XML Code:	`draw:type`
Rules:	The value of this attribute can be `standard`, `lines`, `line`, or `curve`.
DTD:	`<!ATTLIST draw:connector` `draw:type (standard|lines|line|curve) "standard">`

Start Position

The start position attributes specify the start position of a connector.

XML Code:	`svg:x1` `svg:y1`
Rules:	If the start position is connected to a shape, these attributes are optional because the start position defaults to the corresponding glue point on the target shape.

DTD:	`<!ATTLIST draw:connector`
	`svg:x1 %coordinate; #IMPLIED`
	`svg:y1 %coordinate; #IMPLIED>`

Start Shape

The `draw:start-shape` attribute identifies the drawing shape to which the start of this connector is connected.

XML Code:	`draw:start-shape`
Rules:	If a shape is connected to the start of a connector, the start position defaults to the corresponding glue point on the target shape.
DTD:	`<!ATTLIST draw:connector` `draw:start-shape %shapeId; #IMPLIED>`

Start Glue Point

The `draw:start-glue-point` attribute identifies the glue point in the start shape of the connector.

XML Code:	`draw:start-glue-point`
Rules:	If this attribute is not set and the start of the connector is connected to a shape, the application chooses the glue point. If the start of the connector is not connected to a shape, this attribute is ignored.
DTD:	`<!ATTLIST draw:connector` `draw:start-glue-point %integer; #IMPLIED>`

End Position

The end position attributes specify the end position of a connector.

XML Code:	`svg:x2` `svg:y2`
Rules:	If the end position is connected to a shape, these attributes are optional because the end position defaults to the corresponding glue point on the target shape.

DTD:	`<!ATTLIST draw:connector`
	`svg:x2 %coordinate; #IMPLIED`
	`svg:y2 %coordinate; #IMPLIED>`

End Shape

The `draw:end-shape` attribute identifies the drawing shape to which the end of the connector is connected.

XML Code:	`draw:end-shape`
Rules:	If a shape is connected to the end of a connector, the end position defaults to the corresponding glue point on the target shape.
DTD:	`<!ATTLIST draw:connector` `draw:end-shape %shapeId; #IMPLIED>`

End Glue Point

The `draw:end-glue-point` attribute identifies the glue point in the end shape of the connector.

XML Code:	`draw:end-glue-point`
Rules:	If this attribute is not set and the end of the connector is connected to a shape, the application chooses the glue point. If the end of the connector is not connected to a shape, this attribute is ignored.
DTD:	`<!ATTLIST draw:connector` `draw:end-glue-point %integer; #IMPLIED>`

Line Skew

The `draw:line-skew` attribute controls the generation of the lines that connect the start and end points. Depending on the type of connector, this can vary from one to three distances that move the connector lines relative to their normal position.

XML Code:	`draw:line-skew`
Rules:	
DTD:	`<!ATTLIST draw:connector` `draw:line-skew CDATA #IMPLIED>`

5.3.9 Caption

The `<draw:caption>` element represents a rectangular drawing shape with an additional set of lines. It can be used as a description for a fixed point inside a drawing.

XML Code:	`<draw:caption>`
Rules:	

DTD:	`<!ELEMENT draw:caption ( office:events?, (draw:glue-point)*, %draw-text; )>`
	`<!ATTLIST draw:caption`
	`%draw-position;`
	`%draw-size;`
	`%draw-style-name;`
	`%draw-transform;`
	`%draw-layer;`
	`%draw-z-index;`
	`%draw-end-position;`
	`%table-background;`
	`%text-anchor;`
	`%draw-id;>`

The attributes that you can associate with the `<draw:caption>` element are:

- Position, Size, Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Caption point
- Round corners
- Text anchor
- Table background
- Draw end position

Caption Point

The caption point attributes specify the position of the point that is captioned. A set of lines are rendered from the caption area.

XML Code:	`draw:caption-point-x` `draw:caption-point-y`
Rules:	

DTD:	`<!ATTLIST draw:caption`
	`draw:caption-point-x %coordinate; #IMPLIED`
	`draw:caption-point-y %coordinate; #IMPLIED >`

Round Corners

The `draw:corner-radius` attribute specifies the radius of the circle used to round off the corners of the caption.

XML Code:	`draw:corner-radius`
Rules:	
DTD:	`<!ATTLIST draw:caption draw:corner-radius %length; #IMPLIED>`

5.3.10 Measure

The <draw:measure> element represents a shape that is used to measure distances in drawings.

XML Code:	<draw:measure>
Rules:	
DTD:	<!ELEMENT draw:measure (office:events?, (draw:glue-point)*, %draw-text;)> <!ATTLIST draw:measure %draw-style-name; %draw-transform; %draw-end-position; %table-background; %text-anchor; %draw-layer; %draw-z-index; %draw-id;>

The attributes that you can associate with the <draw:measure> element are:

- Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14
- Start position
- End position
- Text anchor
- Table background
- Start position
- Draw end position

Start Position

These attributes specify the start point of the measured distance.

XML Code:	svg:x1 svg:y1
Rules:	
DTD:	<!ATTLIST draw:measure svg:x1 %coordinate; #REQUIRED> <!ATTLIST draw:measure svg:y1 %coordinate; #REQUIRED>

Draw End Position

These attributes specify the end point of the measured distance.

XML Code:	svg:x2 svg:y2
Rules:	
DTD:	<!ATTLIST draw:measure svg:x2 %coordinate; #REQUIRED> <!ATTLIST draw:measure svg:y2 %coordinate; #REQUIRED>

5.3.11 Control

The <draw:control> element represents a shape that is linked to a control inside an <office:forms> element.

XML Code: <draw:control>

Rules:

DTD:
```
<!ELEMENT draw:control EMPTY>
<!ATTLIST draw:control
                          %draw-position;
                          %draw-size;
                          %draw-style-name;
                          %draw-layer;
                          %control-id;
                          %draw-z-index;
                          %draw-end-position;
                          %table-background;
                          %text-anchor;
                          %draw-id;>
```

The attributes that you can associate with the <draw:control> element are:

- Position, Size, Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14.
- Control ID, see the documentation for forms.
- Text anchor
- Table background
- Draw end position

5.3.12 Page Thumbnail

The <draw:page-thumbnail> element represents a rectangular area that displays the thumbnail of a drawing page.

XML Code: <draw:page-thumbnail>

Rules:

DTD:
```
<!ELEMENT draw:page-thumbnail EMPTY>
<!ATTLIST draw:page-thumbnail
                    %draw-position;
                    %draw-size;
                    %presentation-class;
                    %draw-end-position;
                    %table-background;
                    %text-anchor;
                    %draw-layer;
                    %draw-z-index;
                    %draw-id>
```

The attributes that you can associate with the <draw:page-thumbnail> element are:

- Position, Size, Style, Layer, Z-Index, ID, and Transformation – see Section 5.3.14.
- Presentation class – see Section 5.4.1

- Control ID, see the documentation for forms.
- Text anchor
- Table background
- Page number

Page Number

The `draw:page-number` attribute specifies the number of the page that is displayed as a thumbnail.

XML Code:	`draw:page-number`
Rules:	For thumbnails on notes pages, the value of this attribute is fixed to the drawing page of the notes page. For thimbnails on handout master pages, the value of this attribute is relative and specifies in which order the pages are previewed on the handout.
DTD:	`<!ATTLIST draw:page-thumbnail draw:page-number % positiveInteger; #IMPLIED>`

5.3.13 Grouping

The `<draw:g>` element represents a group of drawing shapes.

XML Code:	`<draw:g>`
Rules:	
DTD:	`<!ELEMENT draw:g ( office:events?, (%shapes;)* ) > <!ATTLIST draw:g %draw-z-index %draw-id>`

5.3.14 Common Drawing Shape Attributes

The attributes described in this section are common to all drawing shapes.

Position

The position attributes specify the `x` and `y` coordinates of the start postion of the drawing shape.

XML Code:	`svg:x` and `svg:y`
Rules:	

DTD:	`<!ENTITY % Coordinate "CDATA">`
	`<!ENTITY % draw-position "svg:x %coordinate; #IMPLIED`
	`svg:y %coordinate; #IMPLIED">`

Size

The size attributes specify the width and height of the drawing shape.

XML Code:	`svg:width` and `svg:height`
Rules:	
DTD:	`<!ENTITY % draw-size "svg:width %coordinate; #IMPLIED svg:height %coordinate; #IMPLIED>`

End Position

You only need to specify the end position of a shape when the shape is included in a spreadsheet document and the anchor of the shape is in a cell. You specify the end position using the cell address of the cell in which the end position is located, and the `x` and `y` coordinates of the end position relative to the top left edge of the cell.

XML Code:	`table:end-cell-address`
	`table:end-x`
	`table:end-y`

Rules:	
DTD:	`<!ENTITY % draw-end-position "table:end-cell-address %Cell-Address; #IMPLIED table:end-x %coordinate; #IMPLIED table:end-y %coordinate; #IMPLIED">`

Table Background

The `table:table-background` attribute specifies whether or not the shape is in the table background when a shape is included in a spreadsheet document. If the style does not contain this attribute the shape is not included in the background of the table.

XML Code:	`table:table-background`
Rules:	
DTD:	`<!ENTITY % table-background "table:table-background (true | false ) #IMPLIED">`

Transformation

The `draw:transform` attribute specifies a list of transformations that can be applied to a drawing shape.

XML Code:	`draw:transform`
Rules:	The value of this attribute is a list of transform definitions, which are applied to the drawing shape in the order in which they are listed. The transform definitions in the list must be separated by a white space and/or a comma. The types of transform definitions available include: • `matrix(<a> <b> <c> <d> <e> <f>)`, which specifies a transformation in the form of a transformation matrix of six values. `matrix(a,b,c,d,e,f)` is the equivalent of applying the transformation matrix `[a b c d e f]`. • `translate(<tx> [<ty>])`, which specifies a translation by `tx` and `ty`. • `scale(<sx> [<sy>])`, which specifies a scale operation by `sx` and `sy`. If `<sy>` is not provided, it is assumed to be equal to `<sx>`. • `rotate(<rotate-angle>)`, which specifies a rotation by `<rotate-angle>` about the origin of the shapes coordinate system. • `skewX(<skew-angle>)`, which specifies a skew transformation along the X axis. • `skewY(<skew-angle>)`, which specifies a skew transformation along the Y axis.
DTD:	`<!ENTITY % transform-list; CDATA>` `<!ENTITY % draw-transform "draw:transform %TransformList; #IMPLIED">`

ViewBox

The `svg:viewbox` attribute establishes a user coordinate system inside the physical coordinate system of the shape specified by the position and size attributes. This user coordinate system is used by the `svg:points` attribute and the `<svg:path>` element.

XML Code:	`svg:viewbox`
Rules:	The syntax for using this attribute is the same as the SVG syntax. The value of the attribute is four numbers seperated by white spaces, which define the left, top, right, and bottom dimensions of the user coordinate system.
DTD:	`<!ENTITY %draw-viewbox "svg:viewbox CDATA #REQUIRED">`
Implementation limitation:	The OpenOffice.org drawing core currently ignores the viewbox attribute, but writes out a correct one. The implied coordinate system has its origin at the left, top corner of the shape, without any scaling relative to the shape.

Style

The `draw:style-name` attribute specifies a style for the drawing shape. The attributes of the specified style and its optional parent styles are used to format the shape.

XML Code:	`draw:style-name`
Rules:	The value of this attribute can be a `<style:style>` element with a style family value of `graphic`.
DTD:	`<!ENTITY % draw-style-name "draw:style-name %styleName; #REQUIRED">`

Layer

Each shape can be assigned to a layer. The `draw:layer` attribute specifies the layer to which a shape is assigned.

XML Code:	`draw:layer`
Rules:	The value of this attribute must be the name of a layer inside the layer-set of the document.
DTD:	`<!ENTITY % draw-layer "draw:layer %layerName; #IMPLIED">`

ID

You can use the `draw:id` attribute to assign a unique ID to each drawing shape that is contained in a document. You can reference the drawing shape using the unique ID. The value of the attribute is only exported if the shape is referenced.

XML Code:	`draw:id`
Rules:	This attribute is optional.
DTD:	`<!ENTITY % draw-id "draw:id %shapeId; #IMPLIED">`

Z-Index

Drawing shapes are rendered in a specific order. In general, the shapes are rendered in the order in which they appear in the XML document. To change the order, use the `draw:z-index` attribute.

XML Code:	`draw:z-index`
Rules:	This attribute is optional.
DTD:	`<!ENTITY % draw-z-index "draw:z-index %nonNegativeInteger;  #IMPLIED">`

5.4 Presentation Shapes

Presentation shapes are special shapes contained in a presentation. Presentation shapes use styles with a style family value of `presentation`, unlike drawing shapes which use styles with a style family value of `graphic`. Presentation shapes can be empty, acting only as placeholders.

Standard drawing shapes can also be used in presentations. The `presentation:class` attribute distinguishes presentation shapes from drawing shapes.

5.4.1 Common Presentation Shape Attributes

The attributes described in this section are common to all presentation shapes.

Style

Presentation shapes can have styles from the style family `presentation` assigned to them. You can distinguish a presentation shape from a drawing shape by checking the style attribute used. A drawing shape uses a `draw:`

style-name attribute with a style from the graphics family, while a presentation shape uses a presentation:style-name attribute with a style from the presentation family. This name links to a <style:style> element with the family presentation. The attributes in this style and its optional parent styles are used to format this shape.

XML Code:	presentation:style-name
Rules:	
DTD:	<!ENTITY % presentation-style-name "presentation:style-name % styleName; #IMPLIED">

Class

The presentation:class attribute specifies the class to which the presentation belongs. The presentation:placeholder attribute defines if this is a placeholder or a presentation object with actual content. The presentation:user-transformed attribute specifies whether the size and position of the shape is set by the user or is set by the corresponding presentation shape on the master page.

XML-Code:	presentation:class presentation:placeholder presentation:user-transformed
Rules:	
DTD:	<!ENTITY % presentation-classes "(title|outline|subtitle|text|graphic|object|chart|table|orgchart|page|notes)" > <!ENTITY % presentation-class "presentation:class % presentation-classes; #IMPLIED presentation:placeholder (true|false) #IMPLIED presentation:user-transformed (true|false) #IMPLIED">

5.4.2 Title

Titles are standard text shapes . The class name for this presentation shape is presentation-title.

5.4.3 Outline

Outlines are standard text shapes. The class name for this presentation shape is presentation-outline.

5.4.4 Subtitle

Subtitles are standard text shapes. The class name for this presentation shape is presentation-subtitle.

5.4.5 Text

Presentation texts are standard text shapes. The class name for this presentation shape is presentation-text.

5.4.6 Graphic

Presentation graphics are standard graphic shapes . The class name for this presentation shape is `presentation-text`.

5.4.7 Object

Presentation objects are standard OLE shapes. The class name for this presentation shape is `presentation-object`.

5.4.8 Chart

Presentation charts are standard OLE shapes. The class name for this presentation shape is `presentation-chart`.

5.4.9 Table

Presentation tables are standard OLE shapes. The class name for this presentation shape is `presentation-table`.

5.4.10 Orgcharts

Presentation organization charts are standard OLE shapes. The class name for this presentation shape is `presentation-orgchart`.

Note: Currently, orgcharts are not implemented in OpenOffice.org**.**

5.4.11 Pages

Presentation pages are used on notes pages. The class name for this presentation shape is page.

5.4.12 Notes

Presentation notes are used on notes pages. The class name for this presentation shape is notes.

5.5 3D Shapes

5.5.1 Scene

The `<dr3d:scene>` element is the only element that can contain three-dimensional shapes. A scene is like a group, but it also defines the projection, lighting, and other render details for the shapes inside the scene.

XML Code:	`<dr3d:scene>`
Rules:	

DTD:	`<!ENTITY % shapes3d "(dr3d:scene|dr3d:extrude|dr3d:sphere| dr3d:rotate|dr3d:cube)">`
	`<!ELEMENT dr3d:scene (dr3d:light*,(%shapes3d;)*)>`
	`<!ATTLIST dr3d:scene`
	`%draw-position;`
	`%draw-size;`
	`%dr3d-transform;`
	`%draw-style-name;`
	`%draw-layer;`
	`%draw-z-index;`
	`%draw-id;`
	`%draw-layer;>`

The attributes that you can associate with the `<dr3d:scene>` element are:

- Style, Layer, Z-Index, and ID – see Section 5.3.14
- Camera vectors
- Projection
- Distance
- Focal length
- Shadow slant
- Shade mode
- Ambient color
- Lighting mode

Camera Vectors

The camera vectors define a viewing volume. The `dr3d:vrp` attribute specifies the origin, the `dr3d:vpn` attribute points towards the projected objects, and the `dr3d:vup` attribute defines the up vector.

XML Code:	`dr3d:vrp`
	`dr3d:vpn`
	`dr3d:vup`

Rules:	

DTD:	`<!ATTLIST dr3d:scene dr3d:vrp %vector3D; #IMPLIED>`
	`<!ATTLIST dr3d:scene dr3d:vpn %vector3D; #IMPLIED>`
	`<!ATTLIST dr3d:scene dr3d:vup %vector3D; #IMPLIED>`

Projection

The `dr3d:projection` attribute specifies the projection. The projection can be perspective or parallel. In perspective mode, objects become smaller in the distance.

XML Code:	`dr3d:projection`
Rules:	
DTD:	`<!ATTLIST dr3d:scene dr3d:projection (parallel|perspective) #IMPLIED>`

Distance

The `dr3d:distance` attribute specifies the distance between the camera and the object.

XML Code:	`dr3d:distance`
Rules:	
DTD:	`<!ATTLIST dr3d:scene dr3d:distance %length; #IMPLIED>`

Focal Length

The `dr3d:focal-length` attribute specifies the length of the focus for the virtual camera of this scene.

XML Code:	`dr3d:focal-length`
Rules:	
DTD:	`<!ATTLIST dr3d:scene dr3d:focal-length %length; #IMPLIED>`

Shadow Slant

XML Code:	`dr3d:shadow-slant`
Rules:	
DTD:	`<!ATTLIST dr3d:scene dr3d:shadow-slant %nonNegativeInteger; #IMPLIED>`

Shade Mode

XML Code:	`dr3d:shadow-mode`
Rules:	
DTD:	`<!ATTLIST dr3d:scene dr3d:shade-mode (flat|phong|gouraud|draft) #IMPLIED>`

Ambient Color

XML Code:	dr3d:ambient-color
Rules:	
DTD:	<!ATTLIST dr3d:scene dr3d:ambient-color %color; #IMPLIED>

Lighting Mode

XML Code:	dr3d:lighting-mode
Rules:	
DTD:	<!ATTLIST dr3d:scene dr3d:lighting-mode %boolean; #IMPLIED>

5.5.2 Light

The <dr3d:light> element represents a light inside a scene.

XML Code:	<dr3d:light>
Rules:	This element must be the first element contained in a <dr2d:scene> element. The OpenOffice.org software currently supports up to 8 lights per scene.
DTD:	<!ELEMENT dr3d:light EMPTY> <!ATTLIST dr3d:light dr3d:diffuse-color %color; #IMPLIED dr3d:direction %vector3D; #REQUIRED dr3d:enabled %boolean; #IMPLIED dr3d:specular %boolean; #IMPLIED>

The attributes that you can associate with the <dr3d:light> element are:

- Diffuse color
- Direction
- Enabled
- Specular

Diffuse Color

The dr3d:diffuse-color attribute specifies the base color that the light is emitting.

XML Code:	dr3d:diffuse-color
Rules:	
DTD:	<!ATTLIST dr3d:light dr3d:diffuse-color %color; #IMPLIED>

Direction

The dr3d:direction attribute specifies the direction in which the light is emitted.

XML Code:	dr3d:direction
Rules:	
DTD:	<!ATTLIST dr3d:light dr3d:direction %vector3D; #REQUIRED>

Enabled

The dr3d:enabled attribute specifies whether or not the light is enabled. If a light is not enabled, it does not emit any light.

XML Code:	dr3d:enabled
Rules:	
DTD:	<!ATTLIST dr3d:light dr3d:enabled %boolean; #IMPLIED>

Specular

The dr3d:specular attribute specifies whether or not thes light causes a specular reflection on the objects.

XML Code:	dr3d:specular
Rules:	Currently, this attribute can only be applied to the first light in a scene.
DTD:	<!ATTLIST dr3d:light dr3d:specular %boolean; #IMPLIED>

5.5.3 Cube

The <dr3d:cube> element represents a three-dimensional cube shape.

XML Code:	<dr3d:cube>
Rules:	
DTD:	<!ELEMENT dr3d:cube EMPTY> <!ATTLIST dr3d:cube dr3d:min-edge %vector3D; #IMPLIED dr3d:max-edge %vector3D; #IMPLIED %dr3d-transform; %draw-style-name; %draw-layer; %draw-z-index; %draw-id; %draw-layer;>

The attributes that you can associate with the <dr3d:cube> element are:

- Style, Layer, Z-Index and ID – see Section 5.3.14
- Projection, Distance, Focal length, Shadow slant, Shade mode, Ambient color, Lighting mode – see Section 5.5.1
- Minimum and Maximum Edge

Minimum and Maximum Edge

These attributes specify the minimum and maximum edge of the cube in a 3D space.

XML Code:	`dr3d:min-edge` `dr3d:max-edge`
Rules:	

DTD:	`<!ATTLIST dr3d:cube`
	`dr3d:min-edge %vector3D; #IMPLIED`
	`dr3d:max-edge %vector3D; #IMPLIED>`

5.5.4 Sphere

The `<dr3d:sphere>` element represents a three-dimensional sphere shape.

XML Code:	`<dr3d:sphere>`
Rules:	

DTD:	`<!ELEMENT dr3d:sphere EMPTY>`
	`<!ATTLIST dr3d:sphere`
	`dr3d:center %vector3D; #IMPLIED`
	`dr3d:size %vector3D; #IMPLIED`
	`%dr3d-transform;`
	`%draw-style-name;`
	`%draw-layer;`
	`%draw-z-index;`
	`%draw-id;`
	`%draw-layer;>`

The attributes that you can associate with the `<dr3d:sphere>` element are:

- Style, Layer, Z-Index, and ID – see Section 5.3.14
- Projection, Distance, Focal length, Shadow slant, Shade mode, Ambient color, Lighting mode – see Section 5.5.1
- Center
- Size

Center

The `dr3d:center` attribute defines the center of the sphere in a three-dimensional space.

XML Code:	`dr3d:center`
Rules:	
DTD:	`<!ATTLIST dr3d:sphere` `dr3d:center %vector3D; #IMPLIED>`

Size

The `dr3d:size` attribute defines the size of the sphere in a three-dimensional space.

XML Code:	dr3d:size
Rules:	
DTD:	<!ATTLIST dr3d:sphere dr3d:size %vector3D; #IMPLIED>

5.5.5 Extrude

The <dr3d:extrude> element represents a three-dimensional extrude based on a polygon.

XML Code:	<dr3d:extrude>
Rules:	
DTD:	<!ELEMENT dr3d:extrude EMPTY> <!ATTLIST dr3d:extrude svg:d %pathData; #REQUIRED %draw-viewbox; %dr3d-transform; %draw-style-name; %draw-layer; %draw-z-index; %draw-id; %draw-layer;>

The attributes that you can associate with the <dr3d:extrude> element are:

- Viewbox, Style, Layer, Z-Index, and ID – see Section 5.3.14
- Projection, Distance, Focal length, Shadow slant, Shade mode, Ambient color, Lighting mode – see Section 5.5.1.
- Path Data – see Section 5.3.5

5.5.6 Rotate

The <dr3d:rotate> element represents a three-dimensional rotation shape based on a polygon.

XML Code:	<dr3d:rotate>
Rules:	
DTD:	<!ELEMENT dr3d:rotate EMPTY> <!ATTLIST dr3d:rotate svg:d %pathData; #REQUIRED %draw-viewbox; %dr3d-transform; %draw-style-name; %draw-layer; %draw-z-index; %draw-id; %draw-layer;>

The attributes that you can associate with the <dr3d:rotate> element are:

- Viewbox, Style, Layer, Z-Index, and ID – see Section 5.3.14
- Projection, Distance, Focal length, Shadow slant, Shade mode, Ambient color, Lighting mode – see Section

5.5.1.

- Path Data – see Section 5.3.5

5.6 Graphic Style Elements

The elements described in this section are located in the <office:styles> section of a document and are refered to by a unique name. The following styles for filling graphic objects are available:

- Gradient
- Hatch
- Image
- Transparency
- Marker
- Dash

5.6.1 Gradient

This element defines a gradient for filling a drawing object.

XML Code:	<draw:gradient>
Rules:	This element must be located inside the <office:styles> elements.
DTD:	<!ELEMENT draw:gradient EMPTY>

The attributes that you can associate with the gradient element are:

- Name
- Gradient style
- Gradient center
- Colors
- Intensity
- Angle
- Border

Name

This attribute uniquely identifies a gradient inside an <office:styles> element.

XML Code:	draw:name
Rules:	
DTD:	<!ATTLIST draw:gradient draw:name %styleName; #REQUIRED>

Gradient Style

This attribute specifies the style of the gradient.

XML Code:	`draw:style`
Rules:	The gradient styles that OpenOffice.org currently supports are `linear`, `axial`, `radial`, `ellipsoid`, `square`, and `rectangular`.
DTD:	`<!ENTITY % gradient-style "(linear|axial|radial|ellipsoid| square|rectangular)"> <!ATTLIST draw:gradient draw:style %gradient-style; #REQUIRED>`

Gradient Center

If the gradient style is `radial`, `ellipsoid`, `square`, or `rectangular`, the gradient center attribute specifies the center of the geometry that is used for the gradient.

XML Code:	`draw:cx` and `draw:cy`
Rules:	The values of these attributes are always percentage values.
DTD:	`<!ATTLIST draw:gradient draw:cx %coordinate; #IMPLIED draw:cy %coordinate; #IMPLIED>`

Colors

The gradient interpolates between a start color and an end color, which are specified using the following attributes.

XML Code:	`draw:start-color` and `draw:end-color`
Rules:	
DTD:	`<!ATTLIST draw:gradient draw:start-color %color; #REQUIRED> <!ATTLIST draw:gradient draw:end-color %color; #REQUIRED>`

Intensity

The intensity attributes allow you to use base colors for interpolation and to modify the start and end color intensity using percentage values. In OpenOffice.org, this functionality is only used where a common color is used for the user interface and it can be modified using a different intensity value.

XML Code:	`draw:start-intensity` and `draw:end-intensity`
Rules:	These attributes are optional. If the attributes are not specified, the colors are used as they are, that is at 100% intensity.
DTD:	`<!ATTLIST draw:gradient draw:start-intensity %percentage; #IMPLIED> <!ATTLIST draw:gradient draw:end-intensity %percentage; #IMPLIED>`

Angle

The angle attribute specifies an angle that rotates the axis at which the gradient values are interpolated.

XML Code:	`draw:angle`
Rules:	This attribute is ignored for radial style gradients.
DTD:	`<!ATTLIST draw:gradient draw:angle %angle; #IMPLIED>`

Border

Depending on the style of the gradient, the border attribute specifies a percentage value which is used to scale a border which is filled by the start or end color only.

XML Code:	`draw:border`
Rules:	
DTD:	`<!ATTLIST draw:gradient draw:border %percentage; #IMPLIED>`

5.6.2 Hatch

This element defines a hatch for filling graphic objects. A hatch is a simple pattern of straight lines that is repeated in the fill area.

XML Code:	`<draw:hatch>`
Rules:	These elements must be located inside the `<office:styles>` elements.
DTD:	`<!ELEMENT draw:hatch EMPTY>`

The attributes that you can associate with the hatch element are:

- Name
- Style
- Color
- Distance
- Angle
- Background

Name

This attribute uniquely identifies a hatch inside an `<office:styles>` element.

XML Code:	`draw:name`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:name %styleName; #REQUIRED>`

Style

The style attribute specifies the style of the hatch.

XML Code:	`draw:style`
Rules:	The hatch can have one of three styles: `single`, `double`, or `triple`.
DTD:	`<!ATTLIST draw:hatch draw:style "(single|double|triple)" #IMPLIED>`

Color

The color attribute specifies the color of the hatch lines.

XML Code:	`draw:color`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:color %color; #IMPLIED>`

Distance

The distance attribute specifies the distance between two hatch lines.

XML Code:	`draw:distance`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:distance %length; #IMPLIED>`

Angle

The angle attribute specified the rotation angle of the hatch lines.

XML Code:	`draw:rotation`
Rules:	
DTD:	`<!ATTLIST draw:hatch draw:rotation %angle; #IMPLIED>`

5.6.3 Image

This element specifies a link to a bitmap resource, for example, a .JPG file. This element follows the Xlink specification.

XML Code:	`<draw:fill-image>`
Rules:	These elements must be located inside the `<office:styles>` elements.

DTD:	`<!ELEMENT draw:fill-image EMPTY>`
	`<!ATTLIST draw:fill-image xlink:href %uri; #REQUIRED`
	`xlink:type (simple) #IMPLIED`
	`xlink:show (parsed) #IMPLIED`
	`xlink:actuate (onLoad) #IMPLIED>`

The attributes that you can associate with the fill image element are:

- Name
- Size

Name

This attribute uniquely identifies a fill image inside an <office:styles> element.

XML Code:	draw:name
Rules:	
DTD:	<!ATTLIST draw:fill-image draw:name %styleName; #REQUIRED>

Size

These optional attributes specify the size of the linked image.

XML Code:	svg:width svg:height
Rules:	These values are optional and are overridden by the physical size of the linked image resource. They can be used to get the size of an image before it is loaded.
DTD:	<!ATTLIST draw:fill-image svg:width %length #IMPLIED svg:height %length #IMPLIED>

5.6.4 Transparency Gradient

To specify a transparency gradient for a graphic object, you can define a transparency that works in a similar fashion to a gradient, except that the transparency is interpolated instead of the color.

XML Code:	<draw:transparency>
Rules:	
DTD:	<!ENTITY % gradient-style "(linear|axial|radial|ellipsoid| square|rectangular)"> <!ELEMENT draw:transparency EMPTY> <!ATTLIST draw:transparency draw:name %styleName; #REQUIRED draw:style %gradient-style; #REQUIRED draw:cx %coordinate; #IMPLIED draw:cy %coordinate; #IMPLIED draw:gradient-angle %angle; #IMPLIED draw:gradient-border %percentage; #IMPLIED>

The attributes that you can associate with the <draw:transparency> element are:

- Name
- Style
- Transparency center
- Transparency

- Angle
- Border

Name

This attribute is the same as the name attribute associated with the gradient element. See Section 5.6.1 for information.

Style

This attribute is the same as the style attribute associated with the gradient element. See Section 5.6.1 for information.

Transparency Center

This attribute is the same as the gradient center attribute associated with the gradient element. See Section 5.6.1 for information.

Transparency

The transparency interpolates between a start and an end value.

XML Code:	`draw:start` `draw:end`
Rules:	The values of these attributes are percentages where 0% is fully transparent and 100% is fully opaque.

DTD:	`<!ATTLIST draw:transparency`
	`draw:start %percentage; #IMPLIED`
	`draw:end %percentage; #IMPLIED>`

Angle

This angle rotates the axis at which the transparency values are interpolated. It is the same as the angle attribute associated with the gradient element. See Section 5.6.1 for more information.

Border

Depending on the style of the transparency, the border attribute specifies a percentage value which is used to scale a border which is only the start or end transparency used. This attribute is the same as the border attribute associated with the gradient element. See Section 5.6.1 for more information.

5.6.5 Marker

The marker element represents markers, which are used to draw polygons at the start and end points of strokes.

XML Code:	`<draw:marker>`
Rules:	These elements must be located inside the `<office:styles>` elements.

DTD:	`<!ELEMENT draw:marker svg:path*>`
	`<!ATTLIST draw:marker`
	`draw:name %styleName; #IMPLIED`
	`%draw-viewbox;`
	`svg:d %PathData; #REQUIRED>`

Implementation limitation:	Currently, markers only store the marker geometry.

See Sections 5.3.4 and 5.3.14 for information on the Path Data and ViewBox attributes that you can associate with the `<draw:marker>` element.

5.6.6 Dash

The dash element represents a dash style that can be used to render strokes of shapes.

XML Code:	`<draw:stroke-dash>`
Rules:	These elements must be located inside the `<office:styles>` elements.

DTD:	`<!ELEMENT draw:stroke-dash EMPTY>`
	`draw:name %styleName; #REQUIRED`
	`draw:style (rect|round) #IMPLIED`
	`draw:dots1 %integer; #IMPLIED`
	`draw:dots1-length %length; #IMPLIED`
	`draw:dots2 %integer; #IMPLIED`
	`draw:dots2-length %length; #IMPLIED`
	`draw:distance %length; #IMPLIED>`

The attributes that you can associate with the `<draw:stroke-dash>` element are:

- Name
- Style
- Dots
- Distance

Name

This attribute uniquely identifies a dash inside an `<office:styles>` element.

XML Code:	`draw:name`
Rules:	
DTD:	`<!ATTLIST draw:stroke-dash draw:name %styleName; #REQUIRED>`

Style

This attribute specifies whether the points of a dash are round or rectanglular.

XML Code:	`draw:style`
Rules:	
DTD:	`<!ATTLIST draw:stroke-dash draw:style (rect|round) #IMPLIED>`

Dots

These attributes define a repeating sequence of dots that are used to render the dash. There are two sets of attributes which are used in an alternating sequence. For each sequence, you can specify the number of dots to draw and the length of each dot.

XML Code:	`draw:dots1`
	`draw:dots1-length`
	`draw:dots2`
	`draw:dots2-length`

Rules:	

DTD:	`<!ATTLIST draw:stroke-dash`
	`draw:dots1 %integer; #IMPLIED`
	`draw:dots1-length %length; #IMPLIED`
	`draw:dots2 %integer; #IMPLIED`
	`draw:dots2-length %length; #IMPLIED>`

Distance

The `draw:distance` attribute specifies the distance between the dots of a dash.

XML Code:	`draw:distance`
Rules:	
DTD:	`<!ATTLIST draw:stroke-dash draw:distance %length; #IMPLIED>`

5.7 Stroke Properties

You use the following **stroke properties** to define drawing object line characteristics in all OpenOffice.org **documents:**

- Style
- Dash
- Width
- Color
- Start marker
- End marker
- Start marker width
- End marker width
- Start marker center

- End marker center
- Transparency
- Joint

5.7.1 Style

This attribute specifies the style of the stroke on the current object.

XML Code:	`draw:stroke`
Rules:	
DTD:	`<!ATTLIST style:properties draw:stroke (none|dash|solid) #IMPLIED>`

5.7.2 Dash

This attribute specifies the dash style that is used for the stroke.

XML Code:	`svg:stroke-dash`
Rules:	
DTD:	`<!ATTLIST style:properties draw:stroke-dash %styleName; #IMPLIED>`

5.7.3 Width

This attribute specifies the width of the stroke on the current object in either units of length or as a percentage.

XML Code:	`svg:stroke-width`
Rules:	
DTD:	`<!ATTLIST style:properties svg:stroke-width %length; #IMPLIED>`

5.7.4 Color

This attribute specifies the color of the stroke on the current object.

XML Code:	`svg:stroke-color`
Rules:	
DTD:	`<!ATTLIST style:properties svg:stroke-color %color; #IMPLIED>`

5.7.5 Start Marker

This attribute specifies a line start marker, which is a path that can be connected to the start of a stroke.

XML Code:	`draw:marker-start`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-start %styleName; #IMPLIED>`

5.7.6 End Marker

This attribute specifies a stroke end marker, which is a path that can be connected to the end of a stroke.

XML Code:	`draw:marker-end`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-end %styleName; #IMPLIED>`

5.7.7 Start Marker Width

This attribute specifies the width of the marker at the start of the stroke.

XML Code:	`draw:marker-start-width`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-start-width %length; #IMPLIED>`

5.7.8 End Marker Width

This attribute specifies the width of the marker at the end of thestroke.

XML Code:	`draw:marker-end-width`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-end-width %length; #IMPLIED>`

5.7.9 Start Marker Center

This attribute specifies whether or not a start marker is centered at the start of a stroke.

XML Code:	`draw:marker-start-center`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-start-center % boolean; #IMPLIED>`

5.7.10 End Marker Center

This attribute specifies whether or not an end marker is centered at the end of a stroke.

XML Code:	`draw:marker-end-center`
Rules:	
DTD:	`<!ATTLIST style:properties draw:marker-end-center %boolean; #IMPLIED>`

5.7.11 Opacity

This attribute specifies the opacity of a stroke.

XML Code:	`svg:stroke-opacity`
Rules:	The value of this attribute can be a number between 0 (fully transparent) and 1 (fully opaque) or in a percentage.
DTD:	`<!ATTLIST style:properties svg:stroke-opacity (% floatOrPercentage; #IMPLIED>`

5.7.12 Joint

This attribute specifies the shape at the corners of paths or other vector shapes, when they are stroked.

XML Code:	`svg:stroke-linejoin`
Rules:	
DTD:	`<!ATTLIST style:properties svg:stroke-linejoin (miter|round| bevel|middle|none|inherit)#IMPLIED>`

5.8 Fill Properties

The fill properties used in OpenOffice.org **Draw and OpenOffice.org Impress are as follows:**

- Style
- Color
- Gradient
- Gradient step count
- Hatch
- Solid hatch
- Bitmap
- Transparency

5.8.1 Style

This attribute specifies the fill style for a graphic object. Graphic objects that are not closed, such as a path without a closepath at the end, will not be filled. The fill operation does not automatically close all open subpaths by connecting the last point of the subpath with the first point of the subpath before painting the fill.

XML Code:	`draw:fill`
Rules:	
DTD:	`<!ATTLIST style:properties draw:fill (none|solid|bitmap|gradient|hatch) #IMPLIED>`

5.8.2 Color

This attribute specifies the color of the fill for a graphic object.

XML Code:	`draw:fill-color`
Rules:	
DTD:	`<!ATTLIST style:properties draw:fill-color %color; #IMPLIED>`

5.8.3 Gradient

This attribute specifies a gradient style that is used for filling graphic objects.

XML Code:	`draw:fill-gradient-name`
Rules:	
DTD:	`<!ATTLIST style:properties draw:fill-gradient-name %styleName; #IMPLIED>`

5.8.4 Gradient Step Count

If a gradient is used for filling, you can set the gradient step count of the color interpolation to be a fixed value.

XML Code:	`draw:gradient-step-count`
Rules:	By default, the step count is automatically calculated based on the size and resolution of the filled area.
DTD:	`<!ATTLIST style:properties draw:gradient-step-count (auto|% value;) #IMPLIED>`

5.8.5 Hatch

This attribute specifies a hatch style that is used for filling.

XML Code:	`draw:fill-hatch-name`
Rules:	

XML Code:	draw:fill-hatch-name
DTD:	<!ATTLIST style:properties draw:fill-hatch-name %styleName; #IMPLIED>

5.8.6 Solid Hatch

This attribute specifies whether the background of a hatch filling is solid or transparent.

XML Code:	draw:fill-hatch-solid
Rules:	
DTD:	<!ATTLIST style:properties draw:fill-hatch-solid %boolean; #IMPLIED>

5.8.7 Bitmap

The following attributes are used when an area is to be filled with a bitmap.

Image

The fill image attribute specifies a URL that links to a <draw:fill-image> element.

XML Code:	draw:fill-image-name
Rules:	
DTD:	<!ATTLIST style:properties draw:fill-image-name %styleName; #IMPLIED>

Rendering Style

A bitmap image can either be rendered in the given size, stretched to the filled area, or tiled over the area. The style repeat attribute specifies how the bitmap image should be treated.

XML Code:	style:repeat
Rules:	The value of this attribute can be no-repeat, repeat, or stretch.
DTD:	<!ATTLIST style:properties style:repeat (no-repeat|repeat| stretch) #IMPLIED>

Size

These optional size attributes can be used to override the logical size of the source image data.

XML Code:	draw:fill-image-width and draw:fill-image-height
Rules:	If the value of the style:repeat attribute is stretch, these attributes are ignored.
DTD:	<!ATTLIST style:properties draw:fill-image-width %lengthOrPercentage; #IMPLIED draw:fill-image-height %lengthOrPercentage; #IMPLIED>

Tile Reference Point

These reference point attributes specify the point inside the source image that is used as the top left starting point for tiling.

XML Code:	`draw:refX` and `draw:refY`
Rules:	These attributes are only interpreted if the value of the current `style:repeat` attribute is `repeat`.
DTD:	`<!ATTLIST style:properties draw:refX %percentage; #IMPLIED` `draw:refY %percentage; #IMPLIED>`

Tile Translation

This attribute defines the translation of each tile in relation to the previous tile.

XML Code:	`draw:tile-repeat-offset`
Rules:	This attribute is only interpreted if the value of the current `style:repeat` attribute is `tiled`. The value of this attribute is a percentage value representing the tiles repeat offset relative to the tiles height or width, followed by either the word `horizontal` or `vertical`.
DTD:	`<!ENTITY % tile-repeat-offset "CDATA">` `<!ATTLIST style:properties draw:tile-repeat-offset %tile-repeat-offset; #IMPLIED>`

Example: Tile translation

```
<style:properties draw:tile-repeat-offset="50% horizontal"/>
```

5.8.8 Transparency

The fill area of a graphic object can either have none, linear, or gradient transparency. None and linear transparency is selected using the `draw:transparency` attribute, while gradient transparency is selected using the `draw:transparency-name` attribute.

None and Linear Transparency

The `draw:transparency` attribute disables transparency or sets a linear transparency for the fill area of a graphic object.

XML Code:	`draw:transparency`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:transparency %transparency; #IMPLIED>`

Gradient Transparency

The `draw:transparency-name` attribute specifies a transparency gradient that defines the transparency for the fill area of a graphic object. When applying a transparency gradient, the transparency is interpolated as defined in the referenced transparency gradient style. This fill style is rendered independently from other fill

styles like gradient, image, and hatch.

XML Code:	`draw:transparency-name`
Rules:	The value of this attribute overides the `draw:transparency` attribute.
DTD:	`<!ATTLIST style:properties draw:transparency-name % styleName; #IMPLIED>`

5.9 Text Animation Properties

Drawing objects that contain text and text frames can have optional animation properties. These properties always animate the complete text of a drawing object or text frame. The following attributes define the text animation:

- Animation
- Animation direction
- Animation start inside
- Animation stop inside
- Animation repeat
- Animation delay

Animation

This attribute specifies the type of animation that is used for the text.

XML Code:	`text:animation`
Rules:	The value of this attribute can be one of the following: • `none`, disables the text animation. • `scroll`, scrolls the text from one side to another. • `alternate`, scrolls the text from one side to another and back. • `slide`, scrolls the text from one side to the original text position and stops there.
DTD:	`<!ATTLIST style:properties text:animation (none|scroll|alternate|slide) #IMPLIED>`

Animation Direction

This attribute specifies the scroll direction of animated text.

XML Code:	`text:animation-direction`
Rules:	
DTD:	`<!ATTLIST style:properties text:animation-direction (left|right|up|down) #IMPLIED>`

Animation Start Inside

This attribute specifies whether or not text animation starts at the original position of the text.

XML Code:	`text:animation-start-inside`
Rules:	
DTD:	`<!ATTLIST style:properties text:animation-start-inside %boolean; #IMPLIED>`

Animation Stop Inside

This attribute specifies whether or not text animation stops at the original position of the text.

XML Code:	`text:animation-stop-inside`
Rules:	
DTD:	`<!ATTLIST style:properties text:animation-stop-inside %boolean; #IMPLIED>`

Animation Repeat

This attribute specifies the number of times the animation is repeated. If the value of the attribute is `0`, the animation is repeated indefinitely.

XML Code:	`text:animation-repeat`
Rules:	
DTD:	`<!ATTLIST style:properties text:animation-repeat %integer; #IMPLIED>`

Animation Delay

This attribute specifies a delay before the animation is started.

XML Code:	`text:animation-delay`
Rules:	To conform with ISO 8601, the format of the value of this attribute is `PnYnMnDTnHnMnS`. See Section 5.5.3.2 of ISO 8601 for more detailed information on this time format. See the Preface for a pointer to ISO 8601.
DTD:	`<!ATTLIST style:properties text:animation-delay %timeDuration; #IMPLIED>`

5.10 Text Properties

5.10.1 Auto Grow Width and Height

These attributes specify whether or not to automatically increase the width and height of the text if text is added to a drawing object.

XML Code:	`draw:auto-grow-width` `draw:auto-grow-height`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`draw:auto-grow-width %boolean; #IMPLIED`
	`draw:auto-grow-height %boolean; #IMPLIED>`

5.10.2 Fit To Size

This attribute specifies whether or not to stretch the text content of a drawing object to fill the entire object. If the value of the attribute is `true`, the text content is stretched.

XML Code:	`draw:fit-to-size`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:fit-to-size %boolean #IMPLIED>`

5.10.3 Text Area Vertical Align

This attribute specifies the vertical alignment of the text area inside a shape.

XML Code:	`draw:textarea-vertical-align`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`draw:textarea-vertical-align`
	`(top|middle|bottom|justify) #IMPLIED>`

5.10.4 Text Area Horizontal Align

This attribute specifies the horizontal alignment of the text area inside a shape.

XML Code:	`draw:textarea-horizontal-align`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`draw:textarea-horizontal-align`
	`(left|center|right|justify) #IMPLIED>`

5.11 Graphic Properties

5.11.1 Color Mode

The color mode style affects the output of colors from a source bitmap or raster graphic.

XML Code:	`draw:color-mode`
Rules:	
DTD:	`<!ENTITY % color-mode ; (greyscale|mono|watermark|standard)>` `<!ATTLIST style:properties draw:color-mode %color-mode; #IMPLIED>`

5.11.2 Color Inversion

The color inversion attribute specifies whether or not the colors in the graphic shape should be inverted.

XML Code:	`draw:color-inversion`
Rules:	
DTD:	`<!ATTLIST style:properties draw:color-inversion %boolean; #IMPLIED>`

5.11.3 Adjust Luminance

The luminance attribute specifies a signed percentage value that affects the output luminance of a bitmap or raster graphic.

XML Code:	`draw:luminance`
Rules:	
DTD:	`<!ATTLIST style:properties draw:luminance %signed-percent; #IMPLIED>`

5.11.4 Adjust Contrast

The contrast attribute specifies a signed percentage value that affects the output contrast of a bitmap or raster graphic.

XML Code:	`draw:contrast`
Rules:	
DTD:	`<!ATTLIST style:properties draw:contrast %signed-percent; #IMPLIED>`

5.11.5 Adjust Gamma

The gamma attribute specifies a value that affects the output gamma of a bitmap or raster graphic.

XML Code:	`draw:gamma`
Rules:	
DTD:	`<!ATTLIST style:properties draw:gamma %signed-percent; #IMPLIED>`

5.11.6 Adjust Red

The red attribute specifies a signed percentage value that affects the output of the red color space of a bitmap or raster graphic.

XML Code:	`draw:red`
Rules:	
DTD:	`<!ATTLIST style:properties draw:red %signed-percent; #IMPLIED>`

5.11.7 Adjust Green

The green attribute specifies a signed percentage value that affects the output of the green color space of a bitmap or raster graphic.

XML Code:	`draw:green`
Rules:	
DTD:	`<!ATTLIST style:properties draw:green %signed-percent; #IMPLIED>`

5.11.8 Adjust Blue

The blue attribute specifies a signed percentage value that affects the output of the blue color space of a bitmap or raster graphic.

XML Code:	`draw:blue`
Rules:	
DTD:	`<!ATTLIST style:properties draw:blue %signed-percent; #IMPLIED>`

5.11.9 Adjust Transparency

This attribute adjusts the transparency of an image.

XML Code:	`draw:transparency`
Rules:	The value of this attribute can be an integer in the range from -100 to 100.
DTD:	`<ENTITY % transparency "CDATA">` `<!ATTLIST style:properties draw:transparency %transparency; #IMPLIED>`

5.12 Shadow Properties

Most drawing objects can have a shadow. The following attributes specify how the shadow is rendered.

Shadow

This attribute enables or disables the visibility of a shadow.

XML Code:	`draw:shadow`
Rules:	
DTD:	`<!ATTLIST style:properties draw:shadow (visible|hidden) #IMPLIED>`

Offset

To render a shadow, a copy of the shape is rendered in the single shadow color behind the shape. The offset attributes specify the offset between the top left edge of the shape and the top left edge of the border .

XML Code:	`draw:shadow-distance-x` and `draw:shadow-distance-y`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`draw:shadow-offset-x %length; #IMPLIED`
	`draw:shadow-offset-y %length; #IMPLIED>`

Color

The shadow color attribute specifies the color in which the shadow is rendered.

XML Code:	`draw:shadow-color`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:shadow-color %color; #IMPLIED>`

Transparency

The shadow transparency attribute specifies the transparency in which the shadow is rendered.

XML Code:	`draw:shadow-transparency`
Rules:	The value of this attribute is a percentage value.
DTD:	`<!ATTLIST style:properties` `draw:shadow-transparency %percent; #IMPLIED>`

5.13 Connector Properties

Start Line Spacing

The start line spacing attributes modify the line spacing at the start point of a connector shape.

XML Code:	`draw:start-line-spacing-horizontal` `draw:start-line-spacing-vertical`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`draw:start-line-spacing-horizontal %distance; #IMPLIED`
	`draw:start-line-spacing-vertical %distance; #IMPLIED>`

End Line Spacing

The end line spacing values modify the line spacing at the end point of a connector shape.

XML Code:	`draw:end-line-spacing-horizontal` `draw:end-line-spacing-vertical`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`draw:end-line-spacing-horizontal %distance; #IMPLIED`
	`draw:end-line-spacing-vertical %distance; #IMPLIED>`

5.14 Measure Properties

Line Distance

The `draw:line-distance` attribute specifies the distance from the reference points to the measure line.

XML Code:	`draw:line-distance`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:line-distance %distance; #IMPLIED>`

Guide Overhang

The guides are the two lines from the reference points to the measure line. The `draw:guide-overhang` attribute specifies the distance that the guides are drawn after they cross the measure line.

XML Code:	`draw:guide-overhang`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:guide-overhang %distance; #IMPLIED>`

Guide Distance

The `draw:guide-distance` attribute specifies the distance between the reference points and the measure line where the drawing of the guides starts.

XML Code:	`draw:guide-distance`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:guide-distance %distance; #IMPLIED>`

Start Guide

The `draw:start-guide` attribute specifies the offset to the start of the guide from the first reference point to the measure line.

XML Code:	`draw:start-guide`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:start-guide %distance; #IMPLIED>`

End Guide

The `draw:end-guide` attribute specifies the offset to the start of the guide from the last reference point to the measure line.

XML Code:	`draw:end-guide`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:end-guide %distance; #IMPLIED>`

Placing

The `draw:placing` attribute specifies whether the measure line is rendered below or above the edge defined by the two reference points.

XML Code:	`draw:placing`
Rules:	The value of this attribute can be `below` or `above`.
DTD:	`<!ATTLIST style:properties` `draw:placing (below|above) #IMPLIED>`

Parallel

The `draw:parallel` attributes toggles if the text is parallel to the measure line or perpendicular.

XML Code:	`draw:parallel`
Rules:	
DTD:	`<!ATTLIST style:properties` `  draw:parallel %boolean; #IMPLIED>`

Text Alignment

These attributes determine the text alignment relative to the measure line.

XML Code:	`draw:measure-align` `draw:measure-vertical-align`
Rules:	If value of this attribute is `automatic`, the application chooses the best position.

DTD:	`<!ATTLIST style:properties`
	`  draw:measure-align`
	`  (automatic|left-outside|inside|right-outside)`
	`  #IMPLIED`
	`  draw:measure-vertical-align`
	`  (automatic|above|below|center) #IMPLIED>`

Unit

This attribute specifies the unit used in the textual presentation of a measure shape.

XML Code:	`draw:unit`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`  draw:unit`
	`  (automatic|mm|cm|m|km|pt|pc|inch|ft|mi) #IMPLIED>`

Show Unit

This attribute toggles the display of the unit in the textual presentation of a measure shape.

XML Code:	`draw:show-unit`
Rules:	
DTD:	`<!ATTLIST style:properties` `  show-unit %boolean; #IMPLIED>`

Decimal Places

This attribute specifies the number of decimal places that are used for the measure text.

XML Code:	`draw:decimal-places`
Rules:	
DTD:	`<!ATTLIST style:properties decimal-places %nonNegativeLength; #IMPLIED>`

5.15 Caption Properties

The following attributes can be used in the styles for caption shapes.

- Type
- Angle type
- Angle
- Gap
- Escape direction
- Escape
- Line length
- Fit line length

Type

This attribute specifies the geometry of the line of a caption.

XML Code:	`draw:caption-type`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-type (straight-line|angled-line|angled-connector-line) #IMPLIED>`

Angle Type

This attribute specifies if the escape angle of the line of a caption is fixed or free. If this is set to `free` the application can choose the best possible angle.

XML Code:	`caption-angle-type`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-angle-type (fixed|free) #IMPLIED>`

Angle

This attribute specifies the escape angle of the line of a caption.

XML Code:	`draw:caption-angle`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-angle %nonNegativeInteger; #IMPLIED>`

Gap

This attribute specifies the distance between the text area of the caption and the start of the line.

XML Code:	`draw:caption-gap`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-gap %distance; #IMPLIED>`

Escape Direction

This attribute specifies the escape direction for the line of a caption.

XML Code:	`draw:caption-escape-direction`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-escape-direction (horizontal|vertical|auto) #IMPLIED>`

Escape

This attribute specifies the escape distance for the line of a caption.

XML Code:	`draw:caption-escape`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-escape %lengthOrPercentage; #IMPLIED>`

Line Length

This attribute specifies the length of the caption line.

XML Code:	`draw:caption-line-length`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-line-length %distance; #IMPLIED>`

Fit Line Length

If this attribute is true, the application determines the best possible length for the caption line.

XML Code:	`draw:caption-fit-line-length`
Rules:	
DTD:	`<!ATTLIST style:properties draw:caption-fit-line-length %boolean; #IMPLIED>`

5.16 3D Geometry Properties

Horizontal Segments

If the geometry of a 3D object is generated during run-time, you can choose the number of horizontal segments that are used to generate the geometry using the `dr3d:horizontal-segments` attribute.

XML Code:	`dr3d:horizontal-segments`
Rules:	Values between 2 and 256 are currently supported.
DTD:	`<!ATTLIST style:properties dr3d:horizontal-segements %nonNegativeNumber; #IMPLIED>`

Vertical Segments

If the geometry of a 3D object is generated during run-time, you can choose the number of vertical segments that are used to generate the geometry.

XML Code:	`dr3d:vertical-segments`
Rules:	Values between 2 and 256 are currently supported.
DTD:	`<!ATTLIST style:properties dr3d:vertical-segements %nonNegativeNumber; #IMPLIED>`

Edge Rounding

If the geometry of a 3D object is generated during run-time, you can choose the size of an area at the edge of the geometry that is used for rounding the edges.

XML Code:	`dr3d:edge-rounding`
Rules:	
DTD:	`<!ATTLIST style:properties dr3d:edge-rounding %percentage; #IMPLIED>`

Edge Rounding Mode

The `dr3d:edge-rounding-mode` attribute specifies how to generate rounded edges.

XML Code:	`dr3d:edge-rounding-mode`
Rules:	The value of this attribute can be `correct` or `number`. If the value is `correct`, a mathematically correct method is used. If the value is `attractive`, a method which preserves the visual appearance of the text is used.
DTD:	`<!ATTLIST style:properties dr3d:edge-rounding-mode (correct|attractive) #IMPLIED>`

Back Scale

The `dr3d:back-scale` attribute specifies the proportion of the background geometry for lathe and extrude objects.

XML Code:	`dr3d:back-scale`
Rules:	
DTD:	`<!ATTLIST style:properties dr3d:back-scale %percentage; #IMPLIED>`

Depth

The `dr3d:depth` attribute specifies the extrusion depth for extrude objects.

XML Code:	`dr3d:depth`
Rules:	
DTD:	`<!ATTLIST style:properties dr3d:depth %length; #IMPLIED>`

Backface Culling

The `dr3d:backface-culling` attribute enables or disables backface culling.

XML Code:	`dr3d:backface-culling`
Rules:	
DTD:	`<!ATTLIST style:properties dr3d:backface-culling (enabled|disabled) #IMPLIED>`

5.17 3D Lighting Properties

Mode

The `dr3d:lighting-mode` attribute determines the lighting algorithm used to render the corresponding 3D object.

XML Code:	dr3d:lighting-mode
Rules:	The value of this attribute can be standard or double-sided. If the value is double-sided, the reverse sides of the objects are also lighted.
DTD:	<!ATTLIST style:properties dr3d:lighting-mode (standard|double-sided) #IMPLIED>

Normals Kind

The dr3d:depth attribute specifies how the normal settings for lighting are generated.

XML Code:	dr3d:depth
Rules:	
DTD:	<!ATTLIST style:properties dr3d:normals-kind (object|flat|sphere) #IMPLIED>

Normals Direction

The dr3d:normals-direction attribute allows you to inverse the generated normal lighting settings.

XML Code:	dr3d:normals-direction
Rules:	
DTD:	<!ATTLIST style:properties dr3d:normals-direction (normal|inverse) #IMPLIED>

5.18 3D Texture Properties

Generation Mode

These attributes specify how the texture coordinates are generated.

XML Code:	dr3d:texture-generation-mode-x dr3d:texture-generation-mode-y
Rules:	
DTD:	<!ATTLIST style:properties dr3d:texture-generation-mode-x (object|parallel|sphere) #IMPLIED dr3d:texture-generation-mode-y (object|parallel|sphere) #IMPLIED>

Kind

The dr3d:texture-kind attribute allows you to select whether the texture changes the luminance, intensity, or color of the shape.

XML Code:	`dr3d:texture-kind`
Rules:	
DTD:	`<!ATTLIST style:properties` `dr3d:texture-kind (luminance|intesity|color) #IMPLIED>`

Filter

The `dr3d:texture-filter` attribute allows you to enable or disable texture filtering.

XML Code:	`dr3d:texture-filter`
Rules:	
DTD:	`<!ATTLIST style:properties dr3d:texture-filter` `(enabled|disabled) #IMPLIED>`

Mode

The `dr3d:normals-direction` attribute allows you to specify how the texture is modulated.

XML Code:	`dr3d:normals-direction`
Rules:	
DTD:	`<!ATTLIST style:properties` `dr3d:normals-direction (normal|inverse) #IMPLIED>`

5.19 3D Material Properties

Colors

These attributes specify the four colors that define a material.

XML Code:	`dr3d:ambient-color`
	`dr3d:emissive-color`
	`dr3d:specular-color`
	`dr3d:diffuse-color`

Rules:	

DTD:	`<!ATTLIST style:properties`
	`dr3d:ambient-color %color; #IMPLIED`
	`dr3d:emissive-color %color; #IMPLIED`
	`dr3d:specular-color %color; #IMPLIED`
	`dr3d:diffuse-color %color; #IMPLIED>`

Shininess

The `dr3d:shininess` attribute specifies the shine of the used material.

XML Code:	dr3d:shininess
Rules:	
DTD:	<!ATTLIST style:properties dr3d:shininess %percentage; #IMPLIED>

5.20 3D Shadow Properties

The dr3d:shadow attribute enables or disables a three-dimensional shadow for a three-demensional object.

XML Code:	dr3d:shadow
Rules:	
DTD:	<!ATTLIST style:properties dr3d:shadow (visible|hidden) #IMPLIED>

5.21 Layer Sets

Layer sets define a set of layers.

XML Code:	<draw:layer-set>
Rules:	Currently there can only be one layer set for a Draw or Impress document.
DTD:	<!ELEMENT draw:layer-set (draw:layer*)>

5.21.1 Layer

Each layer is defined and referenced by a name. Each drawing object inside a drawing or impress document can be assigned to a layer. Layers virtually group the shapes. Each shape that is assigned to a layer inherits the settings of the layer.

XML Code:	<draw:layer>
Rules:	
DTD:	<!ELEMENT draw:layer EMPTY> <!ATTLIST draw:layer draw:name %layerName; #REQUIRED>

5.22 Glue Points

Glue points are designated points on the area of a drawing object to which a connector shape can connect. Most drawing objects have four standard glue points at the four edges of the object. You can add your own custom glue points to a drawing object by inserting one or more of the following elements inside a drawing object element.

5.22.1 Glue Point

The glue point element creates a single user-defined glue point if placed inside a drawing object element, for example, a <draw:rectangle> element.

XML Code:	`<draw:glue-point>`
Rules:	

DTD:	`<!ELEMENT draw:glue-point EMPTY>`
	`<!ATTLIST draw:glue-point`
	`draw:id %integer; #REQUIRED`
	`svg:x %coordinateOrPercentage; #REQUIRED`
	`svg:y %coordinateOrPercentage; #REQUIRED>`

Align

The `draw:align` attribute specifies the alignment behavior of the glue point if the drawing object is resized.

XML Code:	`draw:align`
Rules:	
DTD:	`<!ATTLIST draw:glue-point draw:align (top-left|top|top-right|left|center|right|bottom-left|bottom-right) #IMPLIED>`

Escape Direction

The `draw:escape-direction` attribute specifies the direction in which the connection line escapes from the drawing object if a connector connects to the glue point.

XML Code:	`draw:escape-direction`
Rules:	
DTD:	`<!ATTLIST draw:glue-point draw:escape-direction (auto|left|right|up|down|horizontal|vertical) #IMPLIED>`

5.23 Presentation Page Layouts

A presentation page layout is a container for placeholders, which define a set of empty presentation objects, for example, a title or outline. These placeholders are used as templates for creating new presentation objects and to mark the size and position of an object if the presentation page layout of a drawing page is changed.

XML Code:	`<style:presentation-page-layout>`
Rules:	

DTD:	`<!ELEMENT style:presentation-page-layout presentation:placeholder*>`
	`<!ATTLIST style:presentation-page-layout`
	`style:name %styleName; #REQUIRED>`

Note:	For presentations only.

5.23.1 Presentation Placeholder

The presentation placeholder element specifies a placeholder for presentation objects, for example, a title or outline.

XML Code:	`<presentation:placeholder>`
Rules:	

DTD:	`<!ENTITY % presentation-object "(title|outline|subtitle| text|graphic|object|chart|orgchart|table|page|notes| handout)" >`
	`<!ELEMENT presentation:placeholder EMPTY>`
	`<!ATTLIST presentation:placeholder presentation:object % presentation-object; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:x % coordinateOrPercentage; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:y % coordinateOrPercentage; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:width % lengthOrPercentage; #REQUIRED>`
	`<!ATTLIST presentation:placeholder svg:height % lengthOrPercentage; #REQUIRED>`

5.24 Presentation Page Attributes

You can add transition, fade, and audio effects to each presentation page using the following optional presentation attributes:

- Transition Type
- Transition Style
- Transition Speed
- Page Duration
- Page Visibility
- Sound
- Background Size
- Background Objects Visible
- Background Visible

The transition attributes are contained in the style element of the page.

5.24.1 Transition Type

You can set the mode of transition, for example manual, using the transitiontype attribute.

XML Code:	`presentation:transition-type`
Rules:	
DTD:	`<!ENTITY % transition-type ; (manual|automatic|semi-automatic)>` `<!ATTLIST style:properties presentation:transition-type % transition-type; "manual">`
Note:	For presentations only.

5.24.2 Transition Style

The `transitionstyle` attribute specifies the way that each presentation page replaces the previous presentation page, for example left-to-right replacement, or fading.

XML Code:	`presentation:transition-style`
Rules:	
DTD:	`<!ENTITY % page-transition "(none|fade-from-left|fade-from-top|fade-from-right|fade-from-bottom|fade-to-center|fade-from-center|move-from-left|move-from-top|move-from-right|move-from-bottom|roll-from-left|roll-from-right|roll-from-bottom|vertical-stripes|horizontal-stripes|clockwise|counterclockwise|fade-from-upperleft|fade-from-upperright|fade-from-lowerleft|fade-from-lowerright|close-vertical|close-horizontal|open-vertical|open-horizontal|spiralin-left|spiralin-right|spiralout-left|spiralout-right|dissolve|wavyline-from-left|wavyline-from-top|wavyline-from-right|wavyline-from-bottom|random|stretch-from-left|stretch-from-top|stretch-from-right|stretch-from-bottom|vertical-lines|horizontal-lines|move-from-upperleft|move-from-upperright|move-from-lowerright|move-from-lowerleft|uncover-to-left|uncover-to-upperleft|uncover-to-top|uncover-to-upperright|uncover-to-right|uncover-to-lowerright|uncover-to-bottom|uncover-to-lowerleft|vertical-checkerboard|horizontal-checkerboard)"> <!ATTLIST style:properties presentation:transition-style % page-transition; "none">
Note:	For presentations only.

5.24.3 Transition Speed

The `transition-speed` attribute controls the speed at which a presentation page is removed from display, and replaced by a new presentation page.

XML Code:	`presentation:transition-speed`
Rules:	
DTD:	`<!ENTITY % speed (slow|medium|fast)> <!ATTLIST style:properties presentation:transition-speed % speed; "medium">`
Note:	For presentations only.

5.24.4 Page Duration

The `page-duration` attribute controls the amount of time that the presentation page is displayed. T

XML Code:	`presentation:page-duration`
Rules:	
DTD:	`<!ATTLIST style:properties presentation:page-duration % timeDuration; #IMPLIED>`
Note:	For presentations only.

5.24.5 Page Visibility

You can mark a drawing page as hidden during a presentation by using the `visibility` attribute. A page marked with this attribute is only shown while editing the document but not during the presentation.

XML Code:	`presentation:visibility`
Rules:	
DTD:	`<!ENTITY %visibility (visible|hidden)>` `<!ATTLIST style:properties presentation:visibility (visible| hidden) "visible">`
Note:	For presentations only.

5.24.6 Sound

You can add sound effects to your presentation pages using the `sound` element.

XML Code:	`presentation:sound`
Rules:	

DTD:	`<!ELEMENT presentation:sound EMPTY>`
	`<!ATTLIST presentation:sound xlink:href %url; #IMPLIED`
	`xlink:type (simple) #FIXED "simple"`
	`xlink:show (embed) "embed"`
	`xlink:actuate (onLoad) "onLoad">`

Note:	For presentations only.

5.24.7 Background Size

This attribute specifies whether the background of a page is rendered on the full page or only inside the borders of the page.

XML Code:	`draw:background-size`
Rules:	
DTD:	`<!ATTLIST style:properties` `draw:background-size (full|border) #IMPLIED>`

5.24.8 Background Objects Visible

This attribute specifies whether or not to hide objects on the background of the master page when displaying the

presentation page.

XML Code:	`presentation:background-objects-visible`
Rules:	
DTD:	`<!ATTLIST style:properties presentation:background-objects-visible %boolean; #IMPLIED>`
Note:	For presentations only.

5.24.9 Background Visible

This attribute specifies whether or not to hide the background of the master page when displaying the presentation page.

XML Code:	`presentation:background-visible`
Rules:	
DTD:	`<!ATTLIST style:properties presentation:background-visible % boolean; #IMPLIED>`
Note:	For presentations only.

5.25 Presentation Settings

5.25.1 Presentation

The settings for a presentation are stored in a `<presentation:settings>` element inside an `<office:body>` element. These settings affect the behaviour if the document is displayed in a presentation.

XML Code:	`<presentation:settings>`
Rules:	
DTD:	`<!ELEMENT presentation:settings (presentation:show)*>`
Note:	For presentations only.

The attributes that you can associate with the `<presentation:settings>` element are:

- Start page
- Show
- Full screen
- Endless
- Pause
- Show logo
- Force manual
- Mouse visible

- Mouse as pen
- Start with navigator
- Allow animation
- Transition on click
- Stay on top

Start page

This attribute specifies the name of the page on which the presentation starts.

XML Code:	`presentation:start-page`
Rules:	If this attribute is set, it overides the `presentation:show` attribute.
DTD:	`<!ATTLIST presentation:settings` `presentation:start-page %styleName #IMPLIED>`
Note:	For presentations only.

Show

This attribute specifies the name of a show that is used for the presentation.

XML Code:	`presentation:show`
Rules:	If the `presentation:start-page` attribute is set, it overides the value of this attribute.
DTD:	`<!ATTLIST presentation:settings` `presentation:show %styleName #IMPLIED>`
Note:	For presentations only.

Full Screen

This attribute determines whether the presentation is displayed in full screen mode or in a window.

XML Code:	`presentation:full-screen`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `presentation:full-screen %boolean; "true">`
Note:	For presentations only.

Endless

This attribute switches indefinite repetition of a presentation on and off.

XML Code:	`presentation:endless`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `presentation:endless %boolean; "false">`
Note:	For presentations only.

Pause

If a presentation is repeated indefinitely, this attribute specifies a time duration for displaying a pause screen before the presentation is played again. If this attribute is not set or has a value of 0, a pause screen is not displayed in endless mode.

XML Code:	`presentation:pause`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `presentation:pause %timeDuration; #IMPLIED>`
Note:	For presentations only.

Show Logo

This attribute specifies whether or not a presentation application shows its logo on the pause screen.

XML Code:	`presentation:show-logo`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `presentation:show-logo %boolean; "false">`
Note:	For presentations only.

Force Manual

If set, this attribute overides the attribute `presentation:transition-type` and sets it to `manual`.

XML Code:	`presentation:force-manual`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `presentation:force-manual %boolean; "false">`
Note:	For presentations only.

Mouse Visible

This attribute specifies whether or not the mouse pointer is visible during a presentation.

XML Code:	`presentation:mouse-visible`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `  presentation:mouse-visible %boolean; "true">`
Note:	For presentations only.

Mouse As Pen

This attribute specifies if the mouse pointer is displayed as a pen or a pointer. If the mouse is displayed as a pen the user can draw sketches on the pages during a presentation.

XML Code:	`presentation:mouse-as-pen`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `  presentation:mouse-as-pen %boolean; "false">`
Note:	For presentations only.

Start With Navigator

This attribute specifies whether or not the navigator window is initially displayed during a presentation.

XML Code:	`presentation:start-with-navigator`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `  presentation:start-with-navigator %boolean; "false">`
Note:	For presentations only.

Animations

This attribute enables or disables the playback of bitmap animations during a presentation.

XML Code:	`presentation:animations`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `  presentation:animations (enabled|disabled) "enabled">`
Note:	For presentations only.

Transition On Click

This attribute enables or disables a manual transition by a mouse click on the slide during a presentation.

XML Code:	`presentation:transition-on-click`
Rules:	

DTD:	`<!ATTLIST presentation:settings`
	`presentation:transition-on-click (enabled|disabled)`
	`"enabled">`

Note:	For presentations only.

Stay On Top

If this attribute is set to `true`, the presentation window is displayed on top of other windows during a presentation.

XML Code:	`presentation:stay-on-top`
Rules:	
DTD:	`<!ATTLIST presentation:settings` `presentation:stay-on-top %boolean; "false">`
Note:	For presentations only.

5.25.2 Shows

A presentation document can contain one or more shows. A show enables you to customize the order in which the pages are displayed during a presentation. You can also omit pages from the presentation or repeat the pages during the presentation.

XML Code:	`<presentation:show>`
Rules:	This element is optional.
DTD:	`<!ELEMENT presentation:show EMPTY>`
Note:	For presentations only.

The attributes that you can associate with the `<presentation:show>` element are:

- Name
- Pages

Name

This attribute uniquely identifies a `<presentation:show>` element.

XML Code:	`presentation:name`
Rules:	
DTD:	`<!ATTLIST presentation:show` `presentation:name %styleName; #REQUIRED>`

Pages

This attribute contains a comma separated list of page names. The pages are displayed in the order in which they are listed during a presentation that uses this show. Pages can be included more than once.

XML Code:	`presentation:pages`
Rules:	
DTD:	`<!ATTLIST presentation:show` `presentation:pages CDATA #REQUIRED>`

5.26 Presentation Animations

In a presentation document, shapes can be animated. Each presentation page can have an optional `<presentation:animations>` element, which is a container for animation effects. The animation is executed when the page is displayed during a presentation.

XML Code:	`<presentation:animations>`
Rules:	

DTD:	`<!ELEMENT presentation:animations`
	`(presentation:show-shape|`
	`presentation:show-text|`
	`presentation:hide-shape|`
	`presentation:hide-text|`
	`presentation:dim|`
	`presentation:play)*>`

Implementation limitations:	The current file format specification is more fexible than the current software implementation. The current implementation of the OpenOffice.org Impress application only supports one show and one hide effect per shape with an additional show and hide text and one dim and sound effect.

5.26.1 Sound

You can include the `<presentation:sound>` element in any animation effect that supports sound. The sound file referenced by the Xlink attributes is played when the effect is executed.

XML Code:	`<presentation:sound>`
Rules:	

DTD:	`<!ELEMENT presentation:sound EMPTY>`
	`<!ATTLIST presentation:sound`
	`xlink:href %uriReference; #REQUIRED`
	`xlink:type (simple) #FIXED "simple"`
	`xlink:show (new|replace) #IMPLIED`
	`xlink:actuate (onRequest) "onRequest">`

The attributes that you can associate with the `<presentation:sound>` element are:

- Play full

Play Full

If the value of this attribute is `true`, the next effect starts after the sound is played. If the value of this attribute is `false`, the next effect starts when the current effect is finished.

XML Code:	`presentation:play-full`
Rules:	
DTD:	`<!ATTLIST presentation:sound presentation:play-full %boolean; #IMPLIED>`

5.26.2 Show Shape

The `<presentation:show-shape>` element makes a shape visible. If there is a `<presentation:show-shape>` element for one shape, this shape is automatically invisible before the effect is executed.

XML Code:	`<presentation:show-shape>`
Rules:	
DTD:	<!ELEMENT presentation:show-shape (presentation:sound)?>

The attributes that you can associate with the `<presentation:show-shape>` element are:

- Shape
- Effect
- Direction
- Speed
- Start Scale
- Path

Shape

This attribute specifies the shape of this effect using a shape ID.

XML Code:	`draw:shape-id`
Rules:	
DTD:	`<!ATTLIST presentation:show-shape draw:shape-id %shapeId; #REQUIRED>`

Effect

This attribute specifies the type of effect.

XML Code:	`presentation:effect`
Rules:	
DTD:	`<!ENTITY % presentationEffects "(none|fade|move|stripes| open|close|dissolve|wavyline|random|lines|laser|appear|hide| move-short|checkerboard|rotate|stretch)" > <!ATTLIST presentation:show-shape presentation:effect %presentationEffects; "none">`

Direction

This attribute specifies the direction of the effect. This is relevant for some effects only.

XML Code:	`presentation:direction`
Rules:	
DTD:	`<!ENTITY % presentationEffectDirections "(none|from-left| from-top|from-right|from-bottom|from-center|from-upper-left| from-upper-right|from-lower-left|from-lower-right|to-left| to-top|to-right|to-bottom|to-upper-left|to-upper-right|to-lower-right|to-lower-left|path|spiral-inward-left|spiral-inward-right|spiral-outward-left|spiral-outward-right| vertical|horizontal|to-center|clockwise|counter-clockwise)" > <!ATTLIST presentation:show-shape presentation:direction % presentationEffectDirections; "none">`

Speed

This attribute specifies the speed of the effect.

XML Code:	`presentation:speed`
Rules:	
DTD:	`<!ENTITY % presentationSpeeds "(slow|medium|fast)" > <!ATTLIST presentation:show-shape presentation:speed %presentationSpeeds; "medium">`

Start Scale

Some effects scale a shape during execution of the effect. This attribute specifies the start size of the shape as a percentage of its original size.

XML Code:	`presentation:start-scale`
Rules:	
DTD:	`<!ATTLIST presentation:show-shape presentation:start-scale %percentage; "100%">`

Path

This optional attribute applies to `move` effects. The attribute specifies the path to a polygon. The effect moves

along the lines of the specified polygon. The polygon is not visible during the presentation.

XML Code:	`presentation:path-id`
Rules:	
DTD:	`<!ATTLIST presentation:show-shape` `  presentation:path-id %shapeId; #IMPLIED>`

5.26.3 Show Text

This element makes the text of a shape visible. If there is a `<show-text>` element for one shape, the text of the shape is automatically invisible before the effect is executed.

XML Code:	`<presentation:show-text>`
Rules:	

DTD:	`<!ELEMENT presentation:show-text (presentation:sound)?>`
	`<!ATTLIST presentation:show-text`
	`  draw:shape-id %shapeId; #REQUIRED>`
	`  presentation:effect %presentationEffects; "none"`
	`  presentation:direction %`
	`presentationEffectDirections; "none"`
	`  presentation:speed %presentationSpeeds; "medium"`
	`  presentation:start-scale %percentage; "100%"`
	`  presentation:path-id %shapeId; #IMPLIED>`

The attributes that you can associate with the `<presentation:show-text>` element are:

- Shape, Effect, Direction, Speed, Start Scale, Path – see Section 5.26.2

5.26.4 Hide Shape

This element makes a shape invisible.

XML Code:	`<presentation:hide-shape>`
Rules:	

DTD:	`<!ELEMENT presentation:hide-shape (presentation:sound)?>`
	`<!ATTLIST presentation:hide-shape`
	`  draw:shape-id %shapeId; #REQUIRED`
	`  presentation:effect %presentationEffects; "none"`
	`  presentation:direction %`
	`presentationEffectDirections; "none"`
	`  presentation:speed %presentationSpeeds; "medium"`
	`  presentation:start-scale %percentage; "100%"`
	`  presentation:path-id %shapeId; #IMPLIED>`

The attributes that you can associate with the `<presentation:hide-shape>` element are:

- Shape, Effect, Direction, Speed, Start Scale, Path – see Section 5.26.2

5.26.5 Hide Text

This element makes the text of a shape invisible.

XML Code:	`<presentation:hide-text>`
Rules:	

DTD:	`<!ELEMENT presentation:hide-text (presentation:sound)?>`
	`<!ATTLIST presentation:hide-text`
	`draw:shape-id %shapeId; #REQUIRED`
	`presentation:effect %presentationEffects; "none"`
	`presentation:direction %`
	`presentationEffectDirections; "none"`
	`presentation:speed %presentationSpeeds; "medium"`
	`presentation:start-scale %percentage; "100%"`
	`presentation:path-id %shapeId; #IMPLIED>`

The attributes that you can associate with the `<presentation:hide-text>` element are:

- Shape, Effect, Direction, Speed, Start Scale, Path – see Section 5.26.2

5.26.6 Dim

This element fills a shape in a single color.

XML Code:	`<presentation:dim>`
Rules:	

DTD:	`<!ELEMENT presentation:dim (presentation:sound)?>`
	`<!ATTLIST presentation:dim draw:shape-id %shapeId;`
	`#REQUIRED>`

The attributes that you can associate with the `<presentation:dim>` element are:

- Shape – see Section 5.26.2
- Color

Color

This attribute specifies the color that is used to fill the shape when the shape is dimmed.

XML Code:	`draw:color`
Rules:	
DTD:	`<!ATTLIST presentation:dim draw:color %color; #REQUIRED>`

5.26.7 Play

This element starts the animation of a shape that supports animation.

XML Code:	`<presentation:play>`
Rules:	

DTD:	`<!ELEMENT presentation:play EMPTY>`
	`<!ATTLIST presentation:play`
	`draw:shape-id %shapeId; #REQUIRED`
	`presentation:speed %presentationSpeeds; "medium">`

The attributes that you can associate with the `<presentation:play>` element are:

- Shape ID and Speed – see Section 5.26.2

CHAPTER 6

Form Content

This chapter describes the OpenOffice.org XML representation of forms and controls. This chapter contains the following sections:

- Forms
- Controls
- Common Form and Control Attributes
- Common Control Attributes
- Events
- Properties

6.1 Forms

A form is a container for user interface controls which a user interacts with. For example, buttons, text boxes, check boxes, and drop-down lists are user interface controls that can be contained in a form. In the OpenOffice.org XML file format, the following basic rules apply to user interface controls and forms:

- You must locate every control in a form.
- A control that is not hidden must contain text or have an absolute position. Use a frame containing a reference to the control to represent the position of the control.
- You can nest forms.
- Forms are not connected with the text flow and layout of a document. This does not apply to controls.
- Forms can be data-aware. The controls reflect the content of a database.

Forms define rules for the following form behaviour:

- Submitting the form, which is similar to HTML 4.01. Note: You can not submit nested forms and controls that can not be converted to HTML.
- Connecting to a data source. When this happens, the controls in a form become data-aware.

Forms are contained in the `<office:forms>` section of an OpenOffice.org **XML document.**

6.1.1 Form

The `<form:form>` element represents a user interface form and defines the contents and properties of the form.

XML Code:	`<form:form>`
Rules:	This element is contained in one of the following elements: • The `<office:forms>` element of a document. • Another `<form:form>` element. The element contains the following items: • The controls and sub-forms in the form. • A `<form:properties>` element that defines the properties of the form. • A `<office:events>` element that contains the events for the form.

DTD:	`<!ELEMENT form:form (form:properties?, office:events?, (form:control|form:form)*)>`
	`<!ATTLIST form:form %name;`
	`%service-name;>`

The attributes that you can associate with the `<form:form>` are as follows:

- Name. See Section 6.3.
- Service name. See Section 6.3.
- Action
- Target frame
- Method
- Enctype
- Allow deletes
- Allow inserts
- Allow updates
- Apply filter
- Command type
- Command
- Data source
- Master fields
- Detail fields
- Escape processing
- Filter
- Ignore result
- Navigation mode
- Order
- Tabbing cycle

Action

The `xlink:href` attribute represents the URL of the processing agent for the form.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ATTLIST form:form xlink:href %url; #IMPLIED>`

Target Frame

The `office:target-frame` attribute specifies the target frame of the form. See Chapter 2 for more detailed information about this attribute.

XML Code:	`office:target-frame`
Rules:	This attribute can have one of the following values: • `_self`: The form replaces the content of the current frame. • `_blank`: The form is displayed in a new frame. • `_parent`: The form is displayed in the parent frame of the current frame. • `_top`: The form is displayed in the topmost frame, that is the frame that contains the current frame as a child or descendent but is not contained within another frame. • A frame name: The form is displayed in the named frame. If the named frame does not exist, a new frame with that name is created.
DTD:	`<!ATTLIST form:form office:target-frame CDATA "_blank">`

Method

The `form:method` attribute specifies the HTTP method to use to submit the data in the form to the server.

XML Code:	`form:method`
Rules:	The value of this attribute can be `get` or `post`. The default value is `get`. These values are not case sensitive.
DTD:	`<!ATTLIST form:form form:method CDATA "get">`

Enctype

If the value of the `form:method` attribute is `post`, the `form:enctype` attribute specifies the content type used to submit the form to the server.

XML Code:	`form:enctype`
Rules:	The default value of this attribute is `application/x-www-form-urlencoded` Other suitable MIME types are also acceptable. See Section 17.3 of the *HTML 4.01 Specification* for more information. See the Preface for a pointer to this specification.
DTD:	`<!ATTLIST form:form form:enctype CDATA "application/x-www-form-urlencoded">`

Allow Deletes

The `form:allow-deletes` attribute specifies whether or not data records can be deleted.

XML Code:	`form:allow-deletes`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:form form:allow-deletes %boolean; "true">`

Allow Inserts

The `form:allow-inserts` attribute specifies whether or not new data records can be inserted.

XML Code:	`form:allow-inserts`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:form form:allow-inserts %boolean; "true">`

Allow Updates

The `form:allow-updates` attribute specifies whether or not data records can be updated.

XML Code:	`form:allow-updates`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:form form:allow-updates %boolean; "true">`

Apply Filter

The `form:apply-filter` attribute specifies whether or not filters should be applied to the form.

XML Code:	`form:apply-filter`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:form form:apply-filter %boolean; "false">`

Command Type

The `form:command-type` attribute specifies the type of command to execute on the data source.

XML Code:	`form:command-type`
Rules:	The value of this attribute can be one of the following: • `table`: The command contains a table name. The form retrieves all of the data in the table. • `query` : The command contains the name of query. The form retrieves and executes the query. • `command` : The command contains, for example, an SQL statement. The form executes the SQL statement.
DTD:	`<!ATTLIST form:form form:command-type (table|query|command) "command">`

Command

The `form:command` attribute specifies the command to execute on the data source.

XML Code:	`form:command`
Rules:	The value of this attribute can be the name of a database table, the name of a query object or an SQL statement.
DTD:	`<!ATTLIST form:form form:command CDATA #IMPLIED>`

Data Source

The `form:datasource` attribute specifies the name of a data source to use for the form.

XML Code:	`form:datasource`
Rules:	The value of this attribute can be one of the following: • A URL specifying a database connection. • A data source name that OpenOffice.org uses to establish database connections.
DTD:	`<!ATTLIST form:form form:datasource CDATA #IMPLIED>`

Master Fields

The `form:master-fields` attribute is used for nested database forms. The attribute specifies the names of the columns in the parent form result set. The columns are usually the foreign key fields of the parent form. The values of the columns are used to identify the data for the subform. Each time the parent form changes the current row, the subform queries the database again based on the values of the master fields.

XML Code:	`form:master-fields`
Rules:	This attribute can contain a comma separated list of field names.
DTD:	`<!ATTLIST form:form form:master-fields CDATA #IMPLIED>`

Detail Fields

The `form:detail-fields` attribute is used for nested database forms. The attribute specifies the names of the columns in subforms that are related to columns in the parent form. The parent form is specified using the `form:master-fields` attribute. The columns represent part of the primary key fields or their aliases. The columns are used as parameters in the command for the nested form to retrieve the details for a matching master form.

XML Code:	`form:detail-fields`
Rules:	This attribute can contain a comma separated list of field names.
DTD:	`<!ATTLIST form:form form:detail-fields CDATA #IMPLIED>`

Escape Processing

If the value of the `form:command-type` attribute is `command`, the `form:escape-processing` attribute specifies whether or not the application processes the command before the database driver.

XML Code:	`form:escape-processing`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:form form:escape-processing %boolean; "true">`

Filter

The `form:filter` attribute specifies a filter for the command. The filter is usually part of the `WHERE` condition in an SQL query. The `form:apply-filter` attribute specifies whether or not filters are applied to the command.

XML Code:	`form:filter`
Rules:	
DTD:	`<!ATTLIST form:form form:filter CDATA #IMPLIED>`

Ignore Result

The `form:ignore-result` attribute specifies whether or not to discard all results that are retrieved from the data source.

XML Code:	`form:ignore-result`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:form form:ignore-result %boolean; "false">`

Navigation Mode

The `form:navigation-mode` attribute specifies how the records in a database form are navigated.

XML Code:	`form:navigation-mode`
Rules:	The value of this attribute can be one of the following: • `none`: A navigation bar is not provided. The form must be navigated using the TAB and SHIFT/TAB keys on the keyboard. • `current`: A navigation bar is provided and the navigation is performed on the current form. • `parent`: A navigation bar is provided and the navigation is performed on the parent form of the current form.
DTD:	`<!ENTITY %navigation "(none|current|parent)">` `<!ATTLIST form:form form:navigation-mode %navigation; #IMPLIED>`

Order

The `form:order` attribute specifies a sort criteria for the command. The sort is usually part of the `ORDER` condition in an SQL query.

XML Code:	`form:order`
Rules:	
DTD:	`<!ATTLIST form:form form:order CDATA #IMPLIED>`

Tabbing Cycle

The `form:tab-cycle` attribute specifies how the application responds when the user presses the TAB key in the controls in a form. The behavior of the application depends on whether or not the form is bound to a data source.

XML Code:	`form:tab-cycle`
Rules:	The value of this attribute can be one of the following: • `records`: If a user presses the TAB key in the last control of the form, the focus moves to the first control specified in the tab order for the next record of the same form. • `current`: If a user presses the TAB key in the last control of the form, the focus moves to the first control specified in the tab order for the same record. • `page`: If a user presses the TAB key in the last control of a form, the focus moves to the first control specified in the tab order for the next form.
DTD:	`<!ENTITY %cycles "(records|current|page)">` `<!ATTLIST form:form form:tab-cycle %cycles; #IMPLIED>`

6.2 Controls

Users use controls to interact with forms. Each control in a form is identified by a name. Controls are not connected to the text flow of a document. Controls are connected to a form by binding them to a shape that acts as a placeholder for the control.

The `<form:control>` element represents a control.

XML Code:	`<form:control>`
Rules:	

DTD:	`<!ENTITY % controls`
	`"(form:text|form:textarea|form:fixed-text|form:file|`
	`form:password|form:formatted-text|`
	`form:button|form:image|form:checkbox|form:radio|form:listbox|`
	`form:combobox|form:frame|form:hidden|form:image-frame|form:`
	`grid|form:generic-control)">`
	`<!ELEMENT form:control (%controls;+)>`
	`<!ATTLIST form:control %name;`
	`%service-name;`
	`%control-id;>`

Every control has the following two values:

- An initial value
- A current value

Both values are character strings. In general, the initial value is specified in a `form:value` attribute of the `<form:control>` element. The initial value of a control does not change.

The current value is initially set to the initial value. Thereafter the current value can be modified during user interaction and scripting. When a form is reset, the current value of each control is reset to the initial value for the control. If a control does not have an initial value, the result of resetting the form is undefined.

When a user submits a form for processing, the names of some controls are paired with the current values of the controls and the pairs are submitted with the form. These controls are called successful controls. See Section 17.13.2 of the *HTML 4.01 Specification* for more information. See the Preface for a pointer to this specification.

You can also link controls to a database field. The control receives a value from a data source. The modified data is stored in the data source instead of in the document.

Controls contain attributes, which are defined by the file format. Some applications require additional control attributes. These attributes arestored in the `<form:properties>` element in each control. Control events are specified in the `<office:events>` element.

The OpenOffice.org XML file format provides elements for the following standard controls:

- Text
- Text area
- Password
- File
- Formatted text
- Fixed text
- Combo box
- List box
- Button
- Image
- Check box
- Radio button
- Frame
- Image frame
- Hidden
- Grid

You can also define application-specific controls. These controls are described by the `<form:generic-control>` element.

6.2.1 Text

The `<form:text>` element defines a control for inputting text, which also allows a user to alter the text displayed.

XML Code:	`<form:text>`
Rules:	

DTD:	`<!ELEMENT form:text (form:properties?, office:events?)>`
	`<!ATTLIST form:text %current-value;`
	`%disabled;`
	`%max-length;`
	`%printable;`
	`%readonly;`
	`%tab-index;`
	`%tab-stop;`
	`%title;`
	`%value;`
	`%convert-empty;`
	`%data-field;>`

The attributes that you can associate with the `<form:text>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Current Value, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See Section 6.4 for information about these attributes.
- Convert Empty and Data Field. See Section 6.4.1 for information about these attributes.

6.2.2 Text Area

The `<form:textarea>` element defines a control for inputting text on multiple lines, which also allows a user to alter the text displayed.

XML Code:	`<form:textarea>`
Rules:	

DTD:	`<!ELEMENT form:textarea (form:properties?, office:events?)>`
	`<!ATTLIST form:textarea %current-value;`
	`%disabled;`
	`%max-length;`
	`%printable;`
	`%readonly;`
	`%tab-index;`
	`%tab-stop;`
	`%title;`
	`%value;`
	`%convert-empty;`
	`%data-field;>`

The attributes that you can associate with the `<form:textarea>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Current Value, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See Section 6.4 for information about these attributes.
- Convert Empty and Data Field. See Section 6.4.1 for information about these attributes.

6.2.3 Password

The <form:password> element defines a control that hides the text that a user inputs using an echo character, for example, an asterisk. This type of control is often used for inputting sensitive text such as a password

XML Code:	<form:password>
Rules:	
DTD:	<!ELEMENT form:password (form:properties?, office:events?)> <!ATTLIST form:password %disabled; %max-length; %printable; %tab-index; %tab-stop; %title; %value;>

The attributes that you can associate with the <form:password> element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Disabled, Maximum Length, Printable, Tab Index, Tab Stop, Title and Value. See Section 6.4 for information about these attributes.
- Echo Char

Echo Char

The form:echo-char attribute specifies the character that the form uses in place of the text that a user inputs in a password control.

XML Code:	form:echo-char
Rules:	
DTD:	<!ATTLIST form:password form:echo-char CDATA "*">

6.2.4 File

The <form:file> element defines a control for selecting a file.

XML Code:	<form:file>
Rules:	
DTD:	<!ELEMENT form:file (form:properties?, office:events?)> <!ATTLIST form:file %current-value; %disabled; %max-length; %printable; %readonly; %tab-index; %tab-stop; %title; %value;>

The attributes that you can associate with the <form:file> element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Current Value, Disabled, Printable, Read only, Tab Index, Tab Stop, Title and Value. See Section 6.4 for information about these attributes.

6.2.5 Formatted Text

The `<form:formatted-text>` element defines a control for inputting formatted text, which also allows a user to alter the text displayed.

XML Code:	`<form:formatted-text>`
Rules:	

DTD:	`<!ELEMENT form:formatted-text (form:properties?, office:events?)>`
	`<!ATTLIST form:formatted-text %current-value;`
	`%disabled;`
	`%max-length;`
	`%printable;`
	`%readonly;`
	`%tab-index;`
	`%tab-stop;`
	`%title;`
	`%value;`
	`%convert-empty;`
	`%data-field;>`

The attributes that you can associate with the `<form:formatted-text>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Current Value, Disabled, Maximum Length, Printable, Read only, Tab Index, Tab Stop, Title and Value. See Section 6.4 for information about these attributes.
- Convert Empty and Data Field. See Section 6.4.1 for information about these attributes.
- Maximum Value
- Minimum Value
- Validation

Maximum Value

The `form:max-value` attribute specifies the maximum value that a user can enter.

XML Code:	`form:max-value`
Rules:	
DTD:	`<!ATTLIST form:formatted-text form:max-value CDATA #IMPLIED>`

Minimum Value

The `form:min-value` attribute specifies the minimum value that a user can enter.

XML Code:	`form:min-value`
Rules:	
DTD:	`<!ATTLIST form:formatted-text form:min-value CDATA #IMPLIED>`

Validation

The `form:validation` attribute specifies whether or not the text that the user enters is validated.

XML Code:	`form:validation`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:formatted-text form:validation %boolean; "false">`

6.2.6 Fixed Text

The `<form:fixed-text>` element attaches additional information to controls and displays the information in the application. The information is also displayed in error messages and warnings for the related controls. Relations can be established by specifying the `form:for` attribute of the label. Only one label may be associated with the same control.

XML Code:	`<form:fixed-text>`
Rules:	

DTD:	`<!ELEMENT form:fixed-text (form:properties?, office:events?)>`
	`<!ATTLIST form:fixed-text %for;`
	`%disabled;`
	`%label;`
	`%printable;`
	`%title;>`

The attributes that you can associate with the `<form:fixed-text>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Disabled, For, Label, Printable, and Title. See Section 6.4 for information about these attributes.
- Multi-Line

Multi-Line

The `form:multi-line` attribute specifies whether or not the label is displayed on multiple lines.

XML Code:	`form:multi-line`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:fixed-text form:multi-line %boolean; "false">`

6.2.7 Combo Box

The `<form:combobox>` element defines a control containing a list of possible values for a text input control.

XML Code:	`<form:combobox>`
Rules:	

DTD:	`<!ELEMENT form:combobox (form:properties?, office:events?, form:item*)>`
	`<!ATTLIST form:combobox %current-value;`
	`%disabled;`
	`%dropdown;`
	`%max-length;`
	`%printable;`
	`%readonly;`
	`%size;`
	`%tab-index;`
	`%tab-stop;`
	`%title;`
	`%value;`
	`%convert-empty;`
	`%data-field;`
	`%list-source;`
	`%list-source-type;>`

The attributes that you can associate with the `<form:combobox>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Current Value, Disabled, Dropdown, Max Length, Printable, Read only, Size, Tab Index, Tab Stop, Title, and Value. See Section 6.4 for information about these attributes.
- Convert Empty, Data Field, List Source, and List Source Type. See Section 6.4.1 for information about these attributes.
- Automatic Completion

Automatic Completion

The `form:auto-complete` attribute specifies that when a user enters text in the combobox that matches one of the list items in the combobox, the application completes the text for the user.

XML Code:	`form:auto-complete`
Rules:	
DTD:	`<!ATTLIST form:combobox form:auto-complete %boolean; #IMPLIED>`

6.2.8 Item

The `<form:item>` element defines the list items for a combobox control.

XML Code:	`<form:item>`
Rules:	
DTD:	`<!ELEMENT form:item (#PCDATA)>` `<!ATTLIST form:item %label;>`

The attribute that you can associate with the `<form:item>` element is:

- Label. See Section 6.4 for information about this attribute.

6.2.9 List Box

The <form:listbox> element defines an input control that allows a user to select one or more items from a list. It is another representation of a group of radio buttons.

XML Code:	<form:listbox>
Rules:	
DTD:	<!ELEMENT form:listbox (form:properties?, office:events?, form:option*)> <!ATTLIST form:listbox %disabled; %dropdown; %printable; %size; %tab-index; %tab-stop; %title; %bound-column; %data-field; %list-source; %list-source-type;>

The attributes that you can associate with the <form:listbox> element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Disabled, Dropdown, Printable, Read only, Size, Tab Index, Tab Stop, and Title. See Section 6.4 for information about these attributes.
- Bound Column, Data Field, List Source, and List Source Type. See Section 6.4.1 for information about these attributes.
- Multiple

Multiple

The form:multiple attribute determines whether or not a user can select multiple items from a list box.

XML Code:	form:multiple
Rules:	The value of this attribute can be true or false.
DTD:	<!ATTLIST form:listbox form:multiple %boolean; "false">

6.2.10 Option

The <form:option> element defines the list items for a list box control. An item can be preselected and can contain a related value.

XML Code:	<form:option>
Rules:	
DTD:	<!ELEMENT form:option (#PCDATA)> <!ATTLIST form:option %current-selected; %selected; %label; %value;>

The attributes that you can associate with the <form:option> element are:

- Current Selected, Selected, Label, and Value. See Section 6.4 for information about these attributes.

6.2.11 Button

The <form:button> element defines a button. Use a value attribute to define the content of the button. When pressed, a button usually triggers an action.

XML Code:	<form:button>
Rules:	
DTD:	<!ELEMENT form:button (form:properties?, office:events?)> <!ATTLIST form:button %button-type; %disabled; %image-data; %printable; %tab-index; %tab-stop; %target-frame; %target-location; %title; %value;>

The attributes that you can associate with the <form:button> element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Button Type, Control ID, Disabled, Image Data, Printable, Tab Index, Tab Stop, Target Frame, Target Location, Title, and Value. See Section 6.4 for information about these attributes.
- Default Button

Default Button

The form:default-button attribute determines whether or not the button is the default button on the form. If a user clicks the default button or presses Return, the application takes the same action.

XML Code:	form:default-button
Rules:	The value of this attribute can be true or false. If you define more than one default button, the behaviour of the application is undefined.
DTD:	<!ATTLIST form:default-button %boolean; "false">

6.2.12 Image

The <form:image> element defines a graphical button control. This element corresponds to the input element of type image in HTML 4.01. **Note:** HTML 4.01 only allows the button type to be submit for an image button. In OpenOffice.org XML, an image button can be of any type.

XML Code:	`<form:image>`
Rules:	

DTD:	`<!ELEMENT form:image (form:properties?, office:events?)>`
	`<!ATTLIST form:image %button-type;`
	`%disabled;`
	`%image-data;`
	`%printable;`
	`%tab-index;`
	`%tab-stop;`
	`%target-frame;`
	`%target-location;`
	`%title;`
	`%value;>`

The attributes that you can associate with the `<form:image>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Button Type, Control ID, Disabled, Image Data, Printable, Tab Index, Tab Stop, Target Frame, Target Location, Title, and Value. See Section 6.4 for information about these attributes.

6.2.13 Check Box

A check box is an on/off control which a user can toggle. The control is on when the value of the `form:current-state` attribute associated with the control element is `checked`. When a user submits a form, only the controls whose current state is `checked` are successful.

XML Code:	`<form:checkbox>`
Rules:	

DTD:	`<!ELEMENT form:checkbox (form:properties?, office:events?)>`
	`<!ATTLIST form:checkbox %disabled;`
	`%label;`
	`%printable;`
	`%tab-index;`
	`%tab-stop;`
	`%title;`
	`%value;`
	`%data-field;>`

The attributes that you can associate with the `<form:checkbox>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Disabled, Label, Printable, Tab Index, Tab Stop, Title, and Value. See Section 6.4 for information about these attributes.
- Data Field. See Section 6.4.1 for information about this attribute.
- Current State
- Is Tristate
- State

Current State

The `form:current-state` attribute specifies the current state of the check box control. If the value of this attribute is set, it overrides the current state.

XML Code:	`form:current-state`
Rules:	The value of this attribute can be one of the following: • `unchecked`: The check box is not checked. • `checked`: The check box is checked. The value of the control is submitted with the form. • `unknown`: This value is only available when the control is in tristate mode. Use this value in connection with data field binding to indicate that the value is NULL.
DTD:	`<!ENTITY % states "(unchecked|checked|unknown)">` `<!ATTLIST form:checkbox form:current-state %states; #IMPLIED>`

Is Tristate

The `form:is-tristate` attribute specifies that the check box can have three states instead of the common two states.

XML Code:	`form:is-tristate`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST form:checkbox form:is-tristate %boolean; "false">`

State

The `form:state` attribute specifies the default state of the check box control. This state is used to initialise the control.

XML Code:	`form:state`
Rules:	
DTD:	`<!ATTLIST form:checkbox form:state %states; "unchecked">`

6.2.14 Radio Button

Radio buttons are like check boxes except that when several radio buttons share the same control name they are mutually exclusive. When one button is on, all of the other buttons with the same name are off. If no radio button is initially on, the way in which the application chooses which button to turn on initially is undefined.

If a group of radio buttons is bound to one database field, the value of the selected radio button is written into the database field.

XML Code:	`<form:radio>`
Rules:	

DTD:	`<!ELEMENT form:radio (form:properties?, office:events?)>`
	`<!ATTLIST form:radio %current-selected;`
	`%disabled;`
	`%label;`
	`%printable;`
	`%selected;`
	`%tab-index;`
	`%tab-stop;`
	`%title;`
	`%value;`
	`%data-field;>`

The attributes that you can associate with the `<form:radio>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Current Selected, Disabled, Label, Printable, Selected, Tab Index, Tab Stop, Title, and Value. See Section 6.4 for information about these attributes.
- Data Field. See Section 6.4.1 for information about this attribute.

6.2.15 Frame

The `<form:frame>` element defines a frame, which you can use to arrange controls visually. This element does not have a value and it does not allow any user input.

XML Code:	`<form:frame>`
Rules:	

DTD:	`<!ELEMENT form:frame (form:properties?, office:events?)>`
	`<!ATTLIST form:frame %disabled;`
	`%for;`
	`%label;`
	`%printable;`
	`%title;>`

The attributes that you can associate with the `<form:frame>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Disabled, For, Label, Printable, and Title. See Section 6.4 for information about these attributes.

6.2.16 Image Frame

The `<form:image-frame>` element defines a graphical control. The control displays an image, which is located at the image data location.

XML Code:	`<form:image-frame>`
Rules:	

DTD:	`<!ELEMENT form:image-frame (form:properties?, office:events?)>`
	`<!ATTLIST form:image-frame %disabled;`
	`%image-data;`
	`%printable;`
	`%readonly;`
	`%title;`
	`%data-field;>`

The attributes that you can associate with the `<form:image-frame>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Disabled, Image Data, Printable, Read only, and Title. See Section 6.4 for information about these attributes.
- Data Field. See Section 6.4.1 for information about this attribute.

6.2.17 Hidden

The `<form:hidden>` element defines a control that does not have a visual representation. This element is usually used as a container for information.

XML Code:	`<form:hidden>`
Rules:	

DTD:	`<!ELEMENT form:hidden (form:properties?, office:events?)>`
	`<!ATTLIST form:hidden %name;`
	`%service-name;`
	`%value;>`

The attributes that you can associate with the `<form:hidden>` element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Value. See Section 6.4 for information about these attributes.

6.2.18 Grid

The `<form:grid>` element defines a control that displays table data. This control is data-aware and is bound to a form which retrieves data from a data source. Each column in the grid is specified by a `<form:column>` element. Each column is bound to a field in the resulting data table. The rows in the grid contain the data records from the resulting data table.

XML Code:	`<form:grid>`
Rules:	

DTD:	`<!ELEMENT form:grid (form:properties?, office:events?, %form:`
	`column*)>`
	`<!ATTLIST form:grid %disabled;`
	`%printable;`
	`%tab-index;`
	`%tab-stop;`
	`%title;>`

The attributes that you can associate with the <form:grid> element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Control ID, Disabled, Printable, Tab Index, Tab Stop, and Title. See Section 6.4 for information about these attributes.

6.2.19 Column

The <form:column> element defines a column in a grid control. The column contains a control that displays the grid data for the column.

XML Code:	<form:column>
Rules:	
DTD:	<!ENTITY % column-type "(form:text| form:textarea| form:formatted-text| form: checkbox| form:listbox| form:combobox)"> <!ELEMENT form:column (form:properties?, office:events?, column-type;+)> <!ATTLIST form:column %name; %service-name; %label;>

The attributes that you can associate with the <form:column> element are:

- Name and Service Name. See Section 6.3 for information about these attributes.
- Label. See Section 6.4 for information about this attribute.

6.2.20 Generic Control

The <form:generic-control> element defines a placeholder for a generic control. The generic control can contain any properties and any events. The application detects the type of the control and instantiates the correct control.

XML Code:	<form:generic-control>
Rules:	
DTD:	<!ELEMENT form:generic-control (form:properties?, office: events?)>

The attributes that you can associate with the <form:generic-control> element are:

- Name and Service Name. See Section 6.3 for information about these attributes.

6.3 Common Form and Control Attributes

Name

The form:name attribute specifies the name of the form or control element. You can use this attribute to give a form or control element an identity, which is important for scripting and for submitting the content of controls.

XML Code:	`form:name`
Rules:	
DTD:	`<!ENTITY % name "form:name CDATA #IMPLIED">`

Service Name

The `form:service-name` attribute identifies the name of the service that the form or control supports.

XML Code:	`form:service-name`
Rules:	
DTD:	`<!ENTITY % service-name "form:service-name CDATA #IMPLIED">`

6.4 Common Control Attributes

Button Type

The `form:button-type` attribute specifies the type of a button. You can use this attribute with the following elements:

- `<form:button>`
- `<form:image>`

XML Code:	`form:button-type`
Rules:	The value of this attribute can be one of the following: • `submit`: Pressing the button submits the form. • `reset`: Pressing the button resets every control to its initial value. • `push`: Pressing the button does not perform a default action. Usually, there is a script bound to button. and the script is run • `url`: Pressing the button opens the URL that is specified in the `form:target-url` attribute.
DTD:	`<!ENTITY % types "(submit|reset|push|url)">` `<!ENTITY % button-type "form:button-type %types; 'push'">`
Note:	In HTML, the default button type is `submit`. In an OpenOffice.org application, the most common button type is `push`.

Control ID

Controls that are not hidden are contained in a form. The controls contain text or they have an absolute position. The position is represented by a frame that contains a reference to the control element within the form element. Controls that are not hidden must have a `form:id` associated with them, which is used to reference the controls.

You can use this attribute with the following elements:

- `<form:text>`

- <form:textarea>
- <form:password>
- <form:file>
- <form:formatted-text>
- <form:fixed-text>
- <form:combobox>
- <form:listbox>
- <form:button>
- <form:image>
- <form:checkbox>
- <form:radio>
- <form:frame>
- <form:image-frame>
- <form:grid>

XML Code:	form:id
Rules:	The ID can not contain a comma.
DTD:	<!ENTITY % control-id "form:id CDATA #REQUIRED">

Current Selected

The form:current-selected attribute determines the current state of a radio button or option element.

You can use this attribute with the following elements:

- <form:option>
- <form:radio>

XML Code:	form:current-selected
Rules:	The value of this attribute can be true or false.
DTD:	<!ENTITY %current-selected "form:current-selected %boolean; 'false'">

Current Value

The form:current-value attribute specifies the current status of an input control. It overrides the value of a form:value attribute, if one is present. HTML does not have an equivilant attribute.

You can use this attribute with the following elements:

- <form:text>
- <form:textarea>

- <form:file>
- <form:formatted-text>
- <form:combobox>

XML Code:	form:current-value
Rules:	
DTD:	<!ENTITY % current-value "form:current-value CDATA #IMPLIED">

Value

The form:value attribute specifies the value of an input control at a specified time.

You can use this attribute with the following elements:

- <form:text>
- <form:textarea>
- <form:password>
- <form:file>
- <form:formatted-text>
- <form:combobox>
- <form:option>
- <form:button>
- <form:image>
- <form:checkbox>
- <form:radio>
- <form:hidden>

XML Code:	form:value
Rules:	
DTD:	<!ENTITY % value "form:value CDATA #IMPLIED">

Default Value

The form:default-value attribute specifies the default value of a control. This value is displayed when the control is initially displayed or when a user resets a form.

XML Code:	form:default-value
Rules:	
DTD:	<!ENTITY % default-value "form:default-value CDATA #IMPLIED">

Disabled

The `form:disabled` attribute specifies whether or not a control can accept user input. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:fixed-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:frame>`
- `<form:image-frame>`
- `<form:grid>`

XML Code:	`form:disabled`
Rules:	The value of this attribute can be `true` or `false`. Controls that are disabled are not included in the tabbing navigation sequence and can not be selected.
DTD:	`<!ENTITY % disabled "form:disabled %boolean; 'false'">`

Dropdown

The `form:dropdown` attribute specifies whether the list in a combo box or list box is always visible or is only visible when the user clicks the drop-down button. You can use this attribute with the following elements:

- `<form:combobox>`
- `<form:listbox>`

XML Code:	`form:dropdown`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the list is always visible. If the value is `false`, the list is only visible when the user clicks the drop-down button.
DTD:	`<!ENTITY % dropdown "form:dropdown %boolean; 'false'">`

For

The attribute specifies the IDs of the controls with which the control element is used. You can use this attribute with the following elements:

- `<form:fixed-text>`
- `<form:frame>`

XML Code:	`form:for`
Rules:	This attribute contains a comma separated list of control IDs.
DTD:	`<!ENTITY % for "form:for CDATA #IMPLIED">`

Image Data

The `form:image-data` attribute links the control to an external file containing image data. You can use this attribute with the following elements:

- `<form:button>`
- `<form:image>`
- `<form:image-frame>`

XML Code:	`form:image-data`
Rules:	
DTD:	`<!ENTITY % image-data "form:image-data %url; #IMPLIED">`

Label

The `form:label` attribute contains a label for a control such as a radio button or check box. You can use this attribute with the following elements:

- `<form:fixed-text>`
- `<form:item>`
- `<form:option>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:frame>`
- `<form:column>`

XML Code:	`form:label`
Rules:	
DTD:	`<!ENTITY % label "form:label CDATA #IMPLIED">`

Maximum Length

The `form:max-length` attribute specifies the maximum number of characters that a user can enter in an input control. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:formatted-text>`
- `<form:combobox>`

XML Code:	`form:max-length`
Rules:	The default value of this attribute is `unlimited`, which allows a user to enter an unlimited number of characters.
DTD:	`<!ENTITY % max-length "form:max-length CDATA #IMPLIED">`

Printable

The `form:printable` attribute specifies whether or not a control is printed when a user prints the document in which the control is contained. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:fixed-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:frame>`
- `<form:image-frame>`
- `<form:grid>`

XML Code:	`form:printable`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ENTITY % printable "form:printable %boolean; 'true'">`

Read only

The `form:readonly` attribute specifies whether or not a user can modify the value of a control. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:image-frame>`

XML Code:	`form:readonly`
Rules:	The value of this attribute can be `true` or `false`. Read-only controls are included in the tabbing navigation sequence.
DTD:	`<!ENTITY % readonly "form:readonly %boolean; 'false'">`

Selected

The `form:selected` attribute specifies the default state of a radio button or option. When the control is initialised, it is in the default state specified by this attribute. You can use this attribute with the following elements:

- `<form:option>`
- `<form:radio>`

XML Code:	`form:selected`
Rules:	The value of this attribute can be `true` or `false`. In a group of radio buttons that share the same name, only one radio button can have this attribute set to `true`.
DTD:	`<!ENTITY % selected "form:selected %boolean; 'false'">`

Size

The `form:size` attribute specifies the number of rows that are visible at a time in a combo box list or a list box list. You can use this attribute with the following elements:

- `<form:combobox>`
- `<form:listbox>`

XML Code:	`form:size`
Rules:	
DTD:	`<!ENTITY % size "form:size CDATA #IMPLIED">`

Tab Index

The `form:tab-index` attribute specifies the tabbing navigation order of a control within a form. The tabbing order is the order in which controls are given focus when a user navigates through the form using the TAB key on the keyboard. The tabbing order can include elements that are nested in other elements. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:grid>`

XML Code:	`form:tab-index`
Rules:	The rules for tabbing are similar to the tabbing rules used in HTML 4.0. Controls that can be given focus are navigated in the order described in the following rules: 1. The controls that have a positive value for the `form:tab-index` attribute are navigated first. 2. The navigation starts at the control with lowest `form:tab-index` value and ends at the control with the highest value. Values do not have to be sequential and they do not have to begin with a particular value. 3. Controls that have the same values for the `form:tab-index` attribute are navigated according their position in the form. 4. Controls that do not contain the `form:tab-index` attribute or contain the attribute with a value of `0` are navigated next. These controls are navigated according to their position in the form. 5. Controls that have the `form:disabled` attribute set to `true` are not included in the nagigation.
DTD:	`<!ENTITY % tab-index "form:tab-index CDATA #IMPLIED">`

Tab Stop

The `form:tab-stop` attribute specifies whether or not a control is included in the tabbing navigation order. You can use this attribute with the following elements:

- `<form:text>`

- <form:textarea>
- <form:password>
- <form:file>
- <form:formatted-text>
- <form:combobox>
- <form:listbox>
- <form:button>
- <form:image>
- <form:checkbox>
- <form:radio>
- <form:grid>

XML Code:	form:tab-stop
Rules:	The value of this attribute can be true or false. If the value is false, the control is not included in the tabbing navigation.
DTD:	<!ENTITY % tab-stop "form:tab-stop %boolean; 'true'">

Target Frame

The office:target-frame attribute specifies the link target frame of the area. You can use this attribute with the following elements:

- <form:button>
- <form:image>

XML Code:	office:target-frame
Rules:	
DTD:	<!ENTITY % target-frame "office:target-frame CDATA '_blank'">

Target Location

An xlink:href attribute specifies the URL that is displayed if a button is clicked. You can use this attribute with the following elements:

- <form:button>
- <form:image>

XML Code:	xlink:href
Rules:	This attribute is only evaluated if the value of the form:button-type attribute is location.
DTD:	<!ENTITY % target-location "xlink:href %url; #IMPLIED">

Title

The `form:title` attribute contains additional information about a control. The value of the attribute can be used as a tool tip. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:password>`
- `<form:file>`
- `<form:formatted-text>`
- `<form:fixed-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:button>`
- `<form:image>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:image>`
- `<form:image-frame>`
- `<form:grid>`

XML Code:	`form:title`
Rules:	
DTD:	`<!ENTITY % title "form:title CDATA #IMPLIED">`

6.4.1 Database Attributes

You can bind controls to database fields. If you do this, the controls becomes data-aware. The control acquires the values of a database field by going through a result set that is provided by the form. Each time there is a row change in the form, the value of the control can change. The value changes are stored in the associated database field.

Bound Column

The `form:bound-column` attribute specifies the column values of the list source result set that are used to fill the data field values. You can use this attribute with the `<form:listbox>` element.

XML Code	`form:bound-column`
Rules:	
DTD:	`<!ENTITY % bound-column "form:bound-column CDATA #IMPLIED">`

Convert Empty To Null

The `form:convert-empty-to-null` attribute specifies whether or not empty current values are regarded as NULL This attribute is important for data-aware controls to determine which values to store for the bound database field. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:formatted-text>`
- `<form:combobox>`

XML Code:	`form:convert-empty-to-null`
Rules:	The value of this attribute can be `true` or `false`. If the value of the attribute is `true`, the value of the control is regarded as null. If the value of the attribute is `false`, the value of the control is regarded as an empty string.
DTD:	`<!ENTITY % convert-empty "form:convert-empty-to-null % boolean; 'false'">`

Data Field

The `form:data-field` attribute specifies the name of a result set column. The result set is specified for the form. You can use this attribute with the following elements:

- `<form:text>`
- `<form:textarea>`
- `<form:formatted-text>`
- `<form:combobox>`
- `<form:listbox>`
- `<form:checkbox>`
- `<form:radio>`
- `<form:image-frame>`

XML Code:	`form:data-field`
Rules:	
DTD:	`<!ENTITY % data-field "form:data-field CDATA #IMPLIED">`

List Source

The `form:list-source` attribute specifies the source used to populate the list in a list box or combo box. The first column of the list source result set populates the list. You can use this attribute with the following elements:

- `<form:combobox>`
- `<form:listbox>`

XML Code:	form:list-source
Rules:	
DTD:	<!ENTITY % list-source "form:list-source CDATA #IMPLIED">

List Source Type

The form:list-source-type attribute specifies the type of data source that is used to populates the list data in a list box or combo box. You can use this attribute with the following elements:

- <form:combobox>
- <form:listbox>

XML Code:	form:list-source-type
Rules:	The value of this attribute can be one of the following: • table: The list is populated using the content of a database table. • query: The list is populated by executing a query. • sql: The list is populated by executing an SQL statement. • sql-pass-through: The list is populated by executing any type of statement that is passed directly to a database driver. • value-list: The list is populated with values specified by the user using the form:value attribute in the <form:option> element. This setting is only applicable to list boxes. • table-fields: The list is populated using the field names in a database table.
DTD:	<!ENTITY % list-source-types "(table|query|sql|sql-pass-through|value-list|table-fields)"> <!ENTITY % list-source-type "form:list-source-type %list-source-types; #IMPLIED">

6.5 Events

HTML defines a list of standard events for controls. These events are represented by attributes, which are associated with the control elements. In the OpenOffice.org XML file format, these events and any additional events defined by the OpenOffice.org component are stored as elements in an <office:events> element.

For a single event element, the script:event-name attribute specifies the type of event and other attributes specify the language and the event handler.

6.5.1 Events with an Equivalent HTML Event Type

The following table describes the OpenOffice.org XML events that have an equivalent event in HTML.

Value of `script:event-name` Attribute	Equivalent HTML Event	Description of Event
`on-change`	onchange	Occurs when a control is no longer focussed and the value of the control was modified since it was given focus.
`on-focus`	onfocus	Occurs when a control is given focus using the mouse or the TAB key.
`on-blur`	onblur	Occurs when a control is no longer focussed as a result of moving the mouse or by tabbing navigation. It may be used with the same elements as `form:on-focus`.
`on-keydown`	onkeydown	Occurs when a key is pressed on a control.
`on-keyup`	onkeyup	Occurs when a key is released on a control.
`on-mouseover`	onmouseover	Occurs when the mouse pointer is moved over the control.
`on-mousemove`	onmousemove	Occurs when the mouse pointer is moved onto a control.
`on-mousedown`	onmousedown	Occurs when a mouse button is pressed on a control.
`on-mouseup`	onmouseup	Occurs when a mouse button is released on a control.
`on-mouseout`	onmouseout	Occurs when the mouse pointer is moved away from a control.
`on-reset`	onreset	Occurs when a form is reset.
`on-submit`	onsubmit	Occurs when a form is submitted.

6.5.2 OpenOffice.org Event Types

In addition to the HTML event types, OpenOffice.org XML allows additional events to be handled at run time.

Value of `script:event-name` Attribute	Applies To	Description of Event
`on-approveaction`	Button or image.	Occurs before the `on-performeaction` event takes place. Allows the user to veto the action.
`on-performeaction`	Button or image.	Occurs when the control action is to be performed. The common interpretation of this event is “pressing the button”.
`on-textchange`	All controls that allow text input.	Occurs when a user changes the text in a control.
`on-itemstatechange`	Check box or radio button.	Occurs when the state of a check box or radio button changes.
`on-mousedrag`	All controls.	Occurs when a user presses and holds one of the mouse buttons and moves the mouse pointer onto a control.
`on-approvereset`	same objects as for `form:on-reset`	Occurs before the `on-reset` event takes place. Allows the user to veto the reset event.
`on-approveupdate`	All controls that can be bound to a database field, that is controls that contain the `data-field` attribute.	Occurs before the `on-update` event takes place. Allows the user to veto the update.

Value of `script:event-name` Attribute	Applies To	Description of Event
`on-update`	All controls that can be bound to a database field, that is controls that contain the `data-field` attribute.	Occurs when the content of a control that is bound to a database field is committed.
`on-load`	Forms.	Occurs when the form establishes a connection to the data source.
`on-startrealod`	Forms.	Occurs when the form is about to refresh a data source connection.
`on-reload`	Forms.	Occurs when the form has refreshed a data source connection.
`on-startunload`	Forms.	Occurs when the form is about to drop a data source connection.
`on-unload`	Forms.	Occurs when the form has dropped a data source connection.
`on-confirmdelete`	Forms.	Occurs when the user is about to delete a record.
`on-approverowchange`	Forms.	Occurs before the on-rowchange event takes place. Allows the user to veto the change.
`on-rowchange`	Forms.	Occurs after changes to a row are complete, such as deletions, updates, and insertions.
`on-approvecursormove`	Forms.	Occurs before the form is moved to another row. Allows the user to veto the move.
`on-cursormove`	Forms.	Occurs after the form is moved to another row.
`on-supplyparameter`	Forms.	Occurs when the form needs to fill parameters to connect to a data source.
`on-error`	Forms, combo boxes and list boxes.	Occurs when a database-related error occurs.

6.6 Properties

You can use properties to store the following settings for controls and forms:

- Settings that are not known by the document format.
- Settings that are provided by external vendors.
- Settings that are specific to the application.

Properties consist of a name/value pair. The name identifies the property. The value can be given in a fundamental data type or as a list of fundamental data types.

6.6.1 Property Set

The `<form:properties>` element contains the property setting.

XML Code:	`<form:properties>`
Rules:	
DTD:	`<!ELEMENT form:properties (form:property+)>`

6.6.2 Property

The <form:property> element describes the name of the property, the property type, and whether or not the property contains list data or single data.

XML Code:	<form:property>
Rules:	This element is contained in the <form:properties> element.
DTD:	<!ELEMENT form:property (form:property-value*)>

The attributes that you can associate with the <form:property> element are:

- Is List
- Name
- Type

Is List

The form:property-is-list attribute specifies that the property can contain a list of property values.

XML Code:	form:property-is-list
Rules:	The value of this attribute can be true or false. If this attribute is not present or its value is false, only one following <form:property-value> element is evalutated.
DTD:	<!ATTLIST form:property form:property-is-list %boolean; #IMPLIED>

Property Name

The form:property-name attribute specifies the name of a property element.

XML Code:	form:property-name
Rules:	
DTD:	<!ATTLIST form:property form:property-name CDATA #REQUIRED>

Property Type

The form:property-type attribute specifies the data type of the property value.

XML Code:	`form:property-type`
Rules:	The value of this attribute can be one of the following: • `boolean`: requires the value true or false • `short`: specifies a 16-bit integer • `int`: specifies a 32-bit integer • `long`: specifies a 64-bit based integer • `double`: specifies a 64-bit floating point number • `string`: specifies a string
DTD:	`<!ATTLIST form:property form:property-type (boolean|short|int|long|double|string) #REQUIRED>`

6.6.3 Property Value

The `<form:property-value>` element specifies the content of the property.

XML Code:	`<form:property-value>`
Rules:	
DTD:	`<!ELEMENT form:property-value (#PCDATA)>`

Example: Sample properties

```
<form:properties>
  <form:property form:property-name="Name" form:property-type="string">
    <form:property-value>Name 1</form:property-value>
  </form:property>
  <form:property form:property-name="Items" form:property-type="string"
form:property-is-list="true">
    <form:property-value>Item 1</form:property-value>
    <form:property-value>Item 2</form:property-value>
    <form:property-value>Item 3</form:property-value>
  </form:property>
</form:properties>
```

CHAPTER 7

Indexing

This chapter describes the OpenOffice.org XML representation of indexes. It contains the following sections:

- Basic Components of OpenOffice.org XML Indexes
- Index Entries
- Index Source Styles
- Index Marks
- Table of Contents
- Index of Illustrations
- Index of Tables
- Index of Objects
- User-Defined Index
- Alphabetical Index
- Bibliography

7.1 Basic Components of OpenOffice.org XML Indexes

OpenOffice.org can automatically generate several types of index, depending on:

- The type of data to be indexed
- The way in which the data for the index is gathered
- The formatting options required for the index

An example of one type of automatically generated index is a table of contents.

An index is represented by an index element and this element contains the following two child elements:

- Index source
- Index body

7.1.1 Index Source

The index source elements describe how the content of an index is generated. The index source element alone is sufficient to recreate the index content, provided that the user did not change the source content since the last

index was generated. The index source element contains:

- Several attributes that aid the process of creating the index.
- Index entry template elements that describe the exact format of the individual index entries. For example, the index entries in the Table of Contents in this manual contain the section number, the section name, a tab stop, and the page number. The corresponding index entry template element contains one index entry element for each of these items.

7.1.2 Index Body

The index body element contains the text that makes up the body of the index. It is a standard block of text, with the possible addition of an index header element. If the write protection for the index is disabled, the user can modify the index body but these changes are lost when the index is updated again.

Since you can regenerate the index body at any time, it may seem unnecessary to export it. However, it is better to export the index body for the following reasons:

- It makes it easier to process the document using external tools because if it is not exported, the external tools must regenerate the index.
- Users can modify the index, even though their changes are lost when the index is updated.

The `<text:index-body>` element contains the text elements that make up the index. When the index is regenerated, the current index body content is overwritten.

XML Code:	`<text:index-body>`
Rules:	
DTD:	`<!ELEMENT text:index-body %text;>`

7.1.3 Index Title Template

The `<text:index-title-template>` element determines the style of the index title.

XML Code:	`<text:index-title-template>`
Rules:	The `text:style-name` attribute specifies the paragraph style to use for the index title. There can only be one `<text:index-title-template>` element contained in a `<text:table-of-content-source>` element.
DTD:	`<!ELEMENT text:index-title-template CDATA>` `<!ATTLIST text:index-title-template text:style-name % styleName; #IMPLIED>`

7.1.4 Index Entry Templates

The format of an index entry is determined by the index entry template element. There is a template element for each class of entry. For example, for a Table of Contents there is an element for the index header and an element for each outline level. For a Bibliography Index there is an element for the header and an element for each document class.

Each index template element contains a sequence of template elements, where each template element represents

one part of an index entry. The most common format for index entries is the chapter number, the chapter title, a tabbed space filled with dots, and a page number. To achieve this the index entry is configured to contain elements for the chapter number, the entry text, the tab space, and the page number.

Different types of indexes support different index entry elements. Therefore, each type of index supports a specific index entry template element with the valid child elements.

XML Code:	`<text:index-entry-template>`
Rules:	
DTD:	`<!ENTITY % templateElement "text:index-entry-chapter|text:index-entry-page-number|text:index-entry-text|text:index-entry-span|text:index-entry-tab-stop">` `<!ELEMENT text:index-entry-template (%entryElement;)*>`
Limitation:	The OpenOffice.org user interface only allows templates to use one page number and one entry text element.

Template Outline Level

This attribute specifies to which outline level this entry configuration applies. There may not be several `<text:outline-level>` elements for the same outline level within the same parent element.

XML Code:	`text:outline-level`
Rules:	This attribute must be unique among all `<text:outline-level>` elements. within the same parent element.
DTD:	`<!ATTLIST text:index-entry-template text:outline-level %number; #REQUIRED>`

Paragraph Style

The paragraph style attribute names the paragraph style to be used for instantiations of this template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:index-entry-template text:style-name %styleName; #REQUIRED>`

7.1.5 Common Index and Index Source Attributes

The following attribute is supported by all index elements:

- `text:outline-level`

The following attributes are supported by all index source elements:

- `text:use-index-marks`
- `text:index-scope`

Style Name

An index is formatted using a section style. This allows the index to contain multiple columns. Index header elements use their own section style, which enables a multicolumn index to have a single column title. The `text:style-name` attribute identifies the section style used to format the index.

XML Code:	`text:style-name`
Rules:	
Sample DTD:	`<!ATTLIST text:table-of-content text:style-name %styleName; #IMPLIED>`

Write Protection

You can protect indexes so that they are not accidentally changed by a user. If the `text:protected` attribute is set, the user interface should prevent users from modifying the contents of the index.

XML Code:	`text:protected`
Rules:	
Sample DTD:	`<!ATTLIST text:table-of-content text:protected %boolean; "false">`

Index Scope

All index source elements contain a `text:index-scope` attribute which determines whether the index is generated for the entire document or for the current chapter only.

XML Code:	`text:index-scope`
Rules:	
Sample DTD:	`<!ATTLIST text:table-of-content-source text:index-scope (document|chapter) "document">`
Note:	This attribute is not supported for Bibliography Indexes.

Relative Tab Stops in Index Entries

The `text:relative-tab-stop-position` attribute determines whether the position of tab stops is relative to the left margin or to the left indent as determined by the paragraph style. This is useful if you want to copy the same entry configuration for all outline levels because with relative tab stop positions the tabs do not need to be adjusted to the respective paragraph format.

XML Code:	`text:relative-tab-stop-position`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the tab stop position is relative to the left indent. If the value is `false`, the tab stop position is relative to the left margin.
DTD Example:	`<!ATTLIST text:table-of-content-source text:relative-tab-stop-position %boolean; "true">`
Note:	This attribute is not supported for Bibliography Indexes.

7.2 Index Entries

There are nine types of index entries, as follows:

- Chapter number
- Chapter information
- Entry text
- Page number
- Fixed string
- Bibliography information
- Tab stop
- Hyperlink start and end

7.2.1 Chapter Number

The `<text:index-entry-chapter-number>` element displays the chapter number of the index entry. The character style for the chapter number can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-chapter-number>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-chapter-number EMPTY>` `<!ATTLIST text:index-entry-chapter-number text:style-name % styleName; #IMPLIED>`
Note:	This element can only display the chapter number. To display the chapter name, you must use the `<text:index-entry-text>` elements.

7.2.2 Chapter Information

The `<text:index-entry-chapter>` element displays the chapter number of the index entry. The character style for the chapter number can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-chapter>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-chapter-number EMPTY>` `<!ATTLIST text:index-entry-chapter-number text:style-name % styleName; #IMPLIED>`
Note:	This element can only display the chapter number. To display the chapter name, you must use the `<text:index-entry-text>` elements.

Display Chapter Format

The `text:display` attribute displays either the chapter number, the chapter name, or both.

XML Code:	text:display
Rules:	
DTD:	<!ATTLIST text:index-entry-chapter text:display (name | number | number-and-name) "number-and-name">

7.2.3 Entry Text

The <text:index-entry-text> element displays the text of the index entry, for example, the chapter name if the entry is derived from a header or the phrase contained in the index mark if the entry is derived from an index mark. The character style for the entry text can be included in the index entry element as a text:style-name attribute.

XML Code:	<text:index-entry-text>
Rules:	
DTD:	<!ELEMENT text:index-entry-text EMPTY> <!ATTLIST text:index-entry-text text:style-name %styleName; #IMPLIED>

7.2.4 Page Number

The <text:index-entry-page-number> element displays the page number on which the index entry is located. The character style for the page number can be included in the index entry element as a text:style-name attribute.

XML Code:	<text:index-entry-page-number>
Rules:	
DTD:	<!ELEMENT text:index-entry-page-number EMPTY> <!ATTLIST text:index-entry-page-number text:style-name % styleName; #IMPLIED>

7.2.5 Fixed String

The <text:index-entry-span> element represents a fixed string within an index entry. The character style for the entry text can be included in the index entry element as a text:style-name attribute.

XML Code:	<text:index-entry-span>
Rules:	Unlike the <text:span> element, the <text:index-entry-span> element does not have any child elements.
DTD:	<!ELEMENT text:index-entry-string CDATA> <!ATTLIST text:index-entry-string text:style-name %styleName; #IMPLIED>

7.2.6 Bibliography Information

The <text:index-entry-bibliography> element introduces bibliography data into index entry

templates.

XML Code:	`<text:index-entry-bibliography>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-bibliography EMPTY>` `<!ATTLIST text:index-entry-bibliography text:style-name % styleName; #IMPLIED>`

Each `<text:index-entry-bibliography>` element can contain:

- A `text:style-name` attribute specifying a character style for the entry
- A bibliography data field identifier

Bibliography Data Field Identifier

The `text:bibliography-data-field` attribute determines which part of the bibliography data field to display.

XML Code:	`text:bibliography-data-field`
Rules:	
DTD:	`<!ATTLIST text:index-entry-bibliography text: bibliography-data-field (address | annote | author | bibiliographic_type | booktitle | chapter | custom1 | custom2 | custom3 | custom4 | custom5 | edition | editor | howpublished | identifier | institution | isbn | journal | month | note | number | organizations | pages | publisher | report_type | school | series | title | url | volume | year) #REQUIRED>`

7.2.7 Tab Stop

The `<text:index-entry-tab-stop>` element represents a tab stop within an index entry. It also contains the position information for the tab stop.

XML Code:	`<text:index-entry-tab-stop>`
Rules:	
DTD:	`<!ELEMENT text:index-entry-tab-stop EMPTY>` `<!ATTLIST text:index-entry-tab-stop text:style-name % styleName; #IMPLIED>`

The attributes that you can associate with the `<text:index-entry-tab-stop>` element are:

- `style:leader-char`
- `style:type`
- `style:position`

Leader Char

The `style:leader-char` attribute specifies the leader character.

XML Code:	`style:leader-char`
Rules:	
DTD:	`<!ATTLIST text:index-entry-tab-stop style:leader-char %char; " ">`

Tab Type

The `style:type` attribute specifies the tab stop type.

XML Code:	`style:type`
Rules:	The `<text:index-entry-tab-stop>` element only supports two types of tab: `left` and `right`. If the value of this attribute is `left`, the `style:position` attribute must also be used. Otherwise, this attribute must be omitted.
DTD:	`<!ATTLIST text:index-entry-tab-stop style:type (left|right) "left">`

Tab Position

The `style:position` attribute specifies the position of the tab.

XML Code:	`style:position`
Rules:	This attribute can only be used with `<text:index-entry-tab-stop>` elements where the value of the `style:type` attribute is `left`.
DTD:	`<!ATTLIST text:index-entry-tab-stop style:position %length; #IMPLIED>`
Note:	Depending on the value of the `text:relative-tab-stop-position` attribute in the `<text:index-entry-config>` element, the position of the tab is interpreted as being relative to the left margin or the left indent.

7.2.8 Hyperlink Start and End

The `<text:index-entry-link-start>` and `<text:index-entry-link-end>` elements mark the start and end of a hyperlink index entry. The character style for the hyperlink can be included in the index entry element as a `text:style-name` attribute.

XML Code:	`<text:index-entry-link-start>` `<text:index-entry-link-end>`
Rules:	

DTD:	`<!ELEMENT text:index-entry-link-start EMPTY>`
	`<!ELEMENT text:index-entry-link-end EMPTY>`
	`<!ATTLIST text:index-entry-link-start text:style-name % styleName; #IMPLIED>`
	`<!ATTLIST text:index-entry-link-end text:style-name % styleName; #IMPLIED>`

7.2.9 Example of an Index Entry Configuration

The following is an example of the XML code for a table of contents called Table of Content with the following characteristics:

- It uses the top two outline levels.
- Each entry consists of the chapter number, a closing parenthesis, the chapter title, a tab stop, and the page number.
- For the top outline level, the page number is formatted using a style called Bold.
- For the second outline level, a bracket is used instead of a closing parenthesis.

Example: Table of Content

```
<text:table-of-content>
  <text:table-of-content-source
    text:outline-level="2"
    text:use-index-marks="false"
    text:index-scope="document">

    <text:index-title-template text:style-name="Index 1">
      Table of Content
    </text:index-title-template>

    <text:index-entry-template
      text:ouline-level="1"
      text:style-name="Contents 1">
      <text:index-entry-chapter-number/>
      <text:index-entry-span>) </text:index-entry-span>
      <text:index-entry-text/>
      <text:index-entry-tab-stop style:type="right"/>
      <text:index-entry-page-number text:style-name="bold"/>
    </text:index-entry-template>

    <text:index-entry-template
      text:ouline-level="2"
      text:style-name="Contents 2">
      <text:index-entry-chapter-number/>
      <text:index-entry-span>] </text:index-entry-span>
      <text:index-entry-text/>
      <text:index-entry-tab-stop style:type="right"/>
      <text:index-entry-page-number/>
    </text:index-entry-template>

  </text:table-of-content-source>
```

```
  <text:table-of-content-body>
    [... header ...]
    <text:p text:style-name="...">1) Chapter
      <text:tab-stop/><text:span style-name="bold"> 1 </text:span>
    </text:p>
    <text:p text:style-name="...">1.1] Subchapter
       <text:tab-stop/>1
    </text:p>
    [... more entries ...]
  </text:table-of-content-body>

</text:table-of-content>
```

7.3 Index Source Styles

The table of content index can gather index entries from paragraphs formatted using certain paragraph styles. The <text:index-source-styles> element contains all of the <text:index-source-style> elements for a particular outline level.

XML Code:	<text:index-source-styles>
Rules:	
DTD:	<!ELEMENT text:index-source-styles (text:index-source-style)*>

Source Styles Outline Level

The text:outline-levels attribute determines at which outline level to list the index entries gathered from the respective paragraph styles.

XML Code:	text:outline-levels
Rules:	
DTD:	<!ATTLIST text:index-source-styles text:outline-level %number; #IMPLIED>

7.3.1 Index Source Style

All paragraphs formatted using the style specified in the <text:index-source-style> element are included in the index.

XML Code:	<text:index-source-style>
Rules:	
DTD:	<!ELEMENT text:index-source-style EMPTY>

The attribute associated with the <text:index-source-style> element is:

- Style name

Style Name

The text:style-name attribute specifies the paragraph style. Paragraphs formatted using this style are

included in the index.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:index-source-style text:style-name %styleName; #REQUIRED>`

7.4 Index Marks

There are three types of index marks to correspond to the three types of index that use of index marks. The three types of index marks are:

- Table of content index marks
- User-defined index marks
- Alphabetical index marks

The XML code for index marks is similar to the code for Bookmarks and References. The following are some basic rules about index marks:

- Each index mark is represented by a start and an end element.
- Both elements use an ID attribute to match the appropriate start and end elements.
- The start and end elements for an index mark must be contained in the same paragraph, with the start element occurring first.
- The attributes associated with the index mark are attached to the start element.
- The text between the start and end elements is the text the index entry.
- The formatting attributes for index marks can overlap.

7.4.1 Table of Content Index Marks

The `<text:toc-mark-start>` element marks the start of a table of content index entry.

XML Code:	`<text:toc-mark-start>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.

DTD:	`<!ELEMENT text:toc-mark-start EMPTY>`
	`<!ATTLIST text:toc-mark-start text:id ID #REQUIRED>`
	`<!ATTLIST text:toc-mark-start text:outline-level %number; #IMPLIED>`

The attributes associated with the `<text:toc-mark-start>` element are:

- A `text:id` attribute to allow the start and end elements to be matched.
- A `text:outline-level` attribute to specify the outline level of the resulting table of content index entry.

The `<text:toc-mark-end>` element marks the end of a table of contents index entry.

XML Code:	`<text:toc-mark-end>`
Rules:	The ID specified by the `text:id` attribute must be unique. The must be a start element to match the end element located in the same paragraph, with the start element appearing first.
DTD:	`<!ELEMENT text:toc-mark-end EMPTY>` `<!ATTLIST text:toc-mark-end text:id ID #REQUIRED>`

Table of content index marks also have a variant that does not enclose the text to be indexed. This is represented using the `<text:toc-mark>` element which contains a `text:string-value` attribute for the text of the index entry. In this situation, a `text:id` attribute is not necessary because there are no start and end elements to match.

XML Code:	`<text:toc-mark>`
Rules:	

DTD:	`<!ELEMENT text:toc-mark EMPTY>`
	`<!ATTLIST text:toc-mark text:string-value %string #REQUIRED>`
	`<!ATTLIST text:toc-mark text:outline-level %number; #IMPLIED>`

7.4.2 User-Defined Index Marks

The `<text:user-index-mark-start>` element marks the start of a user-defined index entry.

XML Code:	`<text:user-index-mark-start>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.

DTD:	`<!ELEMENT text:user-index-mark-start EMPTY>`
	`<!ATTLIST text:user-index-mark-start text:id ID #REQUIRED>`
	`<!ATTLIST text:user-index-mark-start text:outline-level %`
	`number; #IMPLIED>`

The `<text:user-index-mark-end>` element marks the end of the user-defined index entry.

XML Code:	`<text:user-index-mark-end>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be a start element to match the end element located in the same paragraph, with the start element appearing first.
DTD:	`<!ELEMENT text:user-index-mark-end EMPTY>` `<!ATTLIST text:user-index-mark-end text:id ID #REQUIRED>`

User index marks also have a variant that does not enclose the text to be indexed. This is represented by the `<text:user-index-mark>` element which contains a `text:string-value` attribute for the text of the index entry. In this situation, the `text:id` attribute is not necessary because there are no start and end elements to match.

XML Code:	`<text:user-index-mark>`
Rules:	

DTD:	`<!ELEMENT text:user-index-mark EMPTY>`
	`<!ATTLIST text:user-index-mark text:string-value %string; #REQUIRED>`
	`<!ATTLIST text:user-index-mark text:outline-level %number; #IMPLIED>`

Name of User Index

There can be more than one user-defined index. In this case, the user index must be named using the `text:index-name` attribute. This attribute determines to which user-defined index an index mark belongs. If no name is given, the default user-defined index is used.

XML Code:	`text:index-name`
Rules:	
DTD:	`<!ATTLIST text:user-index-mark text:index-name %string #IMPLIED>` `<!ATTLIST text:user-index-mark-start text:index-name %string #IMPLIED>`

7.4.3 Alphabetical Index Mark

The `<text:alpha-index-mark-start>` element marks the start of an alphabetical index entry. Since alphabetical entries may make use of two keys to structure entries, there are two optional attributes for these keys. There is also a boolean attribute that determines if this entry is intended to be the main entry, if there are several equal entries.

XML Code:	`<text:alpha-index-mark-start>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be an end element to match the start element located in the same paragraph, with the start element appearing first.
DTD:	`<!ELEMENT text:alpha-index-mark-start EMPTY>` `<!ATTLIST text:alpha-index-mark-start text:id ID #REQUIRED>`

The attributes associated with the `<text:toc-mark-start>` element are:

- A `text:id` attribute to allow the start and end elements to be matched.
- Additional keys
- Main entry

The `<text:alpha-index-mark-end>` element marks the end of an alphabetical index entry.

XML Code:	`<text:alpha-index-mark-end>`
Rules:	The ID specified by the `text:id` attribute must be unique. There must be a start element to match the end element located in the same paragraph, with the start element appearing first.
DTD:	`<!ELEMENT text:alpha-index-mark-end EMPTY>` `<!ATTLIST text:alpha-index-mark-end text:id ID #REQUIRED>`

Alphabetical index marks also have a variant that does not enclose the text to be indexed. This is represented using the `<text:alpha-index-mark>` element which contains a `text:string-value` attribute for the text of the index entry. In this situation, a `text:id` attribute is not necessary because there are no start and end elements to match.

XML Code:	`<text:alpha-index-mark>`
Rules:	
DTD:	`<!ELEMENT text:alpha-index-mark EMPTY>` `<!ATTLIST text:alpha-index-mark text:string-value %string; #REQUIRED>`

Additional Keys

The `text:key1` and `text:key2` attributes specify additional keys for the alphabetical index mark.

XML Code:	`text:key1` `text:key2`
Rules:	If only one key is used, it must be contained in the `text:key1` attribute.

DTD:	`<!ATTLIST text:alpha-index-mark-start`
	`text:key1 %string; #IMPLIED`
	`text:key2 %string; #IMPLIED >`
	`<!ATTLIST text:alpha-index-mark`
	`text:key1 %string; #IMPLIED`
	`text:key2 %string; #IMPLIED >`

Main Entry

If there are several index marks for the same entry, you can declare one entry as the main entry using the `text:main-entry` attribute.

XML Code:	`text:main-entry`
Rules:	

DTD:	`<!ATTLIST text:alpha-index-mark-start`
	`text:main-entry %boolean; "false">`
	`<!ATTLIST text:alpha-index-mark`
	`text:main-entry %boolean; "false">`

7.4.4 Bibliography Index Mark

The `<text:bibliography-mark>` element contains the text and informatin for a bibliography index entry. It supports attributes for each type of bibliographical data that a bibliography index may contain.

XML Code: <text:bibliography-mark>

Rules:

DTD:

```
<!ELEMENT text:bibliography-mark (#PCDATA)>
<!ATTLIST text:bibliography-mark text:bibiliographic-type
   ( article | book | booklet | conference | custom1 |
     custom2 | custom3 | custom4 | custom5 | email | inbook |
     incollection | inproceedings | journal | manual |
     mastersthesis | misc | phdthesis | proceedings |
     techreport | unpublished | www ) #REQUIRED >
<!ATTLIST text:bibliography-mark text:identifier CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:address CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:annote CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:author CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:booktitle CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:chapter CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:edition CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:editor CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:howpublished CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:institution CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:journal CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:month CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:note CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:number CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:organizations CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:pages CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:publisher CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:school CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:series CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:title CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:report-type CDATA
#IMPLIED>
<!ATTLIST text:bibliography-mark text:volume CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:year CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:url CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom1 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom2 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom3 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom4 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:custom5 CDATA #IMPLIED>
<!ATTLIST text:bibliography-mark text:isbn CDATA #IMPLIED>
```

7.5 Table of Contents

A table of contents can be created from:

- Table of content index marks in the text
- The outline structure formed by the headers
- Arbitrary paragraph formats

The table of contents is represented by the <text:table-of-content> element.

XML Code:	`<text:table-of-content>`
Rules:	
DTD:	`<!ELEMENT text:table-of-content (text:table-of-content-source, text:index-body)>`

The attribute that you can associate with the `<text:table-of-content>` element is:

- `text:style-name`

  This attribute specifies the section style to use for formatting the table of contents.

DTD:	`<!ATTLIST text:table-of-content text:style-name %styleName; #IMPLIED>`

7.5.1 Table of Content Source

The `<text:table-of-content-source>` element specifies how the table of contents is generated. It specifies how the entries are gathered.

XML Code:	`<text:table-of-content-source>`
Rules:	
DTD:	`<!ELEMENT text:table-of-content-source (text:index-header-template? | text:table-of-content-entry-template* | text:index-source-styles* ) >`

The attributes that you can attach to the `<text:table-of-content-source>` element are:

- Outline level
- Use index marks
- Use index source styles
- Index scope

  See Section 7.1.5 for information about this attribute.

DTD:	`<!ATTLIST text:table-of-content-source text:index-scope (document|chapter) "document">`

- Relative tab stop position

  See Section 7.1.5 for information about this attribute.

DTD:	`<!ATTLIST text:table-of-content-source text:relative-tab-stop-position %boolean; "true">`

Outline Level

The `text:outline-level` attribute specifies which outline levels are used when generating the table of contents.

XML Code:	`text:outline-level`

Rules:	The value of this attribute must be `none` or an integer greater than zero. If the value of this attribute is `none`, no entries are generated from the outline. If this attribute is omitted, all outline levels are used by default.
DTD:	`<!ATTLIST text:table-of-content-source text:outline-level CDATA #REQUIRED>`

Use Index Marks

The `text:use-index-marks` attribute determines whether or not to use index marks to generate the table of content.

XML Code:	`text:use-index-marks`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the table of contents includes entries generated from table of content index marks. See Section 7.4 for more information on index marks.
DTD:	`<!ATTLIST text:table-of-content-source text:use-index-marks %boolean; "true">`

Use Index Source Styles

The `text:use-index-source-styles` attribute determines whether or not index entries are generated for paragraph formatted using certain paragraph styles.

XML Code:	`text:use-index-source-styles`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the table of contents includes an entry for every paragraph formatted with one of the styles specified in a `<text:index-source-style>` element.
DTD:	`<!ATTLIST text:table-of-content-source text:use-index-source-styles %boolean; "false">`

7.5.2 Table of Content Entry Template

The `<text:table-of-content-entry-template>` element determines the format of an index entry for a particular outline level.

XML Code:	`<text:table-of-content-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-chapter-number` • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop` • `text:index-entry-link-start` • `text:index-entry-link-end`
DTD:	`<!ELEMENT text:table-of-content-entry-template ( text:index-entry-chapter | text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop | text:index-entry-link-start | text:index-entry-link-end )*>`
Limitation:	The OpenOffice.org user interface only allows templates to use one page number and one entry text element.

The attributes that you can associate with the `<text:table-of-content-entry-template>` element are:

- Template outline level
- Paragraph style

Template Outline Level

This attribute specifies to which outline level the entry configuration applies.

XML Code:	`text:outline-level`
Rules:	You cannot have several `<text:outline-level>` elements for the same outline level within a parent element.
DTD:	`<!ATTLIST text:table-of-content-entry-template text:outline-level %number; #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for this template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:table-of-content-entry-template text:style-name %styleName; #REQUIRED>`

7.6 Index of Illustrations

The index of illustrations lists all images and graphics in the current document or chapter. The index entries can be derived from the caption of the illustration or the name of the illustration.

XML Code:	`<text:illustration-index>`
Rules:	
DTD:	`<!ELEMENT text:illustration-index (text:illustration-index-source, text:index-body)>`

The attributes that you can attach to the `<text:illustration-index>` element are:

- `text:style-name`

  This attribute specifies the section style to use for the index of illustrations.

  DTD: `<!ATTLIST text:illustration-index text:style-name %styleName; #IMPLIED>`

7.6.1 Index of Illustration Source

The `<text:illustration-index-source>` element specifies how the index of illustrations is generated.

XML Code:	`<text:illustration-index-source>`
Rules:	
DTD:	`<!ELEMENT text:illustration-index-source (text:index-header-template? | text:illustration-index-entry-template?) >`

The attributes you can use with a `<text:illustration-index-source>` element are:

- Use caption
- Caption sequence name
- Caption sequence format
- Index scope

  This attribute specifies whether the index applies to the entire document or only the the current chapter.

  DTD: `<!ATTLIST text:illustration-index-source text:index-scope (document|chapter) "document">`

- `text:relative-tab-stop-position`

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

  DTD: `<!ATTLIST text:illustration-index-source text:relative-tab-stop-position %boolean; "true">`

Use Caption

In OpenOffice.org, each object contained in a text document has a name. In addition, images also have a caption. The image caption or the image name can be gathered for the index of illustrations.

XML Code:	`text:use-caption`
Rules:	This attribute can have a value of `true` or `false`. If the value is `true`, the image caption is used. If the value is `false`, the image name is used.
DTD:	`<!ATTLIST text:illustration-index-source text:use-caption %boolean; "true">`

Caption Sequence Name

Captions are associated with a sequence name. If the `text:use-caption` attribute is set to `true`, you must use this attribute to specify the sequence with which the captions are associated.

XML Code:	`text:caption-sequence-name`
Rules:	If this attribute is omitted, the default sequence for the object type is used, for example the sequence “Illustration” is used for illustrations.
DTD:	`<!ATTLIST text:illustration-index-source text:caption-sequence-name %string; #IMPLIED>`

Caption Sequence Format

If the entries for the index of illustrations are obtained from the image captions, you must use this attribute to specify the format for the entries.

XML Code:	`text:sequence-format`
Rules:	The value of this attribute can be `text`, `category-and-value`, or `caption`.
DTD:	`<!ATTLIST text:illustration-index-source text:sequence-format (text|category-and-value|caption) #IMPLIED>`

7.6.2 Illustration Index Entry Template

The illustration index entry template element determines the format of an index entry for a particular outline level.

XML Code:	`<text:illustration-index-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop`
DTD:	`<!ELEMENT text:illustration-index-entry-template ( text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop )*>`
Limitation:	The OpenOffice.org user interface only allows templates to use one page number and one entry text element.

Since you can only have one `<text:illustration-index-entry-template>` element, you do not need to use the `text:outline-level` attribute. The attribute that you can associate with the `<text:illustration-index-entry-template>` element is:

- Paragraph style

Paragraph Style

This attribute identifies the paragraph style to use for this template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:illustration-index-entry-template text:style-name %styleName; #REQUIRED>`

7.7 Index of Tables

The index of tables lists all of the tables in the current document or chapter. It works in exactly the same way as the index of illustrations.

XML Code:	`<text:table-index>`
Rules:	
DTD:	`<!ELEMENT text:table-index (text:table-index-source, text:index-body)>` `<!ATTLIST text:table-index text:style-name %styleName; #IMPLIED>`

7.7.1 Table Index Source

The `<text:table-index-source>` element specifies how the index of tables is generated.

XML Code:	`<text:table-index-source>`
Rules:	
DTD:	`<!ELEMENT text:table-index-source (text:index-header-template?, text:table-index-entry-template?) >` `<!ATTLIST text:table-index-source text:index-scope (document|chapter) "document" text:relative-tab-stop-position %boolean; "true" text:use-caption %boolean; "true" text:caption-sequence-name %string; #IMPLIED text:sequence-format (text|category-and-value|caption) #IMPLIED>`

The attributes that you can associate with this element are the same as those that can be associated with the `<text:illustration-index-source>` element. See Section 7.6.1 for detailed information about these attributes.

7.7.2 Table Index Entry Template

The table index entry template element determines the format of an index entry for a particular outline level.

XML Code:	`<text:table-index-entry-template>`
Rules:	
DTD:	`<!ELEMENT text:table-index-entry-template ( text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop )*>` `<!ATTLIST text:table-index-entry-template text:style-name %styleName; #REQUIRED>`

The attributes that you can associate with this element are the same as those that can be associated with the `<text:illustration-index-entry-template>` element. See Section 7.6.1 for detailed information about these attributes.

7.8 Index of Objects

The index of objects lists all of the objects in the current document or chapter. It gathers its entries from the known object types.

XML Code:	`<text:object-index>`
Rules:	
DTD:	`<!ELEMENT text:object-index (text:object-index-source, text:index-body)>` `<!ATTLIST text:object-index text:style-name %styleName; #IMPLIED>`

7.8.1 Object Index Source

The `<text:object-index-source>` element determines which object types to include in the index of

objects. It also supports the standard index source attributes.

XML Code:	`<text:object-index-source>`
Rules:	
DTD:	`<!ELEMENT text:object-index-source (text:index-header-template?, text:object-index-entry-template?) >`

The attributes that you can associate with the `<text:object-index-source>` element are:

- Use attributes, `text:use-*-objects`
- Index scope

  This attribute specifies whether the index applies to the entire document or only the the current chapter.

  DTD:
  ```
  <!ATTLIST text:illustration-index-source text:index-scope
  (document|chapter) "document">
  ```

- Relative tab stop position

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

  DTD:
  ```
  <!ATTLIST text:illustration-index-source text:relative-tab-
  stop-position %boolean; "true">
  ```

Use Attributes

The `text:use-*-objects` attributes specify which types of objects to include in the index of objects. There is an attribute for each type of OpenOffice.org object as follows:

- `text:use-spreadsheet-objects`
- `text:use-draw-objects`
- `text:use-chart-objects`
- `text:use-math-objects`

Other objects are included or omitted using the following attribute:

- `text:use-other-objects`

XML Code:	`text:use-spreadsheet-objects`
	`text:use-math-objects`
	`text:use-draw-objects`
	`text:use-chart-objects`
	`text:use-other-objects`

Rules:	
DTD:	`<!ATTLIST text:object-index-source text:use-spreadsheet-objects %boolean; "false" text:use-draw-objects %boolean; "false" text:use-chart-objects %boolean; "false" text:use-other-objects %boolean; "false" text:use-math-objects %boolean; "false">`

7.8.2 Object Index Entry Template

The object index entry template element determines the format of an index entry for a particular outline level.

XML Code:	`<text:object-index-entry-template>`
Rules:	
DTD:	`<!ELEMENT text:object-index-entry-template ( text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop )*>` `<!ATTLIST text:object-index-entry-template text:style-name %styleName; #REQUIRED>`

The attributes that you can associate with this element are the same as those that can be associated with the `<text:illustration-index-entry-template>` element. See Section 7.6.1 for detailed information about these attributes.

7.9 User-Defined Index

A user-defined index combines the capabilities of the indexes discussed earlier in this chapter. A user-defined index can gather entries from the following sources:

- Index marks
- Paragraphs formatted using particular paragraph styles
- Tables, images, or objects
- Text frames

The `<text:user-index>` element represents a user-defined index.

XML Code:	`<text:user-index>`
Rules:	
DTD:	`<!ELEMENT text:user-index (text:user-index-source, text:index-body)>` `<!ATTLIST text:user-index text:style-name %styleName; #IMPLIED>`

7.9.1 User-Defined Index Source

The `<text:user-index-source>` element can contain several attributes that determine how the index entries are gathered. It also supports an attribute that determines how the outline levels of the index entries are gathered.

The paragraph formats that are used as index marks are encoded in `<text:index-source-styles>` elements, just like in `<text:table-of-content-source>` elements.

XML Code:	`<text:user-index-source>`
Rules:	
DTD:	`<!ELEMENT text:user-index-source ( text:index-header-template?, text:user-index-entry-template*, text:index-source-styles* ) >`

The attributes you can use with `<text:user-index-source>` elements are:

- Use attributes, `text:use-*`
- Copy outline level
- Index scope

  This attribute specifies whether the index applies to the entire document or only to the current chapter.

DTD:	`<!ATTLIST text:user-index-source text:index-scope (document|chapter) "document">`

- Index name

  In order to support several user-defined indexes with different contents, user index marks have a `text:index-name` attribute. The same attribute can be used with a `<text:user-index-source>` element to specify which index marks apply to the current index.

DTD:	`<!ATTLIST text:user-index-source text:index-name %string; #IMPLIED>`

- Relative tab stop position

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

DTD:	`<!ATTLIST text:user-index-source text:relative-tab-stop-position %boolean; "true">`

Use Attributes

The `text:use-*` attributes specify which entries to include in the user-defined index. The attributes that you can specify are:

- `text:use-index-marks`
- `text:use-graphics`
- `text:use-tables`
- `text:use-floating-frames`
- `text:use-objects`

XML Code:	`text:use-index-marks`
	`text:use-graphics`
	`text:use-tables`
	`text:use-floating-frames`
	`text:use-objects`

Rules:	

DTD:	`<!ATTLIST text:user-index-source text:use-index-marks %boolean; "false" text:use-graphics %boolean; "false" text:use-tables %boolean; "false" text:use-floating-frames %boolean; "false" text:use-objects %boolean; "false">`

Copy Outline Levels

XML Code:	`text:copy-outline-levels`
Rules:	This attribute can have a value of `true` or `false`. If the value is `true`, the entries are gathered at the outline level of the source element to which they refer. If the value is `false`, all index entries gathered are at the top outline level. For example, if an image appears in Section 1.2.3, the entry for the image is located at outline level 3.
DTD:	`<!ATTLIST text:user-index-source text:copy-outline-levels %boolean; "false">`

7.9.2 User-Defined Index Entry Template

User index entry templates support entry elements for chapter number, page number, entry text, text spans, and tab stops.

XML Code:	`<text:user-index-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-chapter-number` • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop`
DTD:	`<!ELEMENT user-index-entry-template ( text:index-entry-chapter | text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop )*>`

The attributes that you can associate with the `<text:user-index-entry-template>` elements are:

- Template outline level
- Paragraph style

Template Outline Level

The `text:outline-level` attribute specifies to which outline level this entry configuration applies.

XML Code:	`text:outline-level`
Rules:	You cannot have several `<text:outline-level>` elements for the same outline level within a parent element.
DTD:	`<!ATTLIST text:table-of-content-entry-template text:outline-level %number; #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for the template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:table-of-content-entry-template text:style-name %styleName; #REQUIRED>`

7.10 Alphabetical Index

An alphabetical index gathers its entries solely from index marks.

XML Code:	`<text:alphabetical-index>`
Rules:	
DTD:	`<!ELEMENT text:alphabetical-index (text:alphabatical-index-source, text:index-body)>` `<!ATTLIST text:alphabetical-index text:style-name %styleName; #IMPLIED>`

7.10.1 Alphabetical Index Source

The `<text:alphabetical-index-source>` element specifies how the alphabetical index is generated.

XML Code:	`<text:alphabetical-index-source>`
Rules:	
DTD:	`<!ELEMENT text:alphabetical-index-source ( text:index-header-template?, text:alphabetical-index-entry-template* ) >`

The attributes you can associate with `<text:alphabetical-index-source>` elements are:

- Ignore case
- Main entry style name
- Alphabetical separators
- Combine entries attributes
- Use keys as entries
- Capitalize entries

- Comma separated entries
- Index scope

  This attribute specifies whether the index applies to the entire document or only to the current chapter.

DTD:	`<!ATTLIST text:alphabetical-index-source text:index-scope (document|chapter) "document">`

- Relative tab stop position

  This attribute specifies whether the position of tab stops are interpreted relative to the left margin or the left indent.

DTD:	`<!ATTLIST text:alphabetical-index-source text:relative-tab-stop-position %boolean; "true">`

Ignore Case

The `text:ignore-case` attribute determines whether or not the capitalization of words is ignored.

XML Code:	`text:ignore-case`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the capitalization is ignored and entries that are identical except for character case are listed as the same entries. If the value is `false`, the capitalization of words is not ignored.
DTD:	`<!ATTLIST text:alphabetical-index-source text:ignore-case %boolean; "false">`

Main Entry Style Name

The `text:main-entry-style-name` attribute determines the character style to use for main entries. Sub-entries are formatted using the default character style determined by the paragraph style of the entries.

XML Code:	`text:main-entry-style-name`
Rules:	
DTD:	`<!ATTLIST text:alphabetical-index-source text:main-entry-style-name %styleName; #IMPLIED>`

Alphabetical Separators

The `text:alphabetical-separators` attribute determines whether or not entries beginning with the same letter are grouped and separated from the entries beginning with the next letter, and so on.

XML Code:	`text:alphabetical-separators`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, all entries beginning with the same letter are grouped together. The index contains headings for each section, for example, A for all entries starting with the letter A, B for all entries starting with the letter B, and so on.

DTD:	`<!ATTLIST text:alphabetical-index-source text: alphabetical-separators %boolean; "false">`

Combining Entries

The OpenOffice.org software provides several options for dealing with the common situation where you have multiple index entries for the same word or phrase, as follows:

- Multiple entries for the same word can be combined into a single entry using the `text:combine-entries` attribute.
- If the combined entry contains a sequence of pages, the pages can be formatted:
  - As a range of numbers separated by a dash using the `text:combine-entries-with-dash` attribute
  - As the start number with a pp label, or the appropriate label for the chosen language, using the `text:combine-entries-with-pp` attribute

XML Code:	`text:combine-entries`
	`text:combine-entries-with-dash`
	`text:combine-entries-with-pp`

Rules:	

DTD:	`<!ATTLIST text:alphabetical-index-source`
	`text:combine-entries %boolean; "true"`
	`text:combine-entries-with-dash %boolean; "false"`
	`text:combine-entries-with-pp %boolean; "true">`

Example: Combining index entries

An index mark for the word *XML* occurs on pages 45, 46, 47, and 48. The entries can be formatted as follows:

Entry formatted as	Result
Separate entries	XML..... 45 XML..... 46 etc.
Simple combined entries	XML..... 45, 46, 47, 48
Entries combined with dash	XML..... 45-48
Entries combined with pp	XML..... 45pp

Use Keys as Entries

In addition to a keyword, index marks can have up to two keys.

XML Code:	`text:use-keys-as-entries`
Rules:	If the value of this atttribute is `true`, the keys are used as additional entries. If the value of this attribute is `false`, the keys are used as sub-entries.
DTD:	`<!ATTLIST text:alphabetical-index-source text:use-keys-as-entries %boolean; "false">`

Capitalize Entries

The `text:capitalize-entries` attribute determines whether or not the OpenOffice.org software capitalizes all entries in the index.

XML Code:	`text:capitalize-entries`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST text:alphabetical-index-source text: capitalize-entries %boolean; "false">`

Comma Separated Entries

The `text:comma-separated` attribute specifies how to treat multiple index entries. Instead of listing each index entry on a separate line, the OpenOffice.org software can list multiple entries on a single line separated by a comma.

XML Code:	`text:comma-separated`
Rules:	If the value of this attribute is `true`, multiple entries are listed on a single line separated by a comma. By default, the value of this attribute is `false` and each index entry is displayed on a separate line.
DTD:	`<!ATTLIST text:alphabetical-index-source text: comma-separated %boolean; "false">`

7.10.2 Alphabetical Index Entry Template

Alphabetical indexes support three levels; one level for the main index entry, and up to two additional levels for keys associated with the index entries. Alphabetical indexes also use an entry template for the alphabetical separator.

XML Code:	`<text:alphabetical-index-entry-template>`
Rules:	Alphabetical indexes support the following child elements: • `text:index-entry-chapter` • `text:index-entry-page-number` • `text:index-entry-text` • `text:index-entry-span` • `text:index-entry-tab-stop`
DTD:	`<!ELEMENT alphabetical-index-entry-template ( text:index-entry-chapter | text:index-entry-page-number | text:index-entry-text | text:index-entry-span | text:index-entry-tab-stop )*>`

The attributes that you can associate with the `<text:alphabetical-index-entry-template>` elements are:

- Template outline level

- Paragraph style

Template Outline Level

This attribute specifies whether the template applies to:

- One of the three levels 1,2,or 3

or

- The alphabetical separator

XML Code:	`text:outline-level`
Rules:	The value of this attribute can be `1`, `2`, `3`, or `separator`. You cannot have several `<text:outline-level>` elements for the same outline level within the same parent element.
DTD:	`<!ATTLIST text:table-of-content-entry-template text:outline-level (1|2|3|separator) #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for the template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:table-of-content-entry-template text:style-name %styleName; #REQUIRED>`

7.11 Bibliography

A bibliography index gathers its entries from bibliography index marks. The `<text:bibliography>` element represents a bibliography.

XML Code:	`<text:bibliography>`
Rules:	
DTD:	`<!ELEMENT text:bibliography (text:bibliography-source, text:index-body)>` `<!ATTLIST text:bibliography text:style-name %styleName; #IMPLIED>`

7.11.1 Bibliography Index Source

The `<text:bibliography-source>` element specifies how the bibliography is generated.

XML Code:	`<text:bibliography-source>`
Rules:	This element does not have an associated attributes.
DTD:	`<!ELEMENT text:alphabetical-index-source ( text:index-header-template?, text:bibliography-entry-template*) >`
Implementation limitation:	Bibliography do not currently support the common index source attributes described in Section 7.1.5.

7.11.2 Bibliography Entry Template

Bibliography entry templates support entry elements for bibliography data, text spans, and tab stops. There is one entry template element for each type of entry.

XML Code:	`<text:bibliography-entry-template>`
Rules:	This element supports the following child elements: • `text:index-entry-span` • `text:index-entry-tab-stop` • `text:index-entry-bibliography-data-field`
DTD:	`<!ELEMENT text:bibliography-entry-template ( text:index-entry-span | text:index-entry-tab-stop | text:index-entry-bibliography-data-field )*>`

The attributes that you can associate with the `<text:bibligraphy-entry-template>` elements are:

- Bibliography type
- Paragraph style

Bibliography Type

This attribute specifies to which type of bibliographical entry the template applies.

XML Code:	`text:bibliography-type`
Rules:	This attribute must be unique among all `<text:bibliography-type>` elements within the same parent element.
DTD:	`<!ATTLIST text:bibliography-entry-template text:outline-level (article | book | booklet | conference | custom1 | custom2 | custom3 | custom4 | custom5 | email | inbook | incollection | inproceedings | journal | manual | mastersthesis | misc | phdthesis | proceedings | techreport | unpublished | www) #REQUIRED>`

Paragraph Style

The `text:style-name` attribute specifies the paragraph style to use for this template.

XML Code:	`text:style-name`
Rules:	
DTD:	`<!ATTLIST text:table-of-content-entry-template text:style-name %styleName; #REQUIRED>`

CHAPTER 8

Chart Content

This chapter describes OpenOffice.org XML chart content. It contains the following sections:

- Introduction to Chart Documents
- Chart
- Title
- Subtitle
- Legend
- Plot Area
- Wall
- Floor
- Axis
- Series
- Categories
- Data Point
- Common Chart Properties

8.1 Introduction to Chart Documents

In OpenOffice.org XML, chart documents are always contained within other XML documents. There are two types of container chart documents:

- Documents that do not provide data for the chart
  The chart data is contained in a `<table:table>` element inside the `<chart:chart>` element.
- Documents that provide data for the chart
  The chart data may be contained in a `<table:table>` element in the parent document, for example, in a spreadsheet or text document.

To reference the correct table and table cells, you can use the `table:cell-range-address` attributes, which are applied to the `<chart:plot-area>` element that represents the visualization container of all data series in the chart.

8.2 Chart

The chart element represents an entire chart, including titles, a legend , and the graphical object that visualizes the underlying data called the plot area. The data underlying the chart is represented by a table element. This element may also exist for embedded charts that get the data from the container document. In this case the chart can be rendered without getting the data from the container document.

XML Code:	`<chart:chart>`
Rules:	
DTD:	`<!ELEMENT chart:chart ( chart:plot-area, chart:title?, chart:subtitle?, chart:legend? table:table? )>`

Class

The class attribute specifies the chart type. If you need to specify the type more precisely there are additional properties that you can assign using styles. For example, if you want to specify a 3D bar chart with horizontal bars, you must set the class attribute to `bar` and you must set the properties for three dimensional and horizontal arrangement in the corresponding style attribute.

XML Code:	`chart:class`
Rules:	The value of this attribute can be one of the main categories of chart types: `line`, `area`, `circle`, `ring`, `scatter`, `radar`, `bar`, or `stock`. The value `bubble` is not yet supported by the `<chart:chart>` element.
Implementation limitation:	Currently, this attribute is only supported by the `<chart:chart>` element.
DTD:	`<!ENTITY % chart-class "(line|area|circle|ring|scatter| radar|bar|stock|bubble|add-in)"> <!ATTLIST chart:chart chart:class %chart-class; #REQUIRED chart:add-in-name %string; #IMPLIED svg:width %length; #IMPLIED svg:height %length; #IMPLIED chart:column-mapping %string; #IMPLIED chart:row-mapping %string; #IMPLIED chart:style-name %style-name; #IMPLIED >`

The `add-in-name` attribute contains the service name of a chart type addin that is used if the `chart-class` is set to `add-in`. That is a UNO service that is registered in the office and is capable of rendering a chart.

The `svg:width` and `svg:height` attributes define the extent of the entire chart. Normally, the size of the chart is determined by the size of the window in which the chart is displayed. You can set these attributes as a reference size, so that positions and sizes in sub-elements can be adapted.

The `chart:column-mapping` and `chart:row-mapping` attributes contain if provided a list of indexes of series. The numbers define a reordering of data that comes from a container document that provides the data for the chart. A list of ascending numbers beginning with 1 has no effect. If you want to exchange two series, you must write the numbers in exchanged order, for example, `1 3 2 4,` to exchange the second and the third series. The numbering begins with 1. Note that you can only use one of the two attributes.

General Style Properties

The scale text property allows you to specify that all text objects in the chart should be scaled whenever the size of the chart changes .

XML Code:	`chart:scale-text`
Rules:	To enable scaling, set the value of this property to `true`.
DTD:	`<!ATTLIST style:properties chart:scale-text %boolean; "true" >`

To set the background properties for a `<chart:chart>` element, you can use the Fill Properties (described in Section 8.13.1) and the Stroke Properties (described in Section 8.13.2).

8.3 Title

The title element represents a main title object in a chart document.

XML Code:	`<chart:title>`
Rules:	This element can contain fixed text or it can contain a `<table:cell-address>` element pointing to the text that should be displayed as the title. This element can also be a sub-element of `chart:axis`, see Section 8.9. In this case the title is displayed beside the axis object.
Implementation limitation:	Currently, only inplace titles are supported.

DTD:	`<!ELEMENT chart:title text:p?>`
	`<!ATTLIST chart:title`
	`table:cell-range %cell-address; #IMPLIED`
	`svg:x %Coordinate; #IMPLIED`
	`svg:y %Coordinate; #IMPLIED`
	`chart:style-name %style-name; #IMPLIED >`

Properties

You can apply fill and stroke properties to the surrounding title box. See Sections 8.13.1 and 8.13.2 for more information. You can also apply text properties to the title text itself, see Section 8.13.3. You can also apply two alignment properties, Orientation and RotationAngle, see Section 8.13.4.

8.4 Subtitle

The subtitle element represents a subtitle which can be used for additional title information in a chart. The structure of the subtitle element is similar to that of the title element.

XML Code:	`<chart:subtitle>`
Rules:	
Implementation limitation:	Currently, only inplace titles are supported.

DTD:	`<!ELEMENT chart:subtitle text:p?>`
	`<!ATTLIST chart:subtitle`
	`table:cell-range %cell-address; #IMPLIED`
	`svg:x %Coordinate; #IMPLIED`
	`svg:y %Coordinate; #IMPLIED`
	`chart:style-name %style-name; #IMPLIED >`

Properties

You can apply the same properties to the `<chart:subtitle>` element as you can apply to the `<chart:title>` element. See Section 8.3 for more information.

8.5 Legend

The legend element determines whether or not a legend is displayed in the chart. You can set either a relative or an absolute position for the legend. The size of the legend is calculated automatically and therefore cannot be set as attribute.

XML Code:	`<chart:legend>`
Rules:	

DTD:	`<!ELEMENT chart:legend EMPTY>`
	`<!ATTLIST chart:legend`
	`chart:legend-position (top|left|bottom|right) "right"`
	`svg:x %Coordinate; #IMPLIED`
	`svg:y %Coordinate; #IMPLIED`
	`chart:style-name %style-name; #IMPLIED >`

Properties

You can apply fill and stroke properties to the legend object, see Sections 8.13.1and 8.13.2. You can also set text properties for the text inside the legend object, see Section 8.13.3.

8.6 Plot Area

The plot area element is a container for the graphics objects that represent chart data. The main purpose of the plot area is to be a container for the series elements that represent single data series, and the axis elements.

XML Code: `<chart:plot-area>`

Rules:

DTD:

```
<!ELEMENT chart:plot-area
          (chart:series*,
           chart:axis*,
           chart:wall?,
           chart:floor?) >
<!ATTLIST chart:plot-area
           svg:x %Coordinate; #IMPLIED
           svg:y %Coordinate; #IMPLIED
           svg:width %length; #IMPLIED
           svg:height %length; #IMPLIED
           table:cell-range-address %cell-range-address;
#IMPLIED
           chart:table-number-list %string; #IMPLIED
           chart:data-source-has-labels (none|row|column|
both) "none" >
           chart:style-name %style-name; #IMPLIED >
```

The style that you apply to the plot area element is used for all data elements contained inside the plot area, unless you specify extra styles in one of those sub elements. These data elements can be `<chart:series>` and `<chart:data-point>` elements.

If the position and size attributes are not specified, the values are calculated by the render application.

If a chart is embedded in a document that provides the data for the chart, the `table:cell-range-address` attribute reflects the ranges from which all the data for the chart comes. The range given here is interpreted by the chart as consecutive series. If the first row or column, or both contains labels, this is stated by the `chart:data-source-has-labels` attribute.

The `chart:table-number-list` is necessary for storing Calc documents in the old binary format. The attribute must be stored since this information can not be obtained elsewhere. The list consists of indexes of sheets of the Calc document. There is an index for each range in the `table:cell-range-address` element containing the index of the sheet containing the referred cells.

The only purpose of the style attribute is to store scene properties for three-dimensional charts.

Properties

If the chart is three-dimensional, you can apply scene properties to the plot area. See the chapter entitled *Graphic Content* for more information.

Subtype Properties

Use these properties to customize the basic chart type set in the `<chart:chart>` element.

XML Code:	`chart:stock-updown-bars`
	`chart:stock-with-volume`
	`chart:three-dimensional`
	`chart:deep`
	`chart:lines`
	`chart:percentage`
	`chart:solid-type`
	`chart:splines`
	`chart:stacked`
	`chart:symbol`
	`chart:vertical`
	`chart:lines-used`
	`chart:connect-bars`

Rules:	The `chart:deep` property is only relevant with the `chart:three-dimensional` property. The `chart:symbol` property is only relevant with the `chart:lines-used` property. The `chart:vertical` and `chart:connect-bars` properties are for bar charts only. The `chart:stock-updown-bars` and `chart:stock-with-volume` properties are only effective for stock charts.

DTD:	`<!ATTLIST style:properties`
	`chart:stock-updown-bars %boolean; "false"`
	`chart:stock-with-volume %boolean; "false"`
	`chart:three-dimensional %boolean; "false"`
	`chart:deep %boolean; "false"`
	`chart:lines %boolean; "false"`
	`chart:percentage %boolean; "false"`
	`chart:solid-type %chart-solid-type; "cuboid"`
	`chart:splines %nonNegativeInteger; "0"`
	`chart:stacked %boolean; "false"`
	`chart:symbol %integer; "-1"`
	`chart:vertical %boolean; "false"`
	`chart:lines-used %nonNegativeInteger; "0"`
	`chart:connect-bars %boolean; "false">`

8.7 Wall

The wall element can be contained in the plot area element. For two-dimensional charts, the wall element spans the entire plot area. For three-dimensional charts, the wall element usually consists of two perpendicular rectangles. You can use the width attribute to set the width of a wall for three-dimensional charts.

XML Code:	`<chart:wall>`
Rules:	
Implementation limitation:	OpenOffice.org Chart does not yet support the width of the wall. Walls are always drawn as zero-width rectangles.

DTD:	`<!ELEMENT chart:wall EMPTY>`
	`<!ATTLIST chart:wall`
	`svg:width %length; #IMPLIED`
	`chart:style-name %style-name; #IMPLIED >`

Properties

You can apply fill and stroke properties to a wall. See Sections 8.13.1 and 8.13.2 for more information.

8.8 Floor

The floor element can be contained in the plot area element. For three-dimensional charts, the floor element is present in addition to the wall element. The size of the floor is determined in respect of the size of the plot area, which is always a two-dimensional rectangle that serves as a bounding rectangle of the three-dimensional scene. You can use the width attribute to set the width of the floor.

XML Code:	`<chart:floor>`
Rules:	
Implementation limitation:	OpenOffice.org Chart does not yet support the floor thickness. The floor thickness is always a fixed width.

DTD:	`<!ELEMENT chart:floor EMPTY>`
	`<!ATTLIST chart:floor`
	`svg:width %length; #IMPLIED`
	`chart:style-name %style-name; #IMPLIED >`

Properties

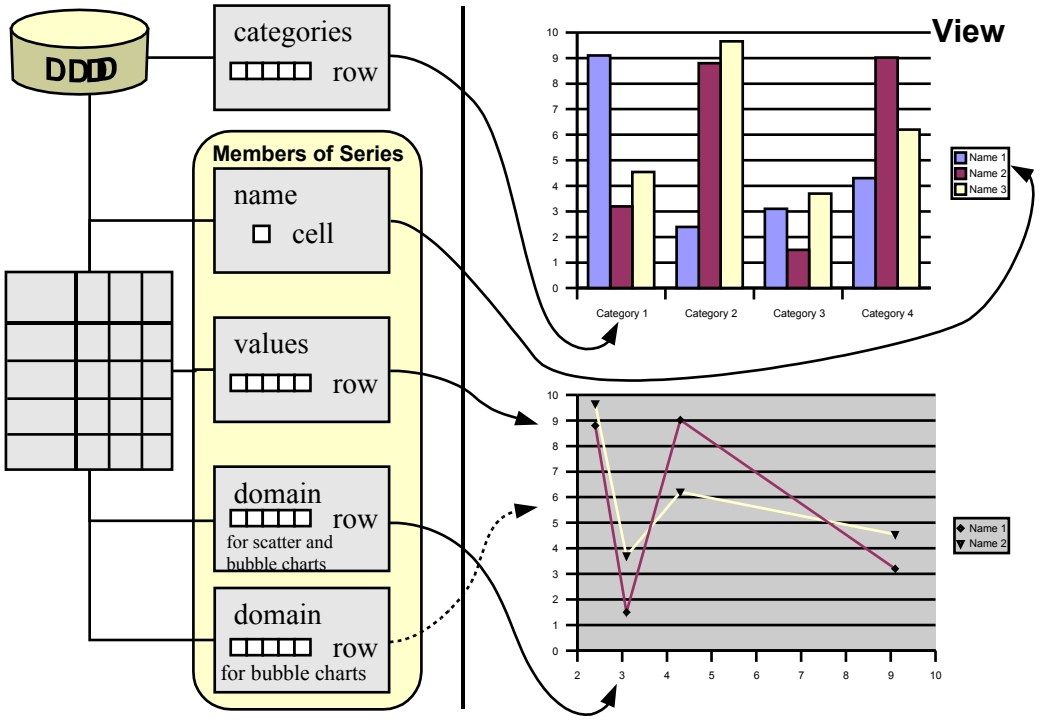
You can apply fill and stroke properties to a floor. See Sections 8.13.1 and 8.13.2 for more information.

8.9 Axis

The axis element mainly contains style information, in particular scaling information. Chart data is usually structured as follows:

- Several data series each consisting of a name, for example, the name of a company.
- Values, for example, the yield of the company in different years.
- One value in each series belongs to a category, for example, the year.

Figure: Chart data and its representation



XML Code:	`<chart:axis>`
Rules:	Use the `chart:axis-class` attribute to specify which type of data the axis is associated with.

DTD:	`<!ELEMENT chart:axis (chart:title?,chart:grid?)>`
	`<!ATTLIST chart:axis`
	`chart:axis-class (category|value|series|domain) #REQUIRED`
	`chart:axis-name %string #IMPLIED;`
	`chart:style-name %style-name; #IMPLIED >`

Current implementation limitations:

- Titles are only supported for a maximum of one axis per class.
- OpenOffice.org Chart only supports the following axes, the numbers in parenthesis indicating how far the value can extended:

Direction	2D		3D	
	Number	Attachable	Number	Attachable
`category`	2	0	1(2-4)	0
`value`	2	2	1(3-4)	1(3-4)
`series`	-	-	1(2-4)	0
`domain` (for scatter/bubble charts)	2	0	-	-

Defining Axes

Here are some guidelines for defining axes in a chart document:

1. The first axis you might want to apply to a chart is an axis representing categories. To do this, you insert an

axis element with the `axis:class` attribute set to `category`.

2. Next, you insert an axis showing a scale for your values. To do this, you insert an axis element with the `axis:class` attribute set to `value`.

3. In three-dimensional charts the names of the series, that are usually displayed in the legend, can also be displayed on an axis. To do this, insert an axis element with the `axis:class` attribute set to `series`.

4. If you have a scatter or bubble chart, each series has a domain of values specifying the x-coordinate, and y-coordinate for bubble charts, apart from the values that are to be visualized. For these types of charts, you can insert an axis element with the `axis:class` attribute set to `domain`, which will result in an axis similar to the axis described in Step 2.

5. A chart can contain more than one axis of the same type. For example, if you have two value axes, data series can be attached to either axis. This way data can be grouped for different scaling. To attach a specific axis to a series element you must refer to the axis by the `chart:axis-name` attribute. The axis name is required whenever you intend to attach data series to an axis. Otherwise the axis becomes a copy of an existing axis of the same class.

The position of an axis in a chart is determined by the render application and depends on the chart type. If you have a chart with horizontal bars, the render application usually paints the value axis on the bottom of the plot area. If you have two value axes, a render application might paint the second axis at the top of the plot area.

Note: If your data consists of numbers only and you want to create a scatter chart, the axis representing the values from the x-axis must have the `axis:class` attribute set to `domain` although your domain consists of values.

Example: Bar chart

In this example, there are two value axes and one axis has the name `primary-value`. You can attach a data series to that named axis by using the name. There is no data attached to the second axis, therefore you do not need to specify a name and the axis is just a copy of the first one.

```
<chart:chart chart:class="bar">
  <chart:title>
   <text:p>Title of my chart</text:p>
  </chart:title>
  <chart:plot-area>
    ...
    <chart:axis chart:axis-class="category" chart:axis-name="x"/>
    <chart:axis chart:axis-class="value" chart:axis-name="primary-value"/>
    <chart:axis chart:axis-class="value"/>  <!-- copy of previous axis -->
    ...
    <chart:series chart:values-address="Sheet1.A1:.A7"
     chart:attached-axis-name="primary-value"/>
    ...
  </chart:plot-area>
</chart:chart>
```

General Properties

You can apply stroke properties to axes, see Section 8.13.2. These properties affect all lines of the axis object. You also can apply text properties to axes, see Section 8.13.3. These properties affect the appearance of all text objects.

Number Format Properties

You can apply number format properties to axes, which affect the numbers displayed beside the axis. See Section

for information on number format properties. If you omit these properties, the standard number format is used. If the chart is embedded in a spreadsheet and you omit the number format properties, the number format is taken from the number format settings of the spreadsheet cells that contain the chart data.

DTD:	`<!ATTLIST style:properties`
	`style:data-style-name %style-name; #IMPLIED`
	`chart:link-data-style-to-source %boolean; "true" >`

The `chart:link-data-style-to-source` attribute can only be used in chart documents that reside in a document that provides the data for the chart. If the value of the attribute is `true`, the number format used for rendering the axis is the format that the container document suggests based on the selected cell range. For example, if you have a cell range with currencies all formatted in €, you will also get this format at this axis.

Visibility Property

To determine whether or not an axis object is visible, use the `chart:axis-visible` style property. This way, you can provide a chart with scaling information without displaying the axis object.

XML Code:	`chart:axis-visible`
Rules:	
DTD:	`<!ATTLIST style:properties chart:axis-visible %boolean; "true" >`

Scaling Properties

If a scaling attribute is omitted, the axis is set to adaptation mode. This means that the value is not set to a fixed value but may be changed by the render application if data changes. However, the `chart:axis-logarithmic` attribute is set to `false`.

DTD:	`<!ATTLIST style:properties`
	`chart:axis-minimum %float; #IMPLIED`
	`chart:axis-maximum %float; #IMPLIED`
	`chart:axis-interval-major %float; #IMPLIED`
	`chart:axis-interval-minor %float; #IMPLIED`
	`chart:axis-origin %float; #IMPLIED`
	`chart:axis-logarithmic %boolean; "false">`

Tickmark Properties

The tickmark properties allow you to specify the existence of tickmarks at an axis. The major marks are drawn with respect to the major interval that may be specified by the `chart:axis-interval-major` attribute. The minor tick marks refer to the `chart:axis-interval-minor` attribute. Inner marks are drawn towards the inside of the plot area, that is to the right for an axis displayed on the left hand side of the plot area, and to the left for an axis displayed on the right hand side of the plot area. Outer marks point in the opposite direction. If both properties are specified, one tick mark is drawn that crosses the axis.

XML Code:	`chart:axis-ticks-major-inner`
	`chart:axis-ticks-major-outer`
	`chart:axis-ticks-minor-inner`
	`chart:axis-ticks-minor-outer`

Rules:	

DTD:	`<!ATTLIST style:properties`
	`chart:axis-ticks-major-inner %boolean; "false"`
	`chart:axis-ticks-major-outer %boolean; "true"`
	`chart:axis-ticks-minor-inner %boolean; "false"`
	`chart:axis-ticks-minor-outer %boolean; "false" >`

Description Properties

The description properties influence the descriptive text underneath the axis object.

XML Code:	`chart:axis-show-text`
	`style:rotation-angle`
	`chart:axis-text-overlap`
	`chart:axis-text-break`

Rules:	

DTD:	`<!ATTLIST style:properties`
	`chart:axis-show-text %boolean; "true"`
	`style:rotation-angle %integer; "0"`
	`chart:axis-text-overlap %boolean; "false"`
	`chart:axis-text-break %boolean; "true" >`

8.9.1 Grid

Grids can be added to axis elements. If you apply a major grid to an axis, the major tickmarks are extended to gridlines. If a grid is minor, any minor tickmarks assigned to the axis are used.

XML Code:	`<chart:grid>`
Rules:	

DTD:	`<!ELEMENT chart:grid EMPTY>`
	`<!ATTLIST chart:grid grid:class (major|minor) "major"`
	`chart:style-name %style-name; >`

General Properties

You can apply stroke properties to grids, which affect the lines of the grid. See Section 8.13.2 for information on these stroke properties.

8.10 Series

The series element represents a data series in a chart.

XML Code:	`<chart:series>`
Rules:	
Implementation limitation:	OpenOffice.org Chart does not currently support the `chart:class` attribute for a series. You can only set this attribute for an entire chart.
DTD:	`<!ELEMENT chart:series (chart:class?, chart:domain*, chart:data-point* )>` `<!ATTLIST chart:series chart:values-cell-range-address %cell-range-address; #IMPLIED chart:label-cell-address %cell-address; #IMPLIED chart:class %chart-class; #IMPLIED chart:style-name %style-name; #IMPLIED >`

The `chart:values-cell-range-address` attribute allows you to specify a range that contains the values that should be visualized by this data series. The `chart:label-cell-address` attribute allows to provide a name for the series. If the chart requires more input data like scatter and bubble charts, you must define `chart:domain` sub-elements that mainly contain the `cell-range-address` of the corresponding data.

General Properties

You can apply fill and stroke properties for series, see Sections 8.13.1 and 8.13.2 for information. You can also apply text properties to the descriptive text underneath the series, see Section 8.13.3 for information.

8.10.1 Domain

For scatter and bubble charts, you must specify a domain for the series. For example, one `cell-range-address` value that points to the coordinate values for the scatter chart, or two `cell-range-address` values for the x and y coordinate values for bubble charts. For these chart types, you need at least one series with the necessary number of domain sub-elements. All other series can omit these, the first domain specified is used.

XML Code:	`<chart:domain>`
Rules:	
Implementation limitation:	In OpenOffice.org Chart, you can only give one range address, which may be compound, from which values are taken in a fixed order. For example, in scatter charts the first row/column specifies the x-values for all series, the second row/column represents the values of the first series and so on.
DTD:	`<!ELEMENT chart:domain EMPTY>` `<!ATTLIST chart:domain chart:coordinate-address %cell-range-address; #REQUIRED >`

8.11 Categories

The categories element represents the range of cell addresses that contains the captions for the categories contained in each series.

XML Code:	`<chart:categories>`
Rules:	
Implementation limitation:	OpenOffice.org Chart does not currently support the `chart:class` attribute for a series. You can only set this attribute for an entire chart.

DTD:	`<!ELEMENT chart:categories EMPTY>`
	`<!ATTLIST chart:categories`
	`table:cell-range-address %cell-range-address; >`

8.12 Data Point

If a single data point in a data series should have a specific appearance, the data point element is used to apply the required properties.

XML Code:	`<chart:data-point>`
Rules:	

DTD:	`<!ELEMENT chart:data-point>`
	`<!ATTLIST chart:data-point chart:index %nonNegativeInteger;`
	`#REQUIRED chart:style-name %style-name; >`

General Properties

You can apply fill and stroke properties to each data point object, see Sections 8.13.1 and 8.13.2. You can also apply text properties to the descriptive text located underneath the data points, see Section 8.13.3.

8.13 Common Chart Properties

The properties described in this section apply to all types of data representation objects, including the elements `<chart:plot-area>`, `<chart:series>`, and `<chart:data-point>`.

Properties are applied in a hierarchical manner. If a property is set in the `<chart:chart>` element, it applies to all data points contained in the chart. If the same property is set in a `<chart:series>` element, it only applies to the data points contained in that specific series. To set a formatting property for one data point only, you should set the property in the `<chart:data-point>` element.

8.13.1 Fill Properties

The fill properties apply to all solid objects like rectangles or circles. See Section Chapter 5 for information on fill properties.

8.13.2 Stroke Properties

The stroke properties apply to all line objects like the axis, grid, or linear parts of a rectangle or circle. See Chapter 5 for information stroke properties.

8.13.3 Text Properties

The text properties apply to all objects that display text, for example, the `legend`, `title`, `subtitle`, `axis`, `chart`, `series`, and `data-point`. See Chapter 3 for information on text properties.

8.13.4 Alignment Properties

The alignment properties described in this section apply to several text objects. They determine the way text is positioned inside the surrounding box.

Stacked Text

This property determines whether or not text is displayed vertically without rotating the letters.

XML Code:	`chart:text-stacked`
Rules:	The value of this property can be `ltr` if text goes from left to right or `ttb` if the text is stacked, that is goes from top to bottom.
DTD:	`<!ATTLIST style:properties fo:direction (ltr|ttb) #IMPLIED>`

Rotation Angle

The `style:rotation-angle` property specifies the value of a rotation angle in degrees. See Chapter 4 for information on using this property.

8.13.5 Data Label Properties

Data labels can be applied to data series and data points as well as to an entire chart. In the latter case, labels are shown for all data points. Data labels can consist of the following three parts:

- The value, which can be displayed as a percentage or the value itself.
- The label of the corresponding series.
- The legend symbol.

Value

This attribute represents the value of the data label.

XML Code:	`chart:data-label-number`
Rules:	
DTD:	`<!ATTLIST style:properties chart:data-label-number (none| value|percentage) "none" >`

Label

This attribute determines whether or not to display the label of the corresponding series.

XML Code:	`chart:data-label-text`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST style:properties chart:data-label-text %boolean; "false" >`

Legend Symbol

This attribute determines whether or not to display the legend symbol.

XML Code:	`chart:data-label-symbol`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST style:properties chart:data-label-symbol %boolean; "false" >`

8.13.6 Statistical Properties

Statistical properties can be applied to data series or to an entire chart. In the latter case, the properties apply to all series in the chart.

Mean Value

This attribute determines whether or not to display a line that represents the statistical mean value of all data points of a series.

XML Code:	`chart:mean-value`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ATTLIST style:properties chart:mean-value %boolean; "false" >`

Error Category

This attribute is used to determine which function is used to display error indicators at data points. You can set the following functions:

- Variance of the values of a series assuming an equal distribution.
- Standard-deviation of the values of a series assuming an equal distribution.
- Use a fixed percentage of each value
- Use a fixed percentage of the biggest value – this is called error-margin.
- Use fixed absolute values for both directions: positive and negative

XML Code:	`chart:error-category`
Rules:	If this attribute is set to any value other than `none`, error indicators are shown. To determine in which direction the indicators are pointing see the attributes `chart:error-upper-indicator` and `chart:error-lower-indicator`.
DTD:	`<!ATTLIST style:properties chart:error-category (none|variance|standard-deviation|percentage|error-margin| constant) "none" >`

Error Percentage

This attribute determines the percentage that is used to display error indicators for each data point of a series.

XML Code:	`chart:error-percentage`
Rules:	
DTD:	`<!ATTLIST style:properties chart:error-percentage %float; >`

Error Margin

This attribute determines the percentage that is used to display error indicators for the biggest value in a series.

XML Code:	`chart:error-margin`
Rules:	
DTD:	`<!ATTLIST style:properties chart:error-margin %float; >`

Constant Error Lower and Upper Limit

If you set the error category to `constant`, these attributes determine the absolute values in a positive and negative direction that are used to display the error indicators.

XML Code:	`chart:error-lower-limit` `chart:error-upper-limit`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`chart:error-lower-limit %nonNegativeInteger; #IMPLIED`
	`chart:error-upper-limit %nonNegativeInteger; #IMPLIED >`

Error Indicators

The `chart:error-lower-indicator` and `chart:error-upper-indicator` attributes determine in which direction indicators should be drawn.

XML Code:	`chart:error-lower-indicator` `chart:error-upper-indicator`
Rules:	

DTD:	`<!ATTLIST style:properties`
	`chart:error-lower-indicator %boolean; "false"`
	`chart:error-upper-indicator %boolean; "false" >`

Regression Curves

Use this attribute to display a regression for a series. With a regression you can approximate the data points you have in a series by a mathematical function. You can use one of the following models for approximation:

- Linear regression – approximate the values of the series using the model: $y = A \cdot x + B$.
- Logarithmic regression – approximate the values of the series using the model: $y = A \cdot \log(x) + B$.
- Exponential regression – approximate the values of the series using the model: $y = A \cdot e^{B \cdot x}$.
- Regression with a power function – approximate the values of the series using the model: $y = A \cdot x^{B}$.

XML Code:	`chart:regression-type`
Rules:	This property is only relevant in scatter charts, because regression needs both *x* and *y* values for calculation
DTD:	`<!ATTLIST style:properties chart:regression-type` `(none|linear|logarithmic|exponential|power) "none" >`

8.13.7 Rotation of Three-Dimensional Diagrams

The `<dr3d:transform>` element represents the rotation of a chart scene, that is the three-dimensional plot area. The `dr3d:transform` element represents a four-dimensional transformation matrix. See the chapter entitled *Graphic Content* for more information.

CHAPTER 9

User Interface Content

This chapter describes the OpenOffice.org XML representation of user interface components. Where possible, the user interface elements correspond to the Extensible User Interface Language (XUL) user interface elements. This enables users to easily migrate user interface elements from one language to another. This chapter contains the following sections:

- Menubars
- Accelerators
- Status bars
- Toolbars
- Events
- Images

You can store user interface content in an OpenOffice.org document or in a global user configuration oackage. Each OpenOffice.org component, such as Writer or Calc, has its own configuration for menus, accelerators, a status bar, and some toolbars. Some toolbars are not context-dependent like the function bar. Each menubar, accelerator, status bar, toolbar, or events configuration is stored as a subdocument in the global user configuration package or in an OpenOffice.org document. The name of the subdocument describes the content of the subdocument, for example `writerstatusbar.xml` contains the StarWriter status bar configuration.

9.1 Menubars

A menu is a list of commands, attributes, or states from which a user can choose. Menus are based on the interface principle of "See-and-Point". A menubar is an area in the user interface where menus reside.

The menubar spans up the container top-level element for all other menu elements embedded in it.
In the OpenOffice.org XML file format, the following basic rules apply to menubars:

- You must locate `<menu:menuitem>`, `<menu:menuseparator>`, and `<menu:menu>` elements in a `<menu:menupopup>` element.
- You can not nest `<menu:menubar>` elements.

9.1.1 Menubar

The `<menu:menubar>` element represents a menubar that an OpenOffice.org component uses and defines the contents and properties of the menubar.

XML Code:	`<menu:menubar>`
Rules:	This element is a container element for all other menu elements.

DTD:	`<!ELEMENT menu:menubar (menu:menu+)>`
	`<!ATTLIST menu:menubar`
	`menu:id %menu-id; #REQUIRED`
	`xmlns:menu CDATA #FIXED "http://openoffice.org/2001/menu">`

The attribute that you can associate with the `<menu:menubar>` is:

- Identifier (see Section 9.1.6)

9.1.2 Menu

The `<menu:menu>` element represents the title of a menu on the menubar. This element can be placed on a menubar or as a submenu inside a menupopup.

XML Code:	`<menu:menu>`
Rules:	Every menu must have a unique ID that represents the function of the menu.

DTD:	`<!ELEMENT menu:menu (menu:menupopup)>`
	`<!ATTLIST menu:menu`
	`menu:id %menu-id; #REQUIRED`
	`menu:label %menu-label; #IMPLIED>`

The attributes that you can associate with the `<menu:menu>` element are:

- Identifier (see Section 9.1.6)
- Label (see Section 9.1.6)

9.1.3 Menu Popup

The `<menu:menupopup>` element represents the popup box that appears when you click on the menu title.

XML Code:	`<menu:menupopup>`
Rules:	This element is a container element for menu items, menu separators, and submenus.
DTD:	`<!ELEMENT menu:menupopup (menu:menuitem | menu:menuseparator | menu:menu)+>`

There are no attributes associated with this element. The element can contain the following elements:

- `<menu:menuitem>`
- `<menu:menuseparator>`
- `<menu:menu>`

9.1.4 Menu Item

The `<menu:menuitem>` element represents an option on the menu.

XML Code:	`<menu:menuitem>`
Rules:	Every menu item must have a unique ID that represents the function of the item.

DTD:	`<!ELEMENT menu:menuitem EMPTY>`
	`<!ATTLIST menu:menuitem`
	`menu:id %menu-id; #REQUIRED`
	`menu:helpid CDATA #IMPLIED`
	`menu:label %menu-label; #IMPLIED>`

The attributes that you can associate with the `<menu:menuitem>` element are:

- Identifier (see Section 9.1.6)
- Label (see Section 9.1.6)
- Help identifier

Help Identifier

The `menu:helpid` attribute specifies the help identifier for the menu item.

XML Code:	`menu:helpid`
Rules:	The value of this attribute is a string that specifies the help identifier for the menu item.
DTD:	`<!ATTLIST menu:menuitem menu:helpid CDATA #IMPLIED>`

9.1.5 Menu Separator

The `<menu:menuseparator>` element separates groups of menu items.

XML Code:	`<menu:menuseparator>`
Rules:	
DTD:	`<!ELEMENT menu:menuseparator EMPTY>`

Example: Sample menubar

The following example shows a menubar with one menu called File and one submenu called New.

```
<menu:menubar menu:id="test">
  <menu:menu menu:id="slot:5300" menu:label="~File">
    <menu:menupopup>
      <menu:menu menu:id="slot:5400" menu:label="~New">
        <menu:menupopup>
          <menu:menuitem menu:id="macro:currency/euro" menu:label="~Euro
converter"/>
        </menu:menupopup>
      </menu:menu>
      <menu:menuitem menu:id="slot:5301" menu:label="~Open"/>
    </menu:menupopup>
  </menu:menu>
</menu:menubar>
```

9.1.6 Common Menubar Attributes

You can associate the attributes described in this section with the menubar elements.

Identifier

The `menu:id` attribute specifies a unique string that identifies the current menu component and defines the operation that is executed.

XML Code:	`menu:id`
Rules:	The value of this attribute must be a unique string within a menu. The attribute defines the operation that is executed after activation. Normally this is a command URL such as "slot:5503" that represents a function.
DTD:	`<!ENTITY % menu:id "CDATA">` `<!ATTLIST menu:menuitem menu:id %menu-id; #REQUIRED>`

Label

The `menu:label` attribute specifies the text that appears on the menu or menu item, such as File or Edit.

XML Code:	`menu:label`
Rules:	This attibute contains the text of a menu or menu item. You can use the special character "~" to specify the key that the user can press to activate the menu or menu item. This letter is typically underlined in the text label. For example, use "~File", to specify that the letter F activates the menu.
DTD:	`<!ENTITY % menu:label "CDATA">` `<!ATTLIST menu:menuitem menu:label %menu-label; #IMPLIED>`

9.2 Accelerators

An accelerator binds a menu command to a keyboard shortcut. This enables a user run a command by pressing a sequence of keys on the keyboard instead of accessing a menu item.

In the OpenOffice.org XML file format, the following basic rules apply to accelerators:

- You must include all `<accel:item>` elements in an `<accel:acceleratorlist>` element.
- You can not nest `<accel:acceleratorlist>` elements.

All accelerator definitions are contained in a subdocument of an OpenOffice.org XML document or in the `soffice.cfg` package for the global user settings.

9.2.1 Accelerator List

The `<accel:acceleratorlist>` element is a container element for all accelerator items.

XML Code:	`<accel:acceleratorlist>`
Rules:	All accelerator items are embedded in this element.

DTD:	`<!ELEMENT accel:acceleratorlist (accel:item*)>`
	`<!ATTLIST accel:acceleratorlist`
	`xmlns:accel CDATA #FIXED "http://openoffice.org/2001/accel"`
	`xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">`

There are no modifiable attributes associated with this element.

9.2.2 Accelerator Item

The `<accel:item>` element specifies the keyboard shortcut and the command to execute.

XML Code:	`<accel:item>`
Rules:	You must specify values for the `xlink:href` and the `accel:code` attributes.

DTD:	`<!ELEMENT accel:item EMPTY>`
	`<!ATTLIST accel:item`
	`accel:code CDATA #REQUIRED`
	`accel:shift %boolean; #IMPLIED`
	`accel:mod1 %boolean; #IMPLIED`
	`accel:mod2 %boolean; #IMPLIED`
	`xlink:href CDATA #REQUIRED>`

The attributes that you can associate with the `<accel:item>` element are:

- Key code
- Shift key state
- Modifier 1 key state
- Modifier 2 key state
- Command URL

Key Code

The `accel:code` attribute specifies the key code of the key that the user must press to activate the command.

XML Code:	`accel:code`
Rules:	The value of this attribute is a virtual key code that represents the key.
DTD:	`<!ATTLIST accel:item accel:code CDATA #REQUIRED>`

The following table lists the virtual key codes that are available. Some of the keys described in the table are not available on all keyboards.

KEY_0	KEY_1	KEY_2	KEY_3
KEY_4	KEY_5	KEY_6	KEY_7
KEY_8	KEY_9	KEY_A	KEY_B
KEY_C	KEY_D	KEY_E	KEY_F
KEY_G	KEY_H	KEY_I	KEY_J
KEY_K	KEY_L	KEY_M	KEY_N
KEY_O	KEY_P	KEY_Q	KEY_R
KEY_S	KEY_T	KEY_U	KEY_V
KEY_W	KEY_X	KEY_Y	KEY_Z
KEY_F1	KEY_F2	KEY_F3	KEY_F4
KEY_F5	KEY_F6	KEY_F7	KEY_F8
KEY_F9	KEY_F10	KEY_F11	KEY_F12
KEY_F13	KEY_F14	KEY_F15	KEY_F16
KEY_F17	KEY_F18	KEY_F19	KEY_F20
KEY_F21	KEY_F22	KEY_F23	KEY_F24
KEY_F25	KEY_F26	KEY_DOWN	KEY_UP
KEY_LEFT	KEY_RIGHT	KEY_HOME	KEY_END
KEY_PAGEUP	KEY_PAGEDOWN	KEY_RETURN	KEY_ESCAPE
KEY_TAB	KEY_BACKSPACE	KEY_SPACE	KEY_INSERT
KEY_DELETE	KEY_ADD	KEY_SUBTRACT	KEY_MULTIPLY
KEY_DIVIDE	KEY_POINT	KEY_COMMA	KEY_LESS
KEY_LESS	KEY_GREATER	KEY_EQUAL	KEY_OPEN
KEY_CUT	KEY_COPY	KEY_PASTE	KEY_UNDO
KEY_REPEAT	KEY_FIND	KEY_PROPERTIES	KEY_FRONT
KEY_CONTEXTMENU	KEY_MENU	KEY_HELP	

Shift Key State

The `accel:shift` attribute specifies whether or not the Shift key is required to activate the keyboard shortcut.

XML Code:	`accel:shift`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the user must press the Shift key to activate the shortcut.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST accel:item accel:shift %boolean; "false">`

Modifier 1 Key State

The `accel:mod1` attribute specifies whether or not the modifier one key is required to activate the keyboard shortcut. The modifier one key is system dependent as follows:

- Windows = Left or right control key

- UNIX = Left or right control key
- Mac = Command key

XML Code:	`accel:mod1`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST accel:item accel:mod1 %boolean; "false">`

Modifier 2 Key State

The `accel:mod2` attribute specifies whether or not the modifier two key is required to activate the keyboard shortcut. The modifier two key is system dependent as follows:.

- Windows = Alt key
- UNIX = Alt key
- Mac = Option key

XML Code:	`accel:mod1`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST accel:item accel:mod2 %boolean; "false">`

Command URL

The `xlink:href` attribute specifies the command to execute if the accelerator is activated.

XML Code:	`xlink:href`
Rules:	
DTD:	`<!ENTITY % url "CDATA">` `<!ATTLIST xlink:href %url; #REQUIRED>`

Example: Sample accelerator list with two accelerator items.

```
<accel:acceleratorlist>
  <accel:item accel:code="KEY_F4" xlink:href="slot:5501"/>
  <accel:item accel:code="KEY_Z" accel:mod1="true" xlink:href="slot:5701"/>
</accel:acceleratorlist>
```

9.3 Status Bars

A status bar is usually placed along the bottom of a window and provides status information to the user. In the OpenOffice.org XML file format, the following basic rules apply to status bars:

- You must locate a `<statusbar:statusbaritem>` element in a `<statusbar:statusbar>` element.
- You can not nest `<statusbar:statusbar>` elements.

A status bar is contained as a separate subdocument of an OpenOffice.org XML document or in the `soffice.cfg` package for the global user settings.

9.3.1 Status Bar

The `<statusbar:statusbar>` element is a container element for status bar items.

XML Code:	`<statusbar:statusbar>`
Rules:	You must include all status bar items in a this element.

DTD:	`<!ELEMENT statusbar:statusbar (statusbar:statusbaritem*)>`
	`<!ATTLIST statusbar:statusbar`
	`xmlns:statusbar CDATA #FIXED "http://openoffice.org/2001/statusbar"`
	`xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">`

9.3.2 Status Bar Item

The `<statusbar:statusbaritem>` element represents an information field that is displayed in the status bar.

XML Code:	`<statusbar:statusbaritem>`
Rules:	You must specify a valid value for the `xlink:href` attribute.

DTD:	`<!ELEMENT statusbar:statusbaritem EMPTY>`
	`<!ATTLIST statusbar:statusbaritem`
	`xlink:href CDATA #REQUIRED`
	`statusbar:align %alignment; #IMPLIED`
	`statusbar:style %style; #IMPLIED`
	`statusbar:autosize %boolean; #IMPLIED`
	`statusbar:ownerdraw %boolean; #IMPLIED`
	`statusbar:width %numeric; #IMPLIED`
	`statusbar:offset %numeric; #IMPLIED>`

The attributes that you can associate with the `<statusbar:statusbaritem>` element are:

- Status URL
- Alignment
- Style
- Autosize
- Ownerdraw
- Width
- Offset

Status URL

The `xlink:href` attribute specifies the state that is displayed for this status bar item.

XML Code:	`xlink:href`
Rules:	The value of this attribute must be a valid status URL. A reference of the currently available status URL can be found at ...???
DTD:	`<!ATTLIST statusbar:statusbaritem xlink:href CDATA #REQUIRED>`

Alignment

The `statusbar:align` attribute specifies how the information is aligned in the bounding box of the status bar.

XML Code:	`statusbar:align`
Rules:	The value of this attribute can be one of the following: • `left`: The information is aligned to the left border of the bounding box. • `center`: The information is aligned to the center of the bounding box • `right`: The information is aligned to the right border of the bounding box. The default value of this attribute is `center`.
DTD:	`<!ENTITY % alignment "(left|center|right)"> <!ATTLIST statusbar:statusbaritem statusbar:align %alignment; "center">`

Style

The `statusbar:style` attribute specifies how the status bar item is displayed in the bounding box of the status bar.

XML Code:	`statusbar:style`
Rules:	The value of this attribute can be one of the following: • `in`: The information is displayed in a box with a 3D effect. • `out`: The information is displayed in a box with a 3D effect . • `flat`: The information is displayed in a flat box without any 3D effect. The default value of this attribute is `in`.
DTD:	`<!ENTITY % style "(in|out|flat)"> <!ATTLIST statusbar:statusbaritem statusbar:style %style; "in">`

Autosize

The `statusbar:autosize` attribute specifies whether or not the size of the bounding box for the status bar item is set automatically by the status bar.

XML Code:	`statusbar:autosize`
Rules:	The value of this attribute can be `true` or `false`. If the value is `true`, the size of the bounding box is set automatically by the status bar. If the value is `false`, the `statusbar:width` attribute determines the size of the bounding box. The default value is `false`.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST statusbar:statusbaritem statusbar:autosize %boolean; "false">`

Ownerdraw

The `statusbar:ownerdraw` attribute specifies whether or not the status bar item is displayed using an external function.

XML Code:	`statusbar:ownerdraw`
Rules:	The value of this attribute can be `true` or `false`.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST statusbar:statusbaritem statusbar:ownerdraw % boolean; "false">`

Width

The `statusbar:width` attribute specifies the width of the bounding box for a status bar item.

XML Code:	`statusbar:width`
Rules:	This attribute is only valid if the value of the `statusbar:autosize` attribute is set to `false`. The value of the attribute must be a positive value. The default value is `0`.
DTD:	`<!ENTITY % numeric "CDATA">` `<!ATTLIST statusbar:statusbaritem statusbar:width %numeric; "0">`

Offset

The `statusbar:offset` attribute specifies the distance by which the text of the status bar item is offset on the x-axis.

XML Code:	`statusbar:offset`
Rules:	The value of this attibute is a numerical value in pixels. The default value of is `5` pixels.
DTD:	`<!ENTITY % numeric "CDATA">` `<!ATTLIST statusbar:statusbaritem statusbar:offset %numeric; "5">`

Example: Sample status bar

The following example shows a status bar with two information fields.

```
<statusbar:statusbar>
  <statusbar:statusbaritem xlink:href="slot:10000" statusbar:align="center"
statusbar:width="35"/>
  <statusbar:statusbaritem xlink:href="slot:21181" statusbar:align="left"
statusbar:autosize="true" statusbar:width="54"/>
</statusbar:statusbar>
```

9.4 Toolbars

A toolbar is a user interface component that provides visual access to the most important and frequently-used functions. Each item on the toolbar is represented as an image or a text label, or both.

The toolbar element spans up the container for all other toolbar elements embedded in it. In the OpenOffice.org XML file format, the following basic rules apply to toolbars:

- You must embed all `<toolbar:toolbaritem>`, `<toolbar:toolbarspace>`, `<toolbar:toolbarbreak>`, `<toolbar:toolbarseparator>` elements in a `<toolbar:toolbar>` element.
- You can not nest `<toolbar:toolbar>` elements.
- The `<toolbar:toolbarlayouts>` element is the top-level container element for layout information of every toolbar used inside OpenOffice.org.
- You must locate a `<toolbar:toolbarlayout>` element in a `<toolbar:toolbarlayouts>` element.
- You cannot nest `<toolbar:toolbarlayouts>` elements.

All toolbar definitions are contained in a subdocument of an OpenOffice.org XML document or can be found in the `soffice.cfg` package for the global user settings.

Every toolbar has additional layout information which is stored in a separate subdocument called `toolbarlayout.xml`. To completely describe a toolbar, there must be an valid entry in the `toolbarlayout.xml` subdocument.

9.4.1 Toolbar

The `<toolbar:toolbar>` element represents a toolbar with buttons and other items such as separators, spaces, and breaks.

XML Code:	`<toolbar:toolbar>`
Rules:	
DTD:	`<!ELEMENT toolbar:toolbar (toolbar:toolbaritem | toolbar:toolbarspace | toolbar:toolbarbreak | toolbar:toolbarseparator)*>` `<!ATTLIST toolbar:toolbar xmlns:toolbar CDATA #FIXED "http://openoffice.org/2001/toolbar" xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink" >`

You can include the following toolbar elements in the `<toolbar:toolbar>` element:

- `<toolbar:toolbaritem>`
- `<toolbar:toolbarspace>` (section missing)

- <toolbar:toolbarbreak> (section missing)
- <toolbar:toolbarseparator> (section missing)

9.4.2 Toolbar Item

The <toolbar:toolbaritem> element defines a button on the toolbar. The button represents a function.

XML Code:	<toolbar:toolbaritem>
Rules:	
DTD:	<!ELEMENT toolbar:toolbaritem EMPTY> <!ATTLIST toolbar:toolbaritem xlink:href CDATA #REQUIRED toolbar:visible %boolean; "true" toolbar:userdefined %boolean; "false" toolbar:text CDATA #IMPLIED toolbar:width %numeric; "0" toolbar:style CDATA #IMPLIED toolbar:bitmap CDATA #IMPLIED toolbar:helpid CDATA #IMPLIED>

The attributes that you can associate with the <toolbar:toolbaritem> element are:

- Command URL
- Visible
- User-defined
- Text
- Width
- Style
- Bitmap
- Help identifier

Command URL

The xlink:href attribute specifies the command that is executed if a user selects the toolbar item.

XML Code:	xlink:href
Rules:	
DTD:	<!ATTLIST toolbar:toolbaritem xlink:href CDATA #REQUIRED>

Visible

The toolbar:visible attribute specifies the minimum value that a user can enter.

XML Code:	`toolbar:visible`
Rules:	The value of this attribute can be `true` or `false`. Default value is “true”.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST toolbar:toolbaritem toolbar:visible %boolean; "true">`

User-defined

The `toolbar:userdefined` attribute specifies whether or not the toolbar item is user-defined.

XML Code:	`toolbar:userdefined`
Rules:	The value of this attribute can be `true` or `false`. The default value is `false`.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST toolbar:toolbaritem toolbar:userdefined %boolean; "false">`

Text

The `toolbar:text` attribute specifies the text for this toolbar item. The text is only displayed if the layout style of the toolbar is set to `text` or `symboltext`. See the `<toolbar:toolbarlayout>` element.

XML Code:	`toolbar:text`
Rules:	
DTD:	`<!ATTLIST toolbar:text CDATA #IMPLIED>`

Width

The `toolbar:width` attribute specifies the width of the toolbar item.

XML Code:	`toolbar:width`
Rules:	The default value of this attribute is `0`.
DTD:	`<!ENTITY % numeric "CDATA">` `<!ATTLIST toolbar:width %numeric; "0">`

Style

The `toolbar:style` attribute specifies additional styles for a toolbar item.

XML Code:	`toolbar:style`
Rules:	The value of this attribute can be a space-separated list of the following styles: • `radio`: This value indicates that the toolbar item belongs to a group of items. Only one item in the group can be activated. • `auto`: This value indicates that you can toggle the toolbar item between activated and deactivated. • `left`: This value is only valid if the layout style of the toolbar is set to `symboltext`. An icon is placed on left side of the text, like an entry in a taskbar. • `autosize`: This value indicates that the size of the toolbar item is automatically adjusted. • `dropdown`: This value indicates that the toolbar item supports a dropdown menu or toolbar for additional functions. • `repeat`: This value indicates that the toolbar item continues to execute the command while you click and hold the mouse button. The default value of this attribute is an empty string.
DTD:	`<!ATTLIST toolbar:style CDATA #IMPLIED>`

Bitmap

The `toolbar:bitmap` attribute specifies the name of a user-defined bitmap for the toolbar item. Currently the bitmap file must be located besides the toolbar subdocument.

XML Code:	`toolbar:bitmap`
Rules:	The value of this attribute can be empty or the name of an image file.
DTD:	`<!ATTLIST toolbar:bitmap CDATA #IMPLIED>`

Help Identifier

The `toolbar:helpid` attribute specifies an optional help identifier. Usually the `xlink:href` associates help text with the toolbar item. This attribute associates additional help text with the toolbar item.

XML Code:	`toolbar:helpid`
Rules:	
DTD:	`<!ATTLIST toolbar:helpid CDATA #IMPLIED>`

Example: Sample toolbar with two toolbar items, one separator, and an additional toolbar item.

```
<toolbar:toolbar>
  <toolbar:toolbaritem xlink:href="slot:5500"/>
  <toolbar:toolbaritem xlink:href="slot:5596" toolbar:width="300"/>
  <toolbar:toolbarseparator/>
  <toolbar:toolbaritem xlink:href="slot:5962" toolbar:style="dropdown"/>
</toolbar:toolbar>
```

9.4.3 Toolbar Layouts

The <toolbar:toolbarlayouts> element is the top-level container element for all specific toolbar layout elements.

XML Code:	<toolbar:toolbarlayouts>
Rules:	This element is a container element for toolbar layout elements.
DTD:	<!ELEMENT toolbar:toolbarlayouts (toolbar:toolbarlayout*)> <!ATTLIST toolbar:toolbarlayouts xmlns:toolbar CDATA #FIXED "http://openoffice.org/2001/toolbar">

There are no attributes associated with this element. The element can contain the following element:

- <toolbar:toolbarlayout>, see the next section.

9.4.4 Toolbar Layout

The <toolbar:toolbarlayout> element specifies the layout details for a specific toolbar.

XML Code:	<toolbar:toolbarlayout>
Rules:	This element must be embedded in a <toolbar:toolbarlayouts> element.
DTD:	<!ELEMENT toolbar:toolbarlayout EMPTY> <!ATTLIST toolbar:toolbarlayout toolbar:id CDATA #REQUIRED toolbar:floatingposleft %numeric; #IMPLIED toolbar:floatingpostop %numeric; #IMPLIED toolbar:floatinglines %numeric; "0" toolbar:dockinglines %numeric; "1" toolbar:align %alignment; "left" toolbar:visible %boolean; "false" toolbar:floating %boolean; "false" toolbar:style %style; "symbol">

The attributes that you can associate with the <toolbar:toolbarlayout> element are:

- Identifier
- Floating state
- Floating position
- Floating lines
- Docking lines
- Docking alignment
- Visible
- Style

Identifier

The `toolbar:id` attribute is a unique identifier for a toolbar layout entry. This identifier is used to find the corresponding toolbar definition. For example, if the value of `toolbar:id` is `writerobjectbar`, there must be a file next to this toolbar layouts file called `writerobjectbar.xml`, to attach both information.

XML Code:	`toolbar:id`
Rules:	This attribute must have a valid name to find the corresponding toolbar definition xml file.
DTD:	`<!ATTLIST toolbar:toolbarlayout toolbar:id CDATA #REQUIRED>`

Floating State

The `toolbar:floating` attribute specifies the initial state of the toolbar.

XML Code:	`toolbar:floating`
Rules:	The default value of this attribute is `true` or `false`. If the value is `true`, the `toolbar:floatingposleft` and `toolbar:floatingpostop` attributes determine the initial position of the toolbar. If the value is `false`, the toolbar is docked and the `toolbar:align` attribute determines the initial docking position.
DTD:	`<!ENTITY % boolean "(true|false)"> <!ATTLIST toolbar:toolbarlayout toolbar:floating %boolean; "false">`

Floating Position

The `toolbar:floatingposleft` attribute specifies the initial left position of the toolbar when the toolbar is in the floating state.

XML Code:	`toolbar:floatingposleft`
Rules:	This attribute is only valid if the value of the `toolbar:floating` attribute is `true`.
DTD:	`<!ENTITY % numeric "CDATA"> <!ATTLIST toolbar:toolbarlayout toolbar:floatingposleft % numeric; #IMPLIED>`

The `toolbar:floatingpostop` attribute specifies the initial top position of the toolbar when the toolbar is in the floating state.

XML Code:	`toolbar:floatingpostop`
Rules:	This attribute is only valid if the value of the `toolbar:floating` attribute is `true`.
DTD:	`<!ENTITY % numeric "CDATA"> <!ATTLIST toolbar:toolbarlayout toolbar:floatingpostop % numeric; #IMPLIED>`

Floating Lines

The `toolbar:floatinglines` attribute specifies the number of lines that are used to display the toolbar if the toolbar is floating.

XML Code:	`toolbar:floatinglines`
Rules:	You must assign a value to this attribute if the value of the `toolbar:floating` attribute is `true`. The default value of this attribute is `0`.
DTD:	`<!ENTITY % numeric "CDATA">` `<!ATTLIST form:formatted-text form:min-value CDATA #IMPLIED>`

Docking Lines

The `toolbar:dockinglines` attribute specifies the number of lines that are used to display the toolbar if the toolbar is docked.

XML Code:	`toolbar:dockinglines`
Rules:	The default value of this attribute is `1`.
DTD:	`<!ENTITY % numeric "CDATA">` `<!ATTLIST toolbar:toolbarlayout toolbar:dockinglines %numeric; "1">`

Docking Alignment

The `toolbar:align` attribute specifies the location at which the toolbar is initially docked.

XML Code:	`toolbar:align`
Rules:	This attribute is only relevant if the value of the `toolbar:floating` attribute is `false`. The value of this attribute can be one of the following: • `top`: The toolbar is docked at the top of the document window. • `bottom`: The toolbar is docked at the bottom of the document window. • `left`: The toolbar is docked at the left side of the document window. • `right`: The toolbar is docked at the right side of the document window. The default value of this attribute is `left`.
DTD:	`<!ENTITY % alignment "(top|bottom|left|right)">` `<!ATTLIST toolbar:toolbarlayout toolbar:align %numeric; "1">`

Visible

The `toolbar:visible` attribute specifies whether or not the toolbar is initially visible.

XML Code:	`toolbar:visible`
Rules:	The value of this attribute can be `true` or `false`. The default value is `false`.
DTD:	`<!ENTITY % boolean "(true|false)">` `<!ATTLIST toolbar:toolbarlayout toolbar:visible %boolean; "false">`

Style

The `toolbar:style` attribute specifies the display style of the toolbar items. This style does not influence the style of the user-defined toolbar items. See description of the toolbaritem attributes.

XML Code:	`toolbar:style`
Rules:	The value of this attribute can be one of the following: • `symbol`: The standard toolbar items are shown as buttons with an image to represent the function of the item. • `text`: The standard toolbar items are shown as buttons with text to describe the function of the item. • `symboltext`: The standard toolbar items are shown as buttons with an image and text to describe the function of the item. The default value of this attribute is `symbol`.
DTD:	`<!ENTITY % style "(symbol|text|symboltext)">` `<!ATTLIST toolbar:toolbarlayout toolbar:style %style; "symbol">`

Example: Sample toolbar layout definition with two toolbars called “functionbar” and “fullscreenbar”.

```
<toolbar:toolbarlayouts>
  <toolbar:toolbarlayout toolbar:id="functionbar" toolbar:align="top"
toolbar:visible="true" toolbar:style="symbol"/>
  <toolbar:toolbarlayout toolbar:id="fullscreenbar" toolbar:align="top"
toolbar:visible="true" toolbar:floating="false" toolbar:style="symbol"/>
</toolbar:toolbarlayouts>
```

9.5 Events

An event is used to execute a macro when a specific action is taken. For example, when you open a document, this action can execute a macro to search the document for specific words. In the OpenOffice.org XML file format, the following basic rules apply to events:

- You must embed every event in the `<event:events>` container element.
- You can not nest events.

Events that are defined in the global context are stored as a subdocument in the `soffice.cfg` package located in the user/configuration directory.

9.5.1 Events

The <event:events> element is a top-level container element for all events.

XML Code:	<event:events>
Rules:	
DTD:	<!ELEMENT event:events (event:event*)> <!ATTLIST event:events xmlns:event CDATA #FIXED "http://openoffice.org/2001/event"> <!ATTLIST event:events xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">

There are no attributes associated with the <event:events> element.

9.5.2 Event

The <event:event> element defines an event.

XML Code:	<event:event>
Rules:	
DTD:	<!ELEMENT event:event EMPTY> <!ATTLIST event:event event:name CDATA #REQUIRED event:language CDATA #REQUIRED event:library CDATA #REQUIRED event:macro-name CDATA #REQUIRED xlink:type CDATA #FIXED "simple" xlink:href CDATA #IMPLIED>

The attributes that you can associate with the <event:event> element are:

- Name
- Language
- Library
- Macro name
- An optional URL where the library can be found

Name

The event:name attribute specifies the name of the event that must occur to activate the macro.

XML Code:	event:name
Rules:	This attribute must have a value.
DTD:	<!ATTLIST event:event event:name CDATA #REQUIRED>

Language

The `event:language` attribute specifies the language used to write the macro. The only language that OpenOffice.org XML currently supports is StarBasic.

XML Code:	`event:name`
Rules:	This attribute must have a value. The only value that OpenOffice.org XML currently supports is `StarBasic`.
DTD:	`<!ATTLIST event:event event:language CDATA #REQUIRED>`

Library

The `event:library` attribute specifies the library where the macro is stored.

XML Code:	`event:library`
Rules:	This attribute must have a value.
DTD:	`<!ATTLIST event:event event:library CDATA #REQUIRED>`

Macro Name

The `event:macro-name` attribute specifies the name of the macro.

XML Code:	`event:macro-name`
Rules:	This attribute must have a value.
DTD:	`<!ATTLIST event:event event:macro-name CDATA #REQUIRED>`

Example: Events with one global event definition for a macro that is executed at application startup.

```
<event:events>
  <event:event event:name="OnStartApp" event:language="StarBasic" event:
library="StarOffice" event:macro-name="Tools.Strings.CheckDouble"/>
</event:events>
```

9.6 Images

You can place images on toolbar buttons and menu items. The images help the user to identify the function of a button or menu item. In the OpenOffice.org XML file format, the following basic rules apply to images:

- The images container element spans up the container top-level element for all other image elements embedded in it.
- You must embed all images, both internal and external, in the top-level container element.
- You must embed entry into an images element and an external entry into an externalimages element. You cannot mix them.
- There can be zero or more images element inside an images container but only one externalimages element.
- You can not nest images or externalimages.

9.6.1 Images Container

The <image:imagescontainer> element is a top-level container for all image definitions.

XML Code:	<image:imagescontainer>
Rules:	
DTD:	<!ELEMENT image:imagescontainer (image:images*, image:externalimages?)> <!ATTLIST image:imagescontainer xmlns:image CDATA #FIXED "http://openoffice.org/2001/image" xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">

There are no attributes associated with the <image:imagescontainer> element.

9.6.2 Images

The <image:images> element defines a container for an internal bitmap that contains one or more images.

XML Code:	<image:images>
Rules:	You must specify a value for the xlink:href attribute.
DTD:	<!ELEMENT image:images (image:entry*)> <!ATTLIST image:images xlink:href %uriReference; #REQUIRED xlink:type CDATA #FIXED "simple" image:maskmode %maskMode; "maskcolor" image:maskcolor %color; "#000000" image:maskurl %url; #IMPLIED>

The attributes that you can associate with the <image:images> element are:

- Reference
- Mask mode
- Mask color
- Mask bitmap

Reference

The xlink:href attribute specifies a uniform resource identifier (URI) for the image, which can contain one or more images.

XML Code:	xlink:href
Rules:	This attribute must have a value.
DTD:	<!ENTITY % uriReference "CDATA"> <!ATTLIST image:images xlink:href %uriReference; #REQUIRED>

Mask Mode

The image:maskmode attribute specifies the mask mode for the image.

XML Code:	image:maskmode
Rules:	The value of this attribute can be one of the following: • maskbitmap: The image contains a second bitmap with images that are used as masks. • maskcolor: A color is used as a transparent color. The default value of this attribute is maskcolor.
DTD:	<!ATTLIST event:event event:language CDATA #REQUIRED>

Mask Color

The image:maskcolor attribute specifies a color that is rendered as transparent.

XML Code:	image:maskcolor
Rules:	This value of this attribute is only valid if the value of the image:maskmode attribute is maskcolor. The value of the attribute must be in the format #rrggbb and the three color values must be hexadecimal. The default value is #000000 which is black.
DTD:	<!ENTITY % color "CDATA"> <!ATTLIST image:images image:maskcolor %color; "#000000">

Mask Bitmap

The image:maskbitmap attribute specifies a second bitmap with embedded images that is used as a mask for the images referenced by xlink:href.

XML Code:	image:maskurl
Rules:	This value of this attribute is only valid if the value of the image:maskmode attribute is maskbitmap.
DTD:	<!ENTITY % url "CDATA"> <!ATTLIST image:maskurl %url; IMPLIED>

9.6.3 Entry

The <image:entry> element defines one image that is embedded in an internal bitmap that is specified by a parent <image:images> element.

XML Code:	`<image:entry>`
Rules:	The `image:command` and `image:bitmap-index` attributes are required.

DTD:	`<!ELEMENT image:entry EMPTY>`
	`<!ATTLIST image:entry`
	`image:command %url; #REQUIRED`
	`image:bitmap-index CDATA #REQUIRED>`

You must associate the following attributes with the `<image:entry>` element:

- Command URL
- Bitmap index

Command URL

The `image:command` attribute specifies the URL for a command that is bound to the image.

XML Code:	`image:command`
Rules:	This attribute must have a value.
DTD:	`<!ENTITY % url "CDATA">` `<!ATTLIST image:command %url; #REQUIRED>`

Bitmap index

The `image:bitmap-index` attribute specifies the index to an image that is embedded in an internal bitmap. The image is addressed by the parent container element `<image:images>`.

XML Code:	`image:bitmap-index`
Rules:	The attribute must have a positive integer value.
DTD:	`<!ATTLIST image:entry image:bitmap-index CDATA #REQUIRED>`

9.6.4 External Images

The `<image:externalimages>` element is a container element for external bitmaps.

XML Code:	`<image:externalimages>`
Rules:	There can only be one `<image:externalimages>` element in the `<image:imagescontainer>` element.
DTD:	`<!ELEMENT image:externalimages (image:externalentry*)>`

There are no attributes associated with the `<image:externalimages>` element. The element can contain the following element:

- `<image:externalimages>`, see next section.

9.6.5 External Entry

The `<image:externalentry>` element defines an image entry that references an external bitmap.

XML Code:	`<image:externalimages>`
Rules:	The element must be contained in an `<image:externalimages>` element. The `image:command` and `xlink:href` attributes are required.

DTD:	`<!ELEMENT image:externalentry EMPTY>`
	`<!ATTLIST image:externalentry`
	`image:command %url; #REQUIRED`
	`xlink:href %uriReference; #REQUIRED`
	`xlink:type CDATA #FIXED "simple">`

The attributes that you must associate with the `<image:externalentry>` element:

- Command URL
- Reference

Command URL

The `image:command` attribute specifies the URL for a command that is bound to the image.

XML Code:	`image:command`
Rules:	This attribute must have a value.
DTD:	`<!ENTITY % url "CDATA">` `<!ATTLIST image:command %url; #REQUIRED>`

Reference

The `xlink:href` attribute specifies a URI for the bitmap. The bitmap is scaled to the correct dimensions when it is rendered on the screen.

XML Code:	`image:bitmap-index`
Rules:	The attribute must have a positive integer value.
DTD:	`<!ATTLIST image:entry image:bitmap-index CDATA #REQUIRED>`

Example: Image configuration

The following example shows an image configuration with one internal and one external image list. Both lists contain two image entries.

```
<image:imagescontainer>
  <image:images xlink:href="/bitmaps/functionbar.bmp image:
maskcolor=#808080>
    <image:entry image:command="slot:5500" image:bitmap-index="0"/>
    <image:entry image:command="slot:5510" image:bitmap-index="1"/>
  </image:images>
  <image:externalimages>
    <images:externalimage image:command="slot:5400" xlink:href="file:///f:/
bitmaps/file_new.bmp"/>
    <images:externalimage image:command="slot:5401" xlink:href="file:///f:/
bitmaps/file_save.bmp"/>
</image:imagescontainer>
```

CHAPTER 10

Dialog Content

This chapter describes the OpenOffice.org XML representation of dialogs. The dialog elements are based on and correspond closely to the Extensible User Interface Language (XUL) dialog elements. This enables easy migration of dialogs from one language to the other. This chapter contains the following sections:

- Dialog Containers
- Styles
- Controls
-

10.1 Dialog Containers

The <dlg:window> element is the top-level container element for all dialog-related elements including the following:

- Bulletin boards, see Section 10.1.1
- Styles, see Section 1.4
- Controls, see Section 6.2
- Events, see Section 2.11

XML Code:	<dlg:window>
Rules:	Each document contains only one <dlg:window> element.
DTD:	<!ELEMENT dlg:window (dlg:styles*, (%event;)*, dlg:bulletinboard*)> <!ATTLIST dlg:window dlg:id CDATA #REQUIRED dlg:left CDATA #IMPLIED dlg:top CDATA #IMPLIED dlg:width CDATA #IMPLIED dlg:height CDATA #IMPLIED dlg:style-id CDATA #IMPLIED dlg:title CDATA #IMPLIED dlg:page CDATA #IMPLIED dlg:tag CDATA #IMPLIED dlg:help-text CDATA #IMPLIED dlg:help-url CDATA #IMPLIED xmlns:dlg CDATA #FIXED "http://openoffice.org/2000/dialog" xmlns:script CDATA #FIXED "http://openoffice.org/2000/script" >

10.1.1 Bulletin Board

OpenOffice.org XML does not facilitate control layout management or automatic positioning. The same applies to <dlg:titledbox> elements which is another container element for controls.

XML Code:	<dlg:bulletinboard>
Rules:	This element can contain control elements and other bulletin board elements. The position of all elements contained in this element is relative to the parent <dlg:bulletinboard> element.
DTD:	<!ELEMENT dlg:bulletinboard ((dlg:bulletinboard| dlg:button| dlg:checkbox| dlg:combobox| dlg:menulist| dlg:radiogroup| dlg:titledbox| dlg:textfield| dlg:text| dlg:filecontrol| dlg:img| dlg:timefield| dlg:datefield| dlg:numericfield| dlg:currencyfield| dlg:patternfield| dlg:formattedfield| dlg:fixedline| dlg:progressmeter|dlg:scrollbar)*)> <!ATTLIST dlg:bulletinboard dlg:left CDATA #IMPLIED dlg:top CDATA #IMPLIED >

The attributes that you can associate with the <dlg:bulletinboard> element are:

- Left
- Top

Left

The dlg:left attribute specifies the left (x) position of the bulletin board.

XML Code:	dlg:left
Rules:	The value of this attribute is numeric.
DTD:	<!ATTLIST dlg:bulletinboard dlg:left CDATA #IMPLIED>

Top

The dlg:top attribute specifies the upper (y) position of the bulletin board.

XML Code:	`dlg:top`
Rules:	The value of this attribute is numeric.
DTD:	`<!ATTLIST dlg:bulletinboard dlg:top CDATA #IMPLIED>`

10.2 Styles

The `<dlg:styles>` element is a container element for `<dlg:style>` elements.

XML Code:	`<dlg:styles>`
Rules:	This element is contained in the top-level `<dlg:window>` element. This element can contain one or more `<dlg:style>` elements.
DTD:	`<!ELEMENT dlg:styles (dlg:style+)>`

10.2.1 Style

The `<dlg:style>` element defines the style attributes for a control. The style attributes that are associated with a control depends of the type of control to which the style is linked. For example, font attributes are only relevant for controls that contain text.

XML Code:	`<dlg:style>`
Rules:	This element does not contain further sub elements. The `dlg:style-id` attribute is required. All other attributes are optional. If you do not specify values for the optional attributes, the application defaults are used.

DTD:	`<!ELEMENT dlg:style EMPTY>`
	`<!ATTLIST dlg:style dlg:style-id CDATA #REQUIRED`
	`dlg:background-color CDATA #IMPLIED`
	`dlg:text-color CDATA #IMPLIED`
	`dlg:fill-color CDATA #IMPLIED`
	`dlg:border (none|3d|simple) #IMPLIED`
	`dlg:font-name CDATA #IMPLIED`
	`dlg:font-height CDATA #IMPLIED`
	`dlg:font-width CDATA #IMPLIED`
	`dlg:font-stylename CDATA #IMPLIED`
	`dlg:font-family (decorative|modern|roman|`
	`script|swiss|system) #IMPLIED`
	`dlg:font-charset (ansi|mac|ibmpc_437|`
	`ibmpc_850|ibmpc_860|ibmpc_861|`
	`ibmpc_863|ibmpc_865|`
	`system|symbol) #IMPLIED`
	`dlg:font-pitch (fixed|variable) #IMPLIED`
	`dlg:font-charwidth CDATA #IMPLIED`
	`dlg:font-weight CDATA #IMPLIED`
	`dlg:font-slant (oblique|italic|reverse_oblique|`
	`reverse_italic) #IMPLIED`
	`dlg:font-underline (single|double|dotted|`
	`dash|longdash|dashdot|`
	`dashdotdot|smallwave|wave|`
	`doublewave|bold|bolddotted|`
	`bolddash|boldlongdash|`
	`bolddashdot|bolddashdotdot|`
	`boldwave) #IMPLIED`
	`dlg:font-strikeout (single|double|bold|`
	`slash|x) #IMPLIED`
	`dlg:font-orientation CDATA #IMPLIED`
	`dlg:font-kerning (true|false) #IMPLIED`
	`dlg:font-wordlinemode (true|false) #IMPLIED`
	`dlg:font-type (raster|device|scalable) #IMPLIED`
	`>`

The attributes that you can associate with the `<dlg:style>` element are:

- `dlg:style-id`

Specifies the ID of the style so that you can associate the style with a control. This attribute is mandatory.

- `dlg:background-color`

Specifies the background color of a control.

- `dlg:text-color`

Specifies the text color of a control.

- `dlg:fill-color`

Specifies the fill color of a control.

- `dlg:border`

Specifies the border style a control.

- `dlg:font-name`

Specifies the name of the font to use for text in the control.

- `dlg:font-height`

Specifies the height of the font to use for text in a control.

- `dlg:font-width`

Specifies the width of the font to use for text in a control.

- `dlg:font-stylename`

Specifies the name of the font style to use for text in a control. For example, `Italic Bold`.

- `dlg:font-family` to

Specifies the font family to use for text in a control. For example, `Roman`.

- `dlg:font-charset` to

Specifies the character set to use for text in a control.

- `dlg:font-pitch` to

Specifies whether the pitch of the font for text in a control is `fixed` or `variable`.

- `dlg:font-charwidth` to

Specifies the character width of the font to use for text in a control.

- `dlg:font-weight` to

Specifies the font weight to use for text in a control.

- `dlg:font-slant` to

Specifies the slant to use for text in a control.

- `dlg:font-underline` to

Specifies the underline style to use for text in a control.

- `dlg:font-strikeout` to

Specifies the strikeout style to use for text in a control.

- `dlg:font-orientation` to

Specifies the orientation in degrees, to use for text in a control, where 0 is the baseline.

- `dlg:font-kerning` to

Specifies whether or not to use font kerning for the text in a control.

- `dlg:font-wordlinemode` to

Specifies whether or not only words are underlined in the text of a control.

- `dlg:font-type` to

Specifies the font type to use for text in a control.

For a complete list of the accepted values for these attributes, please refer to the previous table.

10.3 Controls

To build a user interface dialog, you use a collection of controls such as buttons, text boxes, check boxes, and drop-down lists. The following sections describe the control elements that are available; buttons, list boxes, combo boxes, choices, radio buttons, titled boxes, labels, text fields, images, file dialogs, currency fields, date fields, time fields, numeric fields, pattern fields, fixed lines, scroll bars, and progress bars.

Note: In the following sections:

- The `%default-attributes;` placeholder denotes the common attributes described in Section 10.3.1.
- The `%event` placeholder corresponds to an event element as described in Section 2.11.

10.3.1 Common Control Attributes

There are some attributes that you can associate with any control element. The following entity shows the common control attributes:

```
<!ENTITY % default-attributes "dlg:id CDATA #REQUIRED
                               dlg:left CDATA #REQUIRED
                               dlg:top  CDATA #REQUIRED
                               dlg:width CDATA #REQUIRED
                               dlg:height CDATA #REQUIRED
                               dlg:style-id CDATA #IMPLIED
                               dlg:tab-index CDATA #IMPLIED
                               dlg:disabled (true|false) #IMPLIED
                               dlg:printable (true|false) #IMPLIED
                               dlg:page CDATA #IMPLIED
                               dlg:tag CDATA #IMPLIED
                               dlg:help-text CDATA #IMPLIED
                               dlg:help-url CDATA #IMPLIED
                               ">
```

The purpose of these common control attributes is as follows:

- `dlg:id` is a mandatory string attribute of all controls

Specifies the unique ID of the control within a document. This attribute is mandatory.

- `dlg:style-id`

Contains a numeric string that specifies the ID number of a style to use for the control. This attribute references a `<dlg:style>` element with the same ID. This attribute is optional.

- `dlg:left`

Contains a numeric value that specifies the left position (x) of the control relative to the parent `<dlg:bulletinboard>` element. This attribute is mandatory.

- `dlg:top`

Contains a numeric value that specifies the upper position (y) of the control relative to the parent `<dlg:bulletinboard>` element. This attribute is mandatory.

- `dlg:width`

Contains a numeric value that specifies the width of the control. This attribute is mandatory.

- `dlg:height`

Contains a numeric value that specifies the height of the control. This attribute is mandatory.

- `dlg:tab-index`

Contains a numeric value that specifies the location in the tabbing order when the control gains focus. This attribute is optional.

- `dlg:disabled`

Specifies whether or not a control is disabled. The value of the attribute can be `true` or `false` and the default value is `false`. This attribute is optional.

- `dlg:printable`

Specifies whether or not a control is printable. The value of the attribute can be `true` or `false`. This attribute is optional.

- `dlg:page`

Contains a numeric value that specifies the page of multilayer dialogs. This attribute is optional.

- `dlg:tag`

Contains a text string that specifies additional user data, which can be evaluated by a basic script. This attribute is optional.

- `dlg:help-text`

Contains a text string that specifies help text for the control. The help text can be displayed as a tooltip. This attribute is optional.

- `dlg:help-url`

Contains a text string that specifies a URL that contains help text for the control. If you specify a URL, the URL is accessed and displayed when the user requests help. This attribute is optional.

10.3.2 Button

The `<dlg:button>` element describes a clickable button. When you click on a button, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:button>`
Rules:	This element is contained in one of the following elements: • `<dlg:bulletinboard>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:button ((%event;)*)>`
	`<!ATTLIST dlg:button %default-attributes;`
	`dlg:value CDATA #IMPLIED`
	`dlg:default (true|false) #IMPLIED`
	`dlg:tabstop (true|false) #IMPLIED`
	`dlg:button-type`
	`(standard|ok|cancel|help) #IMPLIED`
	`>`

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:value

Specifies the text label of the button. This attribute is optional.

- dlg:default

Specifies whether or not the button is the default button for the dialog. This attribute is optional.

- dlg:tabstop

Specifies whether or not the button is reachable via the Tab key. This attribute is optional.

- dlg:button-type

Specifies the type of the button. The default button type is standard. This attribute is optional.

10.3.3 Check Box

The <dlg:checkbox> element describes a control that you can check. When you modify the check box, a defined event element notifies an action, which triggers a macro.

XML Code:	<dlg:checkbox>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:checkbox ((%event;)*)> <!ATTLIST dlg:checkbox %default-attributes; dlg:value CDATA #IMPLIED dlg:checked (true|false) #IMPLIED dlg:tristate (true|false) #IMPLIED dlg:tabstop (true|false) #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:value

Specifies the text label of the check box. This attribute is optional.

- dlg:checked

Specifies whether or not the check box is checked. To specify state don't know, set the attribute dlg:tristate to true and omit the dlg:checked attribute. This attribute is optional.

- dlg:tristate

Specifies whether or not the check box can have three states; checked, unchecked, and don't know. If you specify don't know, you must omit the dlg:checked attribute. This attribute is optional.

- dlg:tabstop

Specifies whether or not the check box is reachable via the Tab key. This attribute is optional.

10.3.4 List Box and Combo Box

For list and combo boxes, there is a predefined set of elements from which a user can choose. The combo box includes a text field where the user can enter text, for example, to limit the set of items that the user can select from the list box.

List Box

The <dlg:menulist> element represents a list box. A list box is a list of items from which a user can choose one or more items. When a user chooses an item, a defined event element notifies an action, which triggers a macro. If the list box contains items, the <dlg:menulist> element contains a <dlg:menupopup> sub-element which contains one or more <dlg:menuitem> elements.

XML Code:	<dlg:menulist>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:menulist (dlg:menupopup?, (%event;)*)> <!ATTLIST dlg:menulist %default-attributes; dlg:tabstop (true|false) #IMPLIED dlg:spin (true|false) #IMPLIED dlg:multiselection (true|false) #IMPLIED dlg:linecount CDATA #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:tabstop

Specifies whether or not the list box is reachable via the Tab key.

- dlg:spin

Specifies whether or not the list box has a spin button.

- dlg:multiselection

Specifies whether or not a user can select more than one item from the list box.

- dlg:linecount

Specifies the number of lines of the selection that are visible. A user can use the scrollbar to access the remainder of the selection.

Combo Box

The <dlg:combobox> element represents a list box and text field combination. When a user chooses a item from the list box or enters text in the text field, a defined event element notifies an action, which triggers a macro. If the list box section of the combo box contains predefined items, the <dlg:combobox> element contains a <dlg:menupopup> sub-element which contains one or more <dlg:menuitem> elements.

XML Code:	<dlg:combobox>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:combobox (dlg:menupopup?, (%event;)*)> <!ATTLIST dlg:combobox %default-attributes; dlg:tabstop (true|false) #IMPLIED dlg:readonly (true|false) #IMPLIED dlg:autocomplete (true|false) #IMPLIED dlg:spin (true|false) #IMPLIED dlg:maxlength CDATA #IMPLIED dlg:linecount CDATA #IMPLIED dlg:value CDATA #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:tabstop

Specifies whether or not the combo box is reachable via the Tab key.

- dlg:readonly

Specifies whether or not a user can edit the contents of the combo box.

- dlg:autocomplete

Specifies whether or not the combo box autocompletes manual inputs.

- dlg:spin

Specifies whether or not the combo box has a spin button.

- dlg:maxlength

Specifies the maximum number of characters that is allowed for manual inputs.

- dlg:linecount

Specifies the number of lines of the selection that are visible. A user can use the scrollbar to access the remainder of the selection.

- dlg:value

Specifies the text in the text field.

Menu Popup

The <dlg:menupopup> element serves as a container element for <dlg:menuitem> elements.

XML Code:	<dlg:menupopup>
Rules:	This element is contained in one of the following elements: • <dlg:menulist> • <dlg:combobox>
DTD:	<!ELEMENT dlg:menupopup (dlg:menuitem+)>

Menu Item

The <dlg:menuitem> elements represent the items that you can select from a list box or from the list box section of a combo box.

XML Code:	<dlg:menuitem>
Rules:	This element is contained in the <dlg:menupopup> element and contains one or more style elements.
DTD:	<!ELEMENT dlg:menuitem EMPTY> <!ATTLIST dlg:menuitem dlg:value CDATA #REQUIRED dlg:selected (true|false) #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:value

Specifies the text label of the item.

- dlg:selected

Specifies whether or not the item is selected.

10.3.5 Radio Group and Titled Box

A radio group is a collection of items from which a user can select one item. If you specify that more than one radio item can be selected, it is undefined which one is selected.

Radio Group

The <dlg:radiogroup> element serves as a container element for one or more <dlg:radio> elements. There can be none or one radio item checked at a time.

XML Code:	<dlg:radiogroup>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:radiogroup (dlg:radio+)>

Radio Items

The <dlg:radio> element represents an item in a radio group. When a user selects a radio item, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:radio>`
Rules:	This element is contained in one of the following elements: • `<dlg:radiogroup>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:radio ((%event;)*)>`
	`<!ATTLIST dlg:radio %default-attributes;`
	`dlg:tabstop (true|false) #IMPLIED`
	`dlg:value CDATA #IMPLIED`
	`dlg:tristate (true|false) #IMPLIED`
	`dlg:checked (true|false) #IMPLIED`
	`>`

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the `<dlg:button>` element are:

- `dlg:tabstop`

Specifies whether or not the radio item is reachable via the Tab key.

- `dlg:value`

Specifies the text label of the radio item.

- `dlg:tristate`

Specifies whether or not the item can have three states; `checked`, `unchecked`, and `don't know`. To specify the value to `don't know`, set this to `true` and omit the `dlg:checked` attribute.

- `dlg:checked`

Specifies whether or not the radio item is checked. To specify the tristate value to `don't know`, set the attribute `dlg:tristate` to `true` and omit the `dlg:checked` attribute.

Titled Box

The `<dlg:titledbox>` element groups arbitrary controls and `<dlg:radio>` items and assigns a border and a title to the group. A titled box is sometimes called a group box. When a user moves the mouse pointer over a titled box, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:titledbox>`
Rules:	This element is contained in one of the following elements: • `<dlg:bulletinboard>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:titledbox`
	`(dlg:title?, dlg:radio*,`
	`(dlg:bulletinboard|dlg:button|dlg:checkbox|`
	`dlg:combobox|dlg:menulist|dlg:radiogroup|dlg:titledbox|`
	`dlg:textfield|dlg:text|dlg:filecontrol|dlg:img|`
	`dlg:timefield|dlg:datefield|dlg:numericfield|`
	`dlg:currencyfield|dlg:patternfield|dlg:formattedfield|`
	`dlg:fixedline|dlg:progressmeter|dlg:scrollbar)*,`
	`(%event;)*)`
	`>`
	`<!ATTLIST dlg:titledbox %default-attributes;>`

Title

The <dlg:title> element represents the title label of a <dlg:titledbox>.

XML Code:	<dlg:title>
Rules:	This element is optionally contained in a <dlg:titledbox> element.
DTD:	<!ELEMENT dlg:title EMPTY> <!ATTLIST dlg:title dlg:value CDATA #IMPLIED>

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:value

Specifies the text of the title label of the titled box.

10.3.6 Label

The <dlg:text> element describes a constant string of text. When a user moves the mouse pointer over a text label, a defined event element notifies an action, which triggers a macro.

XML Code:	<dlg:text>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:text ((%event;)*)> <!ATTLIST dlg:text %default-attributes; dlg:align (left|center|right|none) #IMPLIED dlg:multiline (true|false) #IMPLIED dlg:value CDATA #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:align

Specifies how the text is aligned, for example if it is aligned to the left or right of a specified dimension.

- dlg:multiline

Specifies whether or not the text can appear on more than one line.

- dlg:value

Specifies the text of the label.

10.3.7 Text Field

The <dlg:textfield> element represents a text field. A text field serves as an input control for strings of characters. When a user modifies a text field, a defined event element notifies an action, which triggers a macro.

XML Code:	<dlg:textfield>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:textfield ((%event;)*)> <!ATTLIST dlg:textfield %default-attributes; dlg:tabstop (true|false) #IMPLIED dlg:align (left|center|right|none) #IMPLIED dlg:readonly (true|false) #IMPLIED dlg:echochar CDATA #IMPLIED dlg:hard-linebreaks (true|false) #IMPLIED dlg:hscroll (true|false) #IMPLIED dlg:vscroll (true|false) #IMPLIED dlg:maxlength CDATA #IMPLIED dlg:multiline (true|false) #IMPLIED dlg:value CDATA #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:tabstop

Specifies whether or not the text field is reachable via the Tab key.

- dlg:align

Specifies how the input text is aligned, for example if it is aligned to the left or right of a specified dimension.

- dlg:readonly

Specifies whether or not a user can edit the text field.

- dlg:echochar

Specifies a special echo character. If this attribute is enabled, the same character is displayed in the output display regardless of which characters the user inputs. This attribute is useful for password input fields.

- dlg:hard-linebreaks

Specifies whether or not line break characters remain part of the input.

- dlg:hscroll

Specifies whether or not a horizontal scrollbar is displayed for the text field.

- dlg:vscroll

Specifies whether or not a vertical scrollbar is displayed for the text field.

- dlg:maxlength

Specifies the maximum number of input characters that is allowed.

- dlg:multiline

Specifies whether or not the text can appear on more than one line.

- dlg:value

Specifies the input text string of the field.

10.3.8 Image

The <dlg:img> element represents an image. When a user clicks on the image or move the mouse pointer over the image, a defined event element notifies an action, which triggers a macro.

XML Code:	<dlg:img>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:img ((%event;)*)> <!ATTLIST dlg:img %default-attributes; dlg:src CDATA #IMPLIED dlg:scale-image (true|false) #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:src

Specifies a URL for the image source file.

- dlg:scale-image

Specifies whether or not the image is scaled to suit the size of the control in which it resides.

10.3.9 File Dialog

The <dlg:filecontrol> element represents a control that displays a file dialog. The file dialog is used to select a file. When a user accesses a file dialog, a defined event element notifies an action, which triggers a macro.

XML Code:	<dlg:filecontrol>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:filecontrol ((%event;)*)> <!ATTLIST dlg:filecontrol %default-attributes; dlg:tabstop (true|false) #IMPLIED dlg:value CDATA #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:tabstop

Specifies whether the control is reachable via the Tab key.

- dlg:value

Specifies the name of the file that is selected using the file dialog.

10.3.10 Date Field

The <dlg:datefield> element represents a date field that a user can modify. When a user modifies a date field, a defined event element notifies an action, which triggers a macro.

XML Code: <dlg:datefield>

Rules: This element is contained in one of the following elements:

- <dlg:bulletinboard>
- <dlg:titledbox>

DTD:

```
<!ELEMENT dlg:datefield ((%event;)*)>
<!ATTLIST dlg:datefield %default-attributes;
                        dlg:tabstop (true|false) #IMPLIED
                        dlg:readonly (true|false) #IMPLIED
                        dlg:strict-format
                            (true|false) #IMPLIED
                        dlg:date-format
                            (system_short|system_short_YY|
                             system_short_YYYY|system_long|
                             short_DDMMYY|short_MMDDYY|
                             short_YYMMDD|short_DDMMYYYY|
                             short_MMDDYYYY|short_YYYYMMDD|
                             short_YYMMDD_DIN5008|
                             short_YYYYMMDD_DIN5008) #IMPLIED
                        dlg:value CDATA #IMPLIED
                        dlg:value-min CDATA #IMPLIED
                        dlg:value-max CDATA #IMPLIED
                        dlg:spin (true|false) #IMPLIED
                        >
```

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:tabstop

Specifies whether or not the date field is reachable via the Tab key.

- dlg:readonly

Specifies whether or not a user can edit the field.

- dlg:strict-format

Specifies whether or not the system checks the validity of the date while the user is modifying the field.

- dlg:date-format

Specifies the format in which the date value is displayed.

- dlg:value

Specifies the date value by a row of digits YYYYMMDD; a four digit year, followed by a two digit month, and a two digit day. For example, 20010330 represents March 30th, 2001.

- dlg:value-min

Specifies the lower date boundary of the field.

- dlg:value-max

Specifies the upper date boundary of the field.

- dlg:spin

Specifies whether or not the field has a spin button to increase or decrease the date value by a fixed step delta. See dlg:value-step.

- dlg:value-step

Specifies the numeric step delta when using the spin button.

10.3.11 Time Field

The <dlg:timefield> element represents a time field that a user can modify. When a user modifies the time field, a defined event element notifies an action, which triggers a macro.

XML Code:	<dlg:timefield>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:timefield ((%event;)*)> <!ATTLIST dlg:timefield %default-attributes; dlg:tabstop (true|false) #IMPLIED dlg:readonly (true|false) #IMPLIED dlg:strict-format (true|false) #IMPLIED dlg:time-format (24h_short|24h_long|12h_short| 12h_long|Duration_short| Duration_long) #IMPLIED dlg:value CDATA #IMPLIED dlg:value-min CDATA #IMPLIED dlg:value-max CDATA #IMPLIED dlg:spin (true|false) #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:tabstop

Specifies whether or not the field is reachable via the Tab key.

- dlg:readonly

Specifies whether or not a user can edit the field to modify the time.

- dlg:strict-format

Specifies whether or not the system checks the validity of the time setting while the user is modifying the field.

- dlg:time-format

Specifies the format in which the time is displayed.

- dlg:value

Specifies the time value by a row of digits `HHMMSSCC;` two digit hour, followed by minutes, seconds, and 10 milliseconds. For example, `12300000` represents 12:00 p.m.

- `dlg:value-min`

Specifies the lower time boundary of the field.

- `dlg:value-max`

Specifies the upper time boundary of the field.

- `dlg:spin`

Specifies whether or not the field has a spin button to increase or decrease the time value by a fixed step delta. See `dlg:value-step`.

- `dlg:value-step`

Specifies the numeric step delta when using the spin button.

10.3.12 Numeric Field

The `<dlg:numericfield>` element represents a numeric field that a user can modify. When a user modifies the field, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:numericfield>`
Rules:	This element is contained in one of the following elements: • `<dlg:bulletinboard>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:numericfield ((%event;)*)>`
	`<!ATTLIST dlg:numericfield %default-attributes;`
	`dlg:tabstop (true|false) #IMPLIED`
	`dlg:readonly (true|false) #IMPLIED`
	`dlg:strict-format`
	`(true|false) #IMPLIED`
	`dlg:decimal-accuracy CDATA #IMPLIED`
	`dlg:thousands-separator`
	`(true|false) #IMPLIED`
	`dlg:value CDATA #IMPLIED`
	`dlg:value-min CDATA #IMPLIED`
	`dlg:value-max CDATA #IMPLIED`
	`dlg:value-step CDATA #IMPLIED`
	`dlg:spin (true|false) #IMPLIED`
	`>`

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the `<dlg:button>` element are:

- `dlg:tabstop`

Specifies whether or not the field is reachable via the Tab key.

- `dlg:readonly`

Specifies whether or not a user can modify the value of the field.

- `dlg:strict-format`

Specifies whether or not the system checks the validity of the value while the user is modifying the value.

- `dlg:decimal-accuracy`

Specifies the decimal accuracy of the field.

- `dlg:thousands-separator`

Specifies whether or not a thousands separator is displayed in the field.

- `dlg:value`

Specifies the numeric value of the field.

- `dlg:value-min`

Specifies the lower numeric boundary of the field.

- `dlg:value-max`

Specifies the upper numeric boundary of the field.

- `dlg:spin`

Specifies whether or not the field has a spin button to increase or decrease the value by a fixed step delta. See `dlg:value-step`.

- `dlg:value-step`

Specifies the numeric step delta when using the spin button.

10.3.13 Currency Field

The `<dlg:currencyfield>` element represents a field that displays a currency value. When a user modifies the currency field, a defined event element notifies an action, which triggers a macro.

XML Code: `<dlg:currencyfield>`

Rules: This element is contained in one of the following elements:

- `<dlg:bulletinboard>`
- `<dlg:titledbox>`

DTD:

```
<!ELEMENT dlg:currencyfield ((%event;)*)>
<!ATTLIST dlg:currencyfield %default-attributes;
                            dlg:tabstop (true|false) #IMPLIED
                            dlg:readonly (true|false) #IMPLIED
                            dlg:currency-symbol CDATA #IMPLIED
                            dlg:strict-format
                                (true|false) #IMPLIED
                            dlg:decimal-accuracy
                                CDATA #IMPLIED
                            dlg:thousands-separator
                                (true|false) #IMPLIED
                            dlg:value CDATA #IMPLIED
                            dlg:value-min CDATA #IMPLIED
                            dlg:value-max CDATA #IMPLIED
                            dlg:value-step CDATA #IMPLIED
                            dlg:spin (true|false) #IMPLIED
                            >
```

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:tabstop

Specifies whether or not the field is reachable via the Tab key.

- dlg:readonly

Specifies whether or not a user can modify the value of the field.

- dlg:currency-symbol

Specifies the currency symbol.

- dlg:strict-format

Specifies whether or not the system checks the validity of a value while a user is modifying the field.

- dlg:decimal-accuracy

Specifies the decimal accuracy of the field.

- dlg:thousands-separator

Specifies whether or not a thousands separator is displayed in the field.

- dlg:value

Specifies the numeric value of the field.

- dlg:value-min

Specifies the lower numeric boundary of the field.

- dlg:value-max

Specifies the upper numeric boundary of the field.

- dlg:spin

Specifies whether or not the field has a spin button to increase or decrease the value by a fixed step delta. See dlg:value-step.

- dlg:value-step

Specifies the numeric step delta when using the spin button.

10.3.14 Pattern Field

The <dlg:patternfield> element represents a field that displays a pattern that a user can modify. When a user modifies the pattern field, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:patternfield>`
Rules:	This element is contained in one of the following elements: • `<dlg:bulletinboard>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:patternfield ((%event;)*)>`
	`<!ATTLIST dlg:patternfield %default-attributes;`
	`dlg:tabstop (true|false) #IMPLIED`
	`dlg:readonly (true|false) #IMPLIED`
	`dlg:strict-format`
	`(true|false) #IMPLIED`
	`dlg:edit-mask CDATA #IMPLIED`
	`dlg:literal-mask CDATA #IMPLIED`
	`dlg:value CDATA #IMPLIED`
	`>`

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the `<dlg:button>` element are:

- `dlg:tabstop`

Specifies whether or not the field is reachable via the Tab key.

- `dlg:readonly`

Specifies whether or not a user can edit the value of the field.

- `dlg:strict-format`

Specifies whether or not the system checks the validity of a pattern value while a user is modifying the field.

- `dlg:edit-mask`

Specifies the edit mask of the field.

- `dlg:literal-mask`

Specifies the literal mask of the field.

- `dlg:value`

Specifies the pattern.

10.3.15 Formatted Field

The `<dlg:formattedfield>` element represents a formatted field, which displays numbers, currencies, dates, and so on. When a user modifies a formatted field, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:formattedfield>`
Rules:	This element is contained in one of the following elements: • `<dlg:bulletinboard>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:formattedfield ((%event;)*)>`
	`<!ATTLIST dlg:formattedfield %default-attributes;`
	`dlg:tabstop`
	`(true|false) #IMPLIED`
	`dlg:readonly`
	`(true|false) #IMPLIED`
	`dlg:strict-format`
	`(true|false) #IMPLIED`
	`dlg:maxlength CDATA #IMPLIED`
	`dlg:spin (true|false) #IMPLIED`
	`dlg:align`
	`(left|center|right) #IMPLIED`
	`dlg:text CDATA #IMPLIED`
	`dlg:value-default CDATA #IMPLIED`
	`dlg:value-max CDATA #IMPLIED`
	`dlg:value-min CDATA #IMPLIED`
	`dlg:value CDATA; #IMPLIED`
	`dlg:format-code CDATA #IMPLIED`
	`dlg:format-locale CDATA #IMPLIED`
	`>`

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the `<dlg:button>` element are:

- `dlg:tabstop`

Specifies whether or not the field is reachable via the Tab key.

- `dlg:readonly`

Specifies whether or not a user can modify the value of the field.

- `dlg:strict-format`

Specifies whether or not the system checks the validity of a value while a user is modifying the field.

- `dlg:maxlength`

Specifies the maximum number of input characters allowed in the field.

- `dlg:spin`

Specifies whether or not the field contains a spin button to increase or decrease the value of the field.

- `dlg:align`

Specifies how the text is aligned.

- `dlg:text`

Specifies the text displayed in the control.

- `dlg:value-min`

Specifies the minimum effective numeric value of the field.

- `dlg:value-max`

Specifies the maximum effective numeric value of the field.

- `dlg:value`

Specifies the effective numeric value of the field.

- `dlg:value-default`

Specifies the default value of the field (numeric or string).

- `dlg:format-code`

Specifies the format of the field.

- `dlg:format-locale`

Specifies the format locale of the field, which is "ISO_LanguageCode; ISO_CountryCode; Variant". For example, the value of this attribute can be "`de;DE;WIN`". You can omit the final variant.

10.3.16 Fixed Line

The `<dlg:fixedline>` element represents a horizontal or vertical line, which can optionally have a label text. When a user moves the mouse pointer over a fixed line, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:fixedline>`
Rules:	This element is contained in one of the following elements: • `<dlg:bulletinboard>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:fixedline ((%event;)*)>`
	`<!ATTLIST dlg:fixedline %default-attributes;`
	`dlg:align`
	`(horizontal|vertical) #IMPLIED`
	`dlg:value CDATA #IMPLIED`
	`>`

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the `<dlg:button>` element are:

- `dlg:align`

Specifies the orientation of the line.

- `dlg:value`

Specifies the text of the label.

10.3.17 Scrollbar

The `<dlg:scrollbar>` element represents a horizontal or vertical scrollbar. When a user uses the scrollbar, a defined event element notifies an action, which triggers a macro.

XML Code:	<dlg:scrollbar>
Rules:	This element is contained in one of the following elements: • <dlg:bulletinboard> • <dlg:titledbox>
DTD:	<!ELEMENT dlg:scrollbar ((%event;)*)> <!ATTLIST dlg:scrollbar %default-attributes; dlg:align (horizontal|vertical) #IMPLIED dlg:curpos CDATA #IMPLIED dlg:maxpos CDATA #IMPLIED dlg:increment CDATA #IMPLIED dlg:pageincrement CDATA #IMPLIED dlg:visible-size CDATA #IMPLIED >

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the <dlg:button> element are:

- dlg:align

Specifies the orientation of the scrollbar.

- dlg:curpos

Specifies the current position of the scrollbar as a numeric value.

- dlg:maxpos

Specifies the maximum position of the scrollbar as a numeric value.

- dlg:increment

Specifies the increment amount for a single line move of the scrollbar as a numeric value.

- dlg:pageincrement

Specifies the increment amount for a block move of the scrollbar as a numeric value.

- dlg:visible-size

Specifies the visible size of the scrollbar.

10.3.18 Progress Bar

The <dlg:progressmeter> element represents a progress bar. When a user uses the progress bar, a defined event element notifies an action, which triggers a macro.

XML Code:	`<dlg:progressmeter>`
Rules:	This element is contained in one of the following elements: • `<dlg:bulletinboard>` • `<dlg:titledbox>`

DTD:	`<!ELEMENT dlg:progressmeter ((%event;)*)>`
	`<!ATTLIST dlg:progressmeter %default-attributes;`
	`dlg:value CDATA #IMPLIED`
	`dlg:value-min CDATA #IMPLIED`
	`dlg:value-max CDATA #IMPLIED`
	`>`

In addition to the common control attributes described in Section 10.3.1, the attributes that you can associate with the `<dlg:button>` element are:

- `dlg:value`

Specifies the current position of the progress bar as a numeric value.

- `dlg:value-min`

Specifies the lower boundary for progress bar positions as a numeric value.

- `dlg:value-max`

Specifies the upper boundary for progress bar positions as a numeric value.

10.4 Events

In addition to the obsolete `dlg:event` element, there are two other elements that enable you to link to events that occur as a result of using controls. The following example shows an event entity:

```
<!ENTITY % event "(script:event|script:listener-event|dlg:event)">
```

10.4.1 Events Specified Using Common Event Names

The `<script:event>` element represents an event that uses common event names, for example, `on-item-statechange`.

XML Code:	`<script:event>`
Rules:	This element can be contained in any element that represents a control.

DTD:	`<!ELEMENT script:event EMPTY>`
	`<!ATTLIST script:event script:location CDATA #IMPLIED`
	`script:language CDATA #REQUIRED`
	`script:macro-name CDATA #REQUIRED`
	`script:event-name CDATA #REQUIRED`
	`>`

The attributes that you can associate with the `<script:event>` element are:

- `script:location`

Specifies the location where the macro is executed. The only values currently accepted are `application` or `document`. The default is `document`.

- `script:language`

Specifies the programming language of the macro to be executed, for example `StarBasic`.

- `script:macro-name`

Specifies the name of a macro to execute when the event is fired.

- `script:event-name`

Specifies the name of the event that is occurring. Currently the following names are supported:

  - `on-approveaction`
  - `on-performaction`
  - `on-change`
  - `on-textchange`
  - `on-itemstatechange`
  - `on-focus`
  - `on-blur`
  - `on-keydown`
  - `on-keyup`
  - `on-mouseover`
  - `on-mousedrag`
  - `on-mousemove`
  - `on-mousedown`
  - `on-mouseup`
  - `on-mouseout`
  - `on-approvereset`
  - `on-reset`
  - `on-submit`
  - `on-approveupdate`
  - `on-update`
  - `on-load`
  - `on-startreload`
  - `on-reload`
  - `on-startunload`
  - `on-unload`
  - `on-confirmdelete`
  - `on-approverowchange`
  - `on-rowchange`
  - `on-approvecursormove`

- on-cursormove
- on-supplyparameter
- on-error

10.4.2 Explicit Events Specified by Listener

The <script:listener-event> element describes an event by an specific listener type method.

XML Code:	<script:listener-event>
Rules:	This element can be contained in any element that describes a control.
DTD:	<!ELEMENT script:listener-event EMPTY> <!ATTLIST script:listener-event script:location CDATA #IMPLIED script:language CDATA #REQUIRED script:macro-name CDATA #REQUIRED script:listener-type CDATA #REQUIRED script:listener-method CDATA #REQUIRED script:listener-param CDATA #IMPLIED >

The attributes that you can associate with the <script:listener-event> element are:

- script:location

Specifies the location where the macro is executed. The only values currently accepted are application or document. The default is document.

- script:language

Specifies the programming language of the macro to be executed, for example StarBasic.

- script:macro-name

Specifies the name of a macro to execute when the event is fired.

- script:listener-type

Specifies the name of the event listener type, for example com.sun.star.awt.XActionListener.

- script:listener-method

Specifies the destination event method of the listener, for example actionPerformed.

- script:listener-param

Specifies an optional parameter that qualifies the listener, for example when listening for property changes, you have to know the property name.

CHAPTER 11

Package Format

This chapter describes the package format used in OpenOffice.org. It contains the following sections:

- Introduction
- Zip File Structure
- Encryption
- Manifest File

11.1 Introduction

As XML has no native support for binary objects such as images, OLE objects, or other media types, OpenOffice.org uses a package file to store the XML content of a document together with its associated binary data. This package is a standard Zip file, whose structure is discussed below.

Information about the files contained in the package is stored in an XML file called the manifest file. The manifest file is always stored at the pathname META-INF/manifest.xml. The main pieces of information stored in the manifest are as follows:

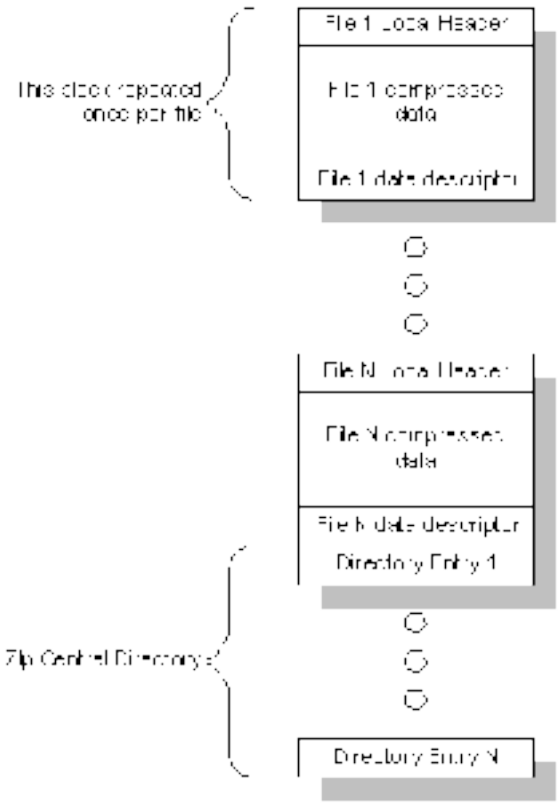
- A list of all of the files in the package.
- The media type of each file in the package.
- If a file stored in the package is encrypted, the information required to decrypt the file is stored in the manifest.

11.2 Zip File Structure

A Zip file starts with a sequence of files, each of which can be compressed or stored in raw format. Each file has a local header immediately before its data, which contains most of the information about the file, including time-stamps, compression method and file name. The compressed file contents immediately follow, and are terminated by an optional data descriptor. The data descriptor contains the CRC and compressed size of the file, which are frequently not available when writing the local file header. If these details were included, the data descriptor can be skipped.

Each file in the archive is laid down sequentially in this format, followed by a central directory at the end of the Zip archive. The central directory is a contiguous set of directory entries, each of which contains all the information in the local file header, plus extras such as file comments and attributes. Most importantly, the central directory contains pointers to the position of each file in the archive, which makes navigation of the Zip file quick and easy.

For more details about the Zip file format, see: http://www.pkware.com/appnote.html.



11.3 Encryption

The encryption process takes place in the following multiple stages:

1. A 20-byte SHA1 digest of the user entered password is created and passed to the package component.
2. The package component initializes a random number generator with the current time.
3. The random number generator is used to generate a random 8-byte initialization vector and 16-yte salt for each file.
4. This salt is used together with the 20-byte SHA1 digest of the password to derive a unique 128-bit key for each file. The algorithm used to derive the key is the PBKDF2 (see RFC 2989) with an iteration count of 1024.
5. The derived key is used together with the intialisation vector to encrypt the file using the Blowfish algorithm in cipher-feedback (CFB) mode.

Each file that is encrypted is compressed before being encrypted. To allow the contents of the package file to be verified, it is necessary that encrypted files are flagged as 'STORED' rather than 'DEFLATED'. As entries which are 'STORED' must have their size equal to the compressed size, it is necessary to store the uncompressed size in the manifest. The compressed size is stored in both the local file header and central directory record of the Zip file.

11.4 Manifest File

The elements and attributes in the manifest file are in the namespace: http://openoffice.org/2001/manifest.

Prefix	Description	Namespace
manifest	The manifest namespace contains all attributes and elements used in the manifest file.	http://openoffice.org/2001/manifest

11.4.1 Manifest Root Element

The root element is called manifest. The root element contains one fixed attribute which specifies the namespace as described above.

XML Code:	<manifest:manifest>
Rules:	This is the root element of the manifest file. It contains multiple 'file-entry' elements, each of which describes a single file in the package.
DTD:	<!ELEMENT manifest:manifest (manifest:file-entry+)*> <!ATTLIST manifest:manifest xmlns:manifest CDATA #FIXED "http://openoffice.org/2001/manifest">

11.4.2 File Entry

The <manifest:file-entry> element represents a single file within the package, and stores the files location in the package, the mime-type of the file and optionally the data required to decrypt this file.

XML Code:	<manifest:file-entry>
DTD:	<!ELEMENT manifest:file-entry (manifest:encryption-data?)>

The attributes associated with a <manifest:file-entry> are as follows:

- Full path
- Size
- Media type

Full Path

The manifest:full-path attribute describes the location of the file within the package.

XML Code:	manifest:full-path
DTD:	<!ATTLIST manifest:file-entry manifest:full-path CDATA #REQUIRED>

Size

The `manifest:size` attribute is only present if the file is stored in an encrypted format. The reason why this attribute is required is explained in Section 1.4.

XML Code:	`manifest:size`
Rules:	This attribute is only used for encrypted files.
DTD:	`<!ATTLIST manifest:file-entry manifest:size CDATA #IMPLIED>`

Media Type

The `manifest:media-type` attribute specifies the mime type of the specified file. For a full list of mime types see http://www.isi.edu/in-notes/iana/assignments/media-types/media-types As an example, all XML streams have the media type "text/xml".

XML Code:	`manifest:media-type`
DTD:	`<!ATTLIST manifest:file-entry manifest:media-type CDATA #IMPLIED>`

11.4.3 Encryption Data

The `<manifest:encryption-data>` element contains all of the information required to decrypt the file.

XML Code:	`<manifest:encryption-data>`
DTD:	`<!ELEMENT manifest:encryption-data (manifest:algorithm, manifest:key-derivation)>`

The `<encryption-data>` element contains the following elements:

- Algorithm
- Key Derivation

11.4.4 Algorithm

The `<manifest:algorithm>` element contains information about the algorithm used to encrypt the data.

XML Code:	`<manifest:algorithm>`
DTD:	`<!ELEMENT manifest:algorithm EMPTY>`

The attributes associated with `<manifest:algorithm>` are as follows:

- Algorithm name
- Initialization vector

Algorithm Name

The `manifest:algorithm-name` attribute specifies the name of the algorithm used to encrypt the file, and

also specifies in which mode this algorithm was used. Currently, the package component only supports the `Blowfish` algorithm in `CFB` mode.

XML Code:	`manifest:algorithm-name`
DTD:	`<!ATTLIST manifest:algorithm manifest:algorithm-name CDATA #REQUIRED>`

Initialization Vector

The `manifest:initialisation-vector` attribute specifies the 8 bytes used as an initialization vector to the stream cipher.

XML Code:	`manifest:initialisation-vector`
DTD:	`<!ATTLIST manifest:algorithm manifest:initialisation-vector CDATA #REQUIRED>`
Notes:	The initialization vector is an 8 byte binary sequence, and so is encoded in base64 when written to the manifest file.

11.4.5 Key Derivation

The `<manifest:key-derivation>` element contains the information that was used to derive the encryption key for this file from the user specified password.

XML Code:	`<manifest:key-derivation>`
DTD:	`<!ELEMENT manifest:key-derivation EMPTY>`

The attributes associated with the `<manifest:key-derivation>` element are as follows:

- Key derivation name
- Salt
- Iteration count

Key Derivation Name

The `manifest:key-derivation-name` attribute specifies the name of the algorithm used to derive the name. At this time, the package component only supports the use of the PBKDF2 key derivation method. For further details see RFC 2898.

XML Code:	`manifest:key-derivation-name`
DTD:	`<!ATTLIST manifest:key-derivation manifest:key-derivation-name CDATA #REQUIRED>`

Salt

The `manifest:key-derivation` attribute specifies the 16-byte sequence used as the 'salt' by the key derivation algorithm.

XML Code:	`manifest:key-derivation`
DTD:	`<!ATTLIST manifest:key-derivation manifest:salt CDATA #REQUIRED>`
Notes:	The salt is a 16 byte binary sequence, and thus is encoded in base64 before being written to the manifest file.

Iteration Count

The `manifest:key-derivation` attribute specifies the number of iterations used by the key derivation algorithm to derive the key.

XML Code:	`manifest:key-derivation`
DTD:	`<!ATTLIST manifest:key-derivation manifest:iteration-count CDATA #REQUIRED>`

Sample Manifest

This is a sample manifest file generated by build 6953 of OpenOffice.org. It is an encrypted OpenOffice.org Writer document that includes one graphic. The formatting has been slightly adjusted for clarity.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE manifest:manifest PUBLIC "-//OpenOffice.org//DTD Manifest 1.0//EN" "Manifest.dtd">
<manifest:manifest xmlns:manifest="http://openoffice.org/2001/manifest">
 <manifest:file-entry manifest:media-type="application/vnd.sun.xml.writer" manifest:full-path="/"/>
 <manifest:file-entry manifest:media-type="image/jpeg"
  manifest:full-path="Pictures/10000000000003200000258912EB1C3.jpg"
  manifest:size="66704">
  <manifest:encryption-data>
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
    manifest:initialisation-vector="T+miu403484="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
    manifest:iteration-count="1024" manifest:salt="aNYdmqv4cObAJSJjm4RzqA=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path="Pictures/"/>
 <manifest:file-entry manifest:media-type="text/xml"
  manifest:full-path="content.xml"
  manifest:size="3143">
  <manifest:encryption-data>
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
    manifest:initialisation-vector="T+miu403484="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
    manifest:iteration-count="1024" manifest:salt="aNYdmqv4cObAJSJjm4RzqA=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml"
  manifest:full-path="styles.xml" manifest:size="5159">
  <manifest:encryption-data>
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
    manifest:initialisation-vector="bChL2No5I+A="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
    manifest:iteration-count="1024" manifest:salt="/kfasyu7X0Ae+1uopdeCtA=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.xml"/>
 <manifest:file-entry manifest:media-type="text/xml"
  manifest:full-path="settings.xml" manifest:size="5317">
  <manifest:encryption-data>
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
    manifest:initialisation-vector="JQxEm6rD+4c="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
    manifest:iteration-count="1024" manifest:salt="PlpDaxloh4KUKx+v1g4V9g=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
</manifest:manifest>
```

Glossary

Term	*Definition*
attributes	Attributes are used to associate name-value pairs with elements. Each attribute specification has a name and a value. Attribute specifications may appear only within start-tags and empty-element tags.
automatic styles	A style that is automatically generated when the document is exported. The automatic style is assigned to objects such as paragraphs, so that a separation of content and layout is achieved.
body group	A body group is a group of rows or columns that do not repeat across pages. Typically, a body group contains the content of the table that is not part of the header.
characters	Parsed data is made up of characters, some of which form character data, and some of which form markup.
collapsing border model	This is a model of how to represent table borders. The collapsing border model means that when two adjacent cells have different borders, the wider border appears as the border between the cells. Each cell receives half of the width of the border.
column group	You can group columns in column groups. These groups specify whether or not to repeat a column on the next page. Column groups can be either body groups or header groups. A table must contain at least one column body group, but only one column header group.
comments	Comments may appear anywhere in a document outside other markup. In addition, they may appear within the document type declaration at places allowed by the grammar. They are not part of the character data of a document. An XML processor may, but need not, make it possible for an application to retrieve the text of comments.
common styles	The style that an OpenOffice.org user who doesn't care about the OpenOffice.org XML file format expects to be a style. The term *common styles* is used to differentiate in cases where there might be confusion with *automatic styles,* otherwise it is the same as *styles*.
content	The text between the start-tag and end-tag is the content of an *element.*
CSS	Cascading Style Sheet
CSS2	Cascading style sheets, level 2.
current file format version	This version stores all the information contained in the document without losing any information when the document is read again.
DDE	Dynamic Data Exchange.
document element	An OpenOffice.org XML document begins with a document *element*.

Term	*Definition*
document fields	The collective name for fields that display information about the current document or a specific part of the current document. For example, the author, the current page number, or the document creation date.
document root	The document root element is the primary element for defining the characteristics of an XML document.
document type declaration	The document type declaration can point to an external subset containing markup declarations, or can contain the markup declarations directly in an internal subset, or can do both. The DTD for a document consists of both subsets taken together.
drawing page	The main location for content in a drawing or presentation document.
DTD	The XML document type declaration contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or DTD.
elements	Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, and may have a set of attribute specifications.
entity	XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Entities may refer to other entities to cause their inclusion in the document.
EOL	End-of-line
external styles	Styles not contained within the XML file that contains the content.
generic identifier	The type of an element.
GI	See *generic identifier*.
header group	A header group is a group of rows or columns that repeat on each page if the table extends over several pages.
internal styles	Styles contained within the XML file that contains the content.
list level styles	A list level style is a specification containing a set of properties to apply to a list of a certain list level.
markup	Code description of the storage layout and logical structure of an XML document. Markup takes the form of start-tags, end-tags, empty-element tags, entity references, character references, comments, CDATA section delimiters, document type declarations, and processing instructions.
master page	The common background for a drawing page. Each drawing page must be linked to one master page.
metadata	Metadata is general information about the document, such as title, author, creation date, and so on.
name	A token beginning with a letter or one of a few punctuation characters, and continuing with letters, digits, hyphens, underscores, colons, or full stops, together known as name characters.
namespaces	Namespaces allow you to define specific names for elements. The purpose of namespaces is to avoid conflicts between documents from various authors with different naming conventions.
nmtoken	Name token; any mixture of name characters.
orphan	An orphan is a short line at the start of a paragraph, which when printed, appears alone at the bottom of the page.

Term	*Definition*
page master	A page-master specifies the size, border and orientation of a master page, q.v.
paragraph elements	The XML elements that are used to represent headings and paragraphs in text.
row group	You can group rows in row groups. These groups specify whether or not to repeat a row on the next page. Row groups can be either body groups or header groups. A table must contain at least one row body group, but only one row header group.
separating border model	This is a model of how to represent table borders. The separating border model means that borders appear within the cell that specifies the border.
sequence variable	A sequence variable numbers items such as tables or images in a document.
SGML	Standard Generalized Markup Language.
simple variable	A simple variable can have different values at different times in a document.
stroke properties	SVG stroke properties define graphic object line characteristics in OpenOffice.org Draw and OpenOffice.org Impress documents:
styles	The XML representations of the styles that are available in the OpenOffice.org user interface.
subtable	A subtable is a table within another table. It occupies one cell and no other content can appear in this cell. If a table cell contains a subtable, it cannot contain any paragraphs. Subtables are sometimes referred to as nested tables.
SVG	Scalar Vector Graphics. An XML language for creating vector graphics in Internet-viewable documents.
type	The *name* in the start-tags and end-tags gives the element type, see also *generic identifier*.
user variable	A user variable has the same value throughout a document.
valid	An XML document is valid if it has an associated document type declaration and if the document complies with the constraints expressed in it.
well-formed document	In a well-formed document, the logical and physical structures in an XML document are properly nested. Consequently, no start-tag, end-tag, empty-element tag, element, comment, processing instruction, character reference, or entity reference can begin in one entity and end in another.
W3C	World Wide Web Committee.
widow	A widow is a short line at the end of a paragraph, which when printed, appear alone at the top of the next page.
XLinks	Hyperlinks in XML documents, identified by the `xml:links` attribute. XLink governs how you insert links into your XML document, and where the link might points to.
XML	The eXtensible Markup Language (XML) is a subset of SGML described in the W3C XML Specification. The goal of XML is to enable generic SGML to be served, received, and processed on the Web in a way comparable to HTML. XML provides ease of implementation and interoperability with both SGML and HTML.
XML documents	A class of data objects described by XML. XML documents are conforming SGML documents.

Term	*Definition*
XML processor	A software module used to read XML documents and provide access to their content and structure.
XPath	XML Path Language; a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer.
XSL	Extensible Style Sheet Language; a stylesheet language for XML. When you use XSL all your documents are formatted the same way, no matter which application or platform they are on.
XSLT	XSLT is a language for transforming XML documents into other XML documents. XSLT is designed for use as part of XSL, or independently of XSL.
XPointer	XPointer governs the fragment identifier that can go on a URL when you're linking to an XML document from anywhere, for example from an HTML file. XPointer and XLinks are part of the same package.

Index

Index

-time 113
% cell-address 279
% cell-range-address 279
% cell-range-address-list 279
%database-table; 162
%num-format; 198
%value-attlist; 195
%variable-declarations; 183
%variable-fields; 183
<accel:acceleratorlist> 493
<accel:item> 493
<chart:axis> 478
<chart:categories> 483
<chart:chart> 472
<chart:data-point> 483
<chart:domain> 482
<chart:floor> 477
<chart:grid> 481
<chart:legend> 474
<chart:plot-area> 475
<chart:series> 482
<chart:subtitle> 474
<chart:title> 473
<chart:wall> 476
<config:config-item-map-entry> 115
<config:config-item-map-indexed> 115
<config:config-item-map-named> 115
<config:config-item-set> 115
<config:config-item> 114
<dc:creator> 37
<dc:date> 38
<dc:description> 36
<dc:language> 41
<dc:subject> 37
<dc:title> 36
<dlg:bulletinboard> 514
<dlg:button> 519
<dlg:checkbox> 520
<dlg:combobox> 521, 522
<dlg:currencyfield> 531
<dlg:datefield> 528
<dlg:filecontrol> 527
<dlg:fixedline> 535
<dlg:formattedfield> 533, 534
<dlg:img> 527
<dlg:menuitem> 521, 522, 523
<dlg:menulist> 521
<dlg:menupopup> 521, 522
<dlg:numericfield> 530
<dlg:patternfield> 532, 533
<dlg:progressmeter> 536, 537
<dlg:radio> 523, 524
<dlg:radiogroup> 523
<dlg:scrollbar> 535, 536
<dlg:style> 516
<dlg:styles> 515
<dlg:text> 525
<dlg:textfield> 525, 526
<dlg:timefield> 529
<dlg:title> 525
<dlg:titledbox> 524, 525
<dlg:window> 513
<draw:a> 105
<draw:applet> 89
<draw:caption> 340
<draw:circle> 336
<draw:connector> 338
<draw:contour-path> 93
<draw:contour-polygon> 93
<draw:control> 343
<draw:desc> 94
<draw:ellipse> 337
<draw:fill-image> 359
<draw:floating-frame> 93
<draw:g> 344
<draw:glue-point> 386
<draw:gradient> 356
<draw:hatch> 358
<draw:image> 86
<draw:line> 332
<draw:marker> 362

<draw:measure> 342
<draw:page-thumbnail> 343
<draw:page-thumbnail>, 327
<draw:page> 329
<draw:param> 92
<draw:path> 334
<draw:plugin> 91
<draw:polygon> 334
<draw:polyline> 333
<draw:rect> 331
<draw:text-box> 85
<draw:transparency> 360
<event:event> 507
<event:events> 507
<form:button> 415
<form:checkbox> 416
<form:column> 420
<form:combobox> 412
<form:control> 407
<form:file> 410
<form:fixed-text> 412
<form:form> 402
<form:formatted-text> 411
<form:frame> 418
<form:generic-control> 420
<form:grid> 419
<form:hidden> 419
<form:image-frame> 418
<form:image> 415
<form:item> 413
<form:listbox> 414
<form:option> 414
<form:password> 410
<form:properties> 434
<form:property-value> 436
<form:property> 435
<form:radio> 418
<form:text> 408
<form:textarea> 409
<image:entry> 510
<image:externalentry> 511
<image:externalimages> 511
<image:images> 509
<image:imagescontainer> 509
<manifest:algorithm> 544
<manifest:encryption-data> 544
<manifest:file-entry> 543
<manifest:key-derivation> 545
<manifest:manifest> 543
<menu:menu> 490
<menu:menubar> 490
<menu:menuitem> 491
<menu:menupopup> 490
<menu:menuseparator> 491
<meta:auto-reload> 39
<meta:creation-date> 38
<meta:document-statistic> 42
<meta:editing-cycles> 41
<meta:editing-duration> 42
<meta:generator> 36
<meta:hyperlink-behaviour> 40
<meta:initial-creator> 37
<meta:keywords> 37
<meta:print-date> 38
<meta:printed-by> 37
<meta:template> 38
<meta:user-defined> 42
<number:am-pm> 74
<number:boolean-style> 75
<number:boolean> 75
<number:currency-style> 67
<number:currency-symbol> 67
<number:date-style> 68
<number:day-of-week> 71
<number:day> 69
<number:era> 70
<number:fraction> 66
<number:hours> 73
<number:minutes> 73
<number:month> 69
<number:number-style> 64
<number:number> 65
<number:percentage-style> 68
<number:quarter> 72
<number:scientific-number> 66
<number:seconds> 74
<number:text-content> 76
<number:text-style> 75
<number:text> 75
<number:time-style> 72
<number:week-of-year> 71
<number:year> 70
<office:annotation> 275
<office:change-info> 113
<office:dde-source> 129
<office:events> 111, 432
<office:font-decls> 63
<office:forms> 402
<office:meta> 36
<office:settings> 114
<presentation:notes> 328
<presentation:placeholder> 387
<presentation:settings> 390
<presentation:show> 394
<script:event> 111, 537
<script:listener-event> 539
<script:script> 110
<style:background-image> 225
<style:column-sep> 232
<style:column> 231
<style:columns> 230
<style:drop-cap> 222

<style:font-decl> 63
<style:footer-left> 60
<style:footer-style> 60
<style:footer> 60
<style:footnote-layout> 61
<style:footnote-sep> 62
<style:handout-master> 327
<style:header-left> 60
<style:header-style> 60
<style:header> 60
<style:map> 49
<style:master-page> 326
<style:master-page> 58
<style:page-master> 52
<style:presentation-page-layout> 386
<style:properties> 45
<style:style> 46
<style:tab-stop> 219
<style:tab-stops> 219
<table:calculation-settings> 240
<table:cell-content-change> 253, 254
<table:cell-content-deletion> 245
<table:cell-range-source> 273
<table:change-deletion> 245
<table:change-track-table-cell> 252
<table:consolidation> 313
<table:content-validation> 280
<table:content-validations> 280
<table:covered-table-cell> 268
<table:cut-offs> 248
<table:data-pilot-field> 310
<table:data-pilot-level> 311
<table:data-pilot-member> 312
<table:data-pilot-members> 312
<table:data-pilot-subtotal> 312
<table:data-pilot-subtotals> 312
<table:data-pilot-table> 306
<table:data-pilot-tables> 305
<table:database-range> 296
<table:database-ranges> 296
<table:database-source-query> 300
<table:database-source-sql> 298
<table:database-source-table> 299
<table:dde-link> 315
<table:dde-links> 315
<table:deletion> 247
<table:deletions> 245
<table:dependence> 245
<table:dependences> 244
<table:detective> 276
<table:error-macro> 283
<table:error-message> 282
<table:filter-and> 293
<table:filter-condition> 294
<table:filter-or> 294
<table:filter> 292
<table:help-message> 282
<table:highlighted-range> 276
<table:insertion-cut-off> 249
<table:insertion> 246
<table:iteration> 243
<table:label-range> 289
<table:label-ranges> 288
<table:movement-cut-off> 249
<table:movement> 250
<table:named-expression> 291
<table:named-expressions> 290
<table:named-range> 290
<table:null-date> 242
<table:operation> 277
<table:scenario> 258
<table:shapes> 260
<table:sort-by> 301
<table:sort-groups> 303
<table:sort> 300
<table:source-cell-range> 309
<table:source-service> 308
<table:sub-table> 284
<table:subtotal-field> 304
<table:subtotal-rule> 304
<table:subtotal-rules> 302
<table:table-cell> 268
<table:table-column-group> 261
<table:table-column> 262
<table:table-columns> 262
<table:table-header-columns> 262
<table:table-header-rows> 265
<table:table-row-group> 264
<table:table-row> 265
<table:table-rows> 265
<table:table-source> 257
<table:table> 255
<table:tracked-changes> 244
<text:a> 122
<text:alpha-index-mark-end> 450
<text:alpha-index-mark-start> 449
<text:alpha-index-mark> 450
<text:alphabetical-index-entry-template> 466
<text:alphabetical-index-source> 463
<text:alphabetical-index> 463
<text:author-initials> 161
<text:author-name> 161
<text:bibliography-entry-template> 468
<text:bibliography-mark> 450
<text:bibliography-source> 468
<text:bibliography> 467
<text:bookmark-end> 124
<text:bookmark-ref> 181
<text:bookmark-start> 124
<text:bookmark> 124
<text:change-end> 235
<text:change-start> 234

<text:change> 235
<text:changed-region> 234
<text:chapter> 174
<text:conditional-text> 171
<text:creation-date> 167
<text:creation-time> 167
<text:creator> 170
<text:database-display> 163
<text:database-name> 166
<text:database-next> 164
<text:database-row-number> 165
<text:database-row-select> 165
<text:date> 153
<text:dde-connection-decl> 178
<text:dde-connection-decls> 178
<text:dde-connection> 180
<text:deletion> 236
<text:description> 167
<text:editing-cycles> 169
<text:editing-duration> 169
<text:endnote-body> 152
<text:endnote-citation> 151
<text:endnote-ref> 181
<text:endnote> 151
<text:endnotes-configuration> 149
<text:execute-macro> 178
<text:expression> 191
<text:file-name> 175
<text:footnote-body> 150
<text:footnote-citation> 149
<text:footnote-continuation-notice-backward> 148
<text:footnote-continuation-notice-forward> 148
<text:footnote-ref> 181
<text:footnote> 149
<text:footnotes-configuration> 146
<text:format-change> 236
<text:get-page-variable> 177
<text:h> 118
<text:hidden-paragraph> 173
<text:hidden-text> 172
<text:illustration-index-entry-template> 457
<text:illustration-index-source> 455
<text:illustration-index> 455
<text:index-body> 438
<text:index-entry-bibliography> 442
<text:index-entry-chapter-number> 441
<text:index-entry-chapter> 441
<text:index-entry-link-end> 445
<text:index-entry-link-start> 445
<text:index-entry-page-number> 442
<text:index-entry-span> 442
<text:index-entry-tab-stop> 443
<text:index-entry-template> 439
<text:index-entry-text> 442
<text:index-source-style> 446
<text:index-source-styles> 446
<text:index-title-template> 438
<text:initial-creator> 166
<text:insertion> 235
<text:keywords> 169
<text:line-break> 121
<text:linenumbering-configuration> 142
<text:linenumbering-seperator> 145
<text:list-header> 132
<text:list-item> 133
<text:list-level-style-bullet> 138
<text:list-level-style-image> 139
<text:list-level-style-number> 135
<text:list-level-style-numbering> 137
<text:list-style> 134
<text:modification-date duration> 170
<text:modification-time> 170
<text:object-index-entry-template> 460
<text:object-index-source> 459
<text:object-index> 458
<text:ordered-list> 131
<text:outline-level-style> 141
<text:outline-style> 141
<text:p> 118
<text:page-continuation> 157
<text:page-number> 155
<text:placeholder> 161
<text:print-date> 168
<text:print-time> 168
<text:printed-by> 168
<text:reference-mark-end> 125
<text:reference-mark-start> 125
<text:reference-mark> 125
<text:reference-ref> 181
<text:ruby-base > 202
<text:ruby-text > 203
<text:ruby> 201
<text:s> 120
<text:section-source> 128
<text:section> 127
<text:sender-city> 160
<text:sender-company> 159
<text:sender-country> 160
<text:sender-email> 158
<text:sender-fax> 159
<text:sender-firstname> 157
<text:sender-initials> 158
<text:sender-lastname> 158
<text:sender-phone-private> 159
<text:sender-phone-work> 159
<text:sender-position> 158
<text:sender-postal-code> 160
<text:sender-state-or-province> 160
<text:sender-street> 159
<text:sender-title> 158
<text:sequence-decl> 189
<text:sequence-ref> 181

<text:sequence> 190
<text:set-page-variable> 177
<text:span> 121
<text:subject> 169
<text:tab-stop> 121
<text:table-index-entry-template> 458
<text:table-index-source> 457
<text:table-index> 457
<text:table-of-content-entry-template> 453
<text:table-of-content-source> 452
<text:table-of-content> 452
<text:template-name> 176
<text:text-input> 192
<text:time> 154
<text:title> 168
<text:toc-mark-end> 448
<text:toc-mark-start> 447
<text:toc-mark> 448
<text:tracked-changes> 233
<text:unordered-list> 131
<text:user-defined> 167
<text:user-field-decl> 187
<text:user-field-get> 187
<text:user-field-input> 188
<text:user-index-entry-template> 462
<text:user-index-mark-end> 448
<text:user-index-mark-start> 448
<text:user-index-mark> 449
<text:user-index-source> 461
<text:user-index> 460
<text:variable-decl> 184
<text:variable-get> 185
<text:variable-input> 186
<text:variable-set> 184
<toolbar:toolbar> 499
<toolbar:toolbaritem> 500
<toolbar:toolbarlayout> 503
3D geometry properties 381
3D lighting properties 382
3D material properties 384
3D shapes 349
3D texture properties 383

A

accel:code 493
accel:mod1 494
accel:shift 494
alternative text 94
AM/PM 74
anchor position 199
anchor type 199
animation properties 375
annotation element 275
author fields 160
automatic reload 39
automatic style 31
automatic styles 44
automatic text indent 224
automatic update for styles 48
automatically order 79
axis 477

B

background attributes 330
background style, for drawing shapes 330
base cell address 291
bibliography data 442
bitmap 368
body element 33
bookmarks 124
Boolean 75
Boolean style 74
border and border line width for frames 100
break inside 218
bullet character 138
bullet level style 137

C

caption 340, 456
caption point attributes 341
categories 482
cell address entity 278
cell current Boolean value attribute 272
cell current currency value attribute 272
cell current date value attribute 272
cell current numeric value attribute 271
cell current string value attribute 272
cell current time value attribute 272
cell range address attribute 289, 290
cell range address entity 279
cell style attribute 269
cell value type attribute 271
center point attributes 336
chaining 86
change end 235
change position 235
change start 234
change tracking 112, 233, 243
changed region 234
chapter fields 174
chapter number 441
chart axis 477
chart data labels 484
chart data series 481
chart floor 477
chart legend 474

chart plot area 474
chart properties, common 483
chart subtitle 473
chart title 473
chart wall 476
chart:axis-show-text 481
chart:axis-text-break 481
chart:axis-text-overlap 481
chart:axis-ticks-major-inner 481
chart:axis-ticks-major-outer 481
chart:axis-ticks-minor-inner 481
chart:axis-ticks-minor-outer 481
chart:axis-visible 480
chart:class 472
chart:data-label-number 484
chart:data-label-symbol 485
chart:data-label-text 485
chart:scale-text 473
chart:stock-updown-bars
chart:stock-with-volume
chart:three-dimensional
chart:deep
chart:lines
chart:percentage
chart:solid-type
chart:splines
chart:stacked
chart:symbol
chart:vertical
chart:lines-used
chart:connect-bars 476
chart:text-stacked 484
charts 472
checked 524
circle 335
class attribute 30
clipping 103
color 203
column description 262
column formatting properties 319
column group element 261
column separator 232
column style attribute 262
columns 100
common styles 31
conditional text fields 170
connector 338
consecutive numbering 135
continue numbering 132
contour 93
contour wrapping mode 102
contour-only wrapping 101
control ID 421
controls 407
controls, button 415
controls, check box 416
controls, column grid 420
controls, combo box 412
controls, combo box list items 413
controls, file selection 410
controls, fixed text input 412, 507, 509, 510, 511
controls, formatted text input 411
controls, frame 418
controls, generic 420
controls, graphical button 415
controls, grid for table data 419
controls, hidden 419
controls, image frames 418
controls, list box 414
controls, list box items 414
controls, password 410
controls, radio buttons 417
controls, text box 408
conventions 24
country 78, 209
creation date and time 38
creator 37
crossing out 204
currency language and country 67
currency style 66
currency symbol 67
current file format 34
current file format version 34
current version 34

D

data pilot tables 305
data point 483
data style formatting properties 76
data style mappings 76
data styles 64
data styles namespace 64
database fields 162
database range 296
database source query 300
database source table 299
date 38
date adjustment 154
date fields 153
date style 68
date value 154
day of the month 69
day of week element 71
DDE connection attributes 256
DDE connection fields 180
DDE connections 178
decimal places 74, 82
decimal replacement 83
delay 40
deletion 235, 247
delimiter character 219

description 36
disclaimer 23
display duplicates attribute 293
display factor 83
display levels 137
dlg: 520, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537
dlg:align 525, 526, 534, 535, 536
dlg:autocomplete 522
dlg:button-type 520
dlg:checked 520, 524
dlg:curpos 536
dlg:currency-symbol 532
dlg:date-format 528
dlg:decimal-accuracy 531, 532
dlg:default 520
dlg:echochar 526
dlg:edit-mask 533
dlg:format-code 535
dlg:format-locale 535
dlg:hard-linebreaks 526
dlg:hscroll 526
dlg:increment 536
dlg:linecount 521, 522
dlg:literal-mask 533
dlg:maxlength 522, 526, 534
dlg:maxpos 536
dlg:multiline 525, 526
dlg:multiselection 521
dlg:pageincrement 536
dlg:readonly 522, 526, 528, 529, 530, 532, 533, 534
dlg:scale-image 527
dlg:selected 523
dlg:spin 521, 522, 529, 530, 531, 532, 534
dlg:src 527
dlg:strict-format 528, 529, 530, 532, 533, 534
dlg:tabstop 520, 521, 522, 524, 526, 527, 528, 529, 530, 532, 533, 534
dlg:text 534
dlg:thousands-separator 531, 532
dlg:time-format 529
dlg:tristate 520, 524
dlg:value 520, 522, 523, 524, 525, 526, 527, 528, 529, 531, 532, 533, 535, 537
dlg:value-default 535
dlg:value-max 529, 530, 531, 532, 534, 537
dlg:value-min 528, 530, 531, 532, 534, 537
dlg:value-step 529, 530, 531, 532
dlg:visible-size 536
dlg:vscroll 526
document creation date and time 38
document description 36
document fields 153
document keywords 37
document modification date 38
document root element 28, 29
document subject 36
document template name fields 176
document title 36
document type attribute 30
domain 482
dr3d:ambient-color 384
dr3d:back-scale 382
dr3d:backface-culling 382
dr3d:depth 382, 383
dr3d:diffuse-color 384
dr3d:edge-rounding 381
dr3d:edge-rounding-mode 382
dr3d:emissive-color 384
dr3d:horizontal-segments 381
dr3d:lighting-mode 382
dr3d:normals-direction 383, 384
dr3d:shininess 384
dr3d:specular-color 384
dr3d:texture-filter 384
dr3d:texture-generation-mode-x 383
dr3d:texture-generation-mode-y 383
dr3d:texture-kind 383
dr3d:vertical-segments 381
draw:angle 358
draw:archive 90
draw:auto-grow-width
draw:auto-grow-height 372
draw:blue 374
draw:border 358
draw:caption-point-x 341
draw:caption-point-y 341
draw:code 90
draw:codebase 90
draw:color-mode 373
draw:contrast 373
draw:corner-radius 332, 341
draw:distance 359
draw:end-color 357
draw:end-glue-point 340
draw:end-guide 377
draw:end-intensity 357
draw:end-line-spacing-horizontal 376
draw:end-line-spacing-vertical 376
draw:end-shape 340
draw:fill 367
draw:fill-color 367
draw:fill-gradient-name 367
draw:fill-hatch-name 367
draw:fill-image-height 368
draw:fill-image-name 368
draw:fill-image-width 368
draw:filter-name 87
draw:fit-to-size 372
draw:gamma 374
draw:gradient-step-count 367
draw:green 374

draw:guide-distance 377
draw:guide-overhang 377
draw:id 347
draw:kind 337
draw:layer 347
draw:line-distance 376
draw:line-skew 340
draw:luminance 373
draw:marker-end 365
draw:marker-end-center 366
draw:marker-end-width 365
draw:marker-start 365
draw:marker-start-center 365
draw:marker-start-width 365
draw:mayscript 91
draw:measure-align 378
draw:measure-vertical-align 378
draw:mimetype 91
draw:name 85
draw:object 90
draw:parallel 378
draw:placing 377
draw:red 374
draw:refX 369
draw:refY 369
draw:rotation 359
draw:shadow 375
draw:shadow-color 375
draw:shadow-distance-x 375
draw:shadow-distance-y 375
draw:shadow-transparency 376
draw:start-color 357
draw:start-glue-point 339
draw:start-guide 377
draw:start-intensity 357
draw:start-line-spacing-horizontal 376
draw:start-line-spacing-vertical 376
draw:start-shape 339
draw:stroke 364
draw:style-name 86, 87, 346
draw:textarea-horizontal-align 372
draw:textarea-vertical-align 372
draw:tile-repeat-offset 369
draw:transform 346
draw:transparency 369
draw:transparency-name 369
draw:value 92
draw:z-index 347
draw:z-index 96
drawing page 328
drawing shapes 331
drawing shapes, common attributes 344
drawing shapes, group 344
drop caps 222

E

echo character 410
editable 100
editing cycles 41
editing duration 42
ellipse 337
encryption 542
end-of-line 33
end-of-line handling 33
endnotes 149
era element 70
event name 112
event tables 111
event:language 508
event:library 508
event:macro-name 508
event:name 507
events 111
events for controls 432
expression fields 191

F

field attributes, common 194
field value attributes 194
field value type 194
fields 152
fields, author 160
fields, common characteristics 152
fields, date 153
fields, document 153
fields, expression 191
fields, page numbers 155
fields, sender 157
fields, sequence 190
fields, time 154
fields, variable 183
file name fields 175
fill color 367
fill properties 366
fill style 367
filter condition 294
filter name 87
filter-and 293
filter-or 293
filters 292
fixed and minimum frame heights 97
fixed and minimum frame widths 97
fixed attribute 195
fixed fields 152
fixed index string 442
fixed line height 216
fixed text 75
floor 477
fo:background-color 225

fo:border 227
fo:break-after 225
fo:break-before 225
fo:clip 103
fo:color 204
fo:column-gap 230
fo:columns-count 230
fo:country 210
fo:country-asian 210
fo:country-complex 210
fo:direction 321
fo:end-indent 231
fo:font-family 206
fo:font-size 208
fo:font-size-asian 208
fo:font-size-complex 208
fo:font-style 210
fo:font-style-asian 210
fo:font-style-complex 210
fo:font-variant 203
fo:font-weight 212
fo:font-weight-asian 212
fo:font-weight-complex 212
fo:glyph-orientation-vertical 321
fo:height 140
fo:hyphenate 220
fo:hyphenation-keep 220
fo:hyphenation-ladder-count 221
fo:hyphenation-push-char-count 221
fo:hyphenation-remain-char-count 221
fo:language 209
fo:language-asian 209
fo:language-complex 209
fo:letter-spacing 209
fo:line-height 216
fo:margin-bottom 224
fo:margin-left 223
fo:margin-right 223
fo:margin-top 224
fo:max-width
fo:max-height 98
fo:min-height 97
fo:min-width 97
fo:orphans 218
fo:overflow 322
fo:padding 228
fo:page-height 53
fo:page-width 53
fo:score-spaces 212
fo:space-after 231
fo:space-before 231
fo:start-indent 231
fo:text-align 136, 217
fo:text-indent 224
fo:text-shadow 211
fo:text-transform 203
fo:vertical-align 320
fo:widows 218
fo:width 140
fo:wrap-option 322
font character set 207
font declaration 63
font family 205
font pitch 207
font size 208
font style 206, 210
font variant 203
font weight 211
footnote citation text 146
footnote continuation 148
footnote layout 61
footnote maximum height 61
footnote paragraph style 147
footnote reference ID 150
footnote spacing 61
footnotes 146, 149
footnotes position 148
form:allow-deletes 404
form:allow-inserts 404
form:allow-updates 404
form:apply-filter 404
form:auto-complete 413
form:bound-column 430
form:button-type 421
form:command 405
form:command-type 404
form:convert-empty-to-null 431
form:current-selected 422
form:current-state 417
form:current-value 422
form:data-field 431
form:datasource 405
form:default-button 415
form:default-value 423
form:detail-fields 405
form:disabled 424
form:dropdown 424
form:echo-char 410
form:enctype 403
form:escape-processing 405
form:filter 406
form:for 425
form:id 421
form:ignore-result 406
form:image-data 425
form:is-tristate 417
form:label 425
form:list-source 431
form:list-source-type 432
form:master-fields 405
form:max-length 426
form:max-value 411

form:method 403
form:min-value 411
form:multi-line 412
form:multiple 414
form:name 421
form:navigation-mode 406
form:order 406
form:printable 426
form:property-is-list 435
form:property-name 435
form:property-type 436
form:readonly 427
form:selected 427
form:service-name 421
form:size 427
form:state 417
form:tab-cycle 407
form:tab-index 428
form:tab-stop 428
form:title 430
form:validation 412
form:value 423
format change 236
format source 79
formatting properties 44
formatting properties, complex 45
formatting properties, simple 45
formatting property sets 45
forms 401
forms element 33
formula 197
formula attribute 270
forward-compatible processing 34
fraction 66
frame background 100
frame formatting properties 97
frames 84
frames in text documents 199

G

generator 36
generic font family 206
gradient 356
gradient transparency 369
graphic properties 373
graphic style elements 356
grid 481
group of drawing shapes 344
grouping separator 82

H

hatch 358
heading level 119
headings and paragraphs 117
hidden paragraph fields 173
hidden text fields 172
horizontal position 99
horizontal relation 99
hours 73
hyperlink behavior 40
hyperlink index entry 444
hyperlinks, in text documents 122
hyperlinks, simple 105
hyphenation 220
hyphenation keep 220
hyphenation push char count 221
hyphenation remain char count 220

I

image 359
image level style 138
image size 139
image vertical alignment 140
image:bitmap-index 511
image:command 511, 512
image:maskbitmap 510
image:maskcolor 510
image:maskmode 510
images 86
index body 438
index entries 125, 441
index entries, combining 465
index entry template 438
index entry text 442
index entry, alphabetical 449
index entry, main 450
index entry, user-defined 448
index marks 447
index of illustrations 455
index of objects 458
index of tables 457
index source 437
index tab stop 443
index title 438
index, alphabetical 463
index, bibliography 467
index, user-defined 460
initial creator 37
insertion 235, 246
ISO 3166 24
ISO 639 24
ISO 8601 24

J

justify single word 217

K

keep with next 228
keywords 37
keywords 37

L

label alignment 136
language 41, 77, 209
layer ID 95, 96
leader character 220
left and right margins for frames 98
left and right margins for paragraphs 223
legend 474
letter kerning 212
letter synchronization 110
line 332
line breaks 121
line distance 216
line end center 366
line numbering 142
link location 105
link name 106
link target frame 105
list header 132
list style 48
list style name 131
list styles 134
lists, bulleted and numbered 130
lists, ordered 131
lists, unordered 131

M

macro fields 178
major version 34
manifest file 543
manifest:algorithm-name 545
manifest:full-path 543
manifest:initialisation-vector 545
manifest:key-derivation 546
manifest:key-derivation-name 545
manifest:media-type 544
manifest:size 544
map 49
map applied style 50, 51, 281
map condition 49, 280, 282, 283, 284
marker element 361
master pages 326
master styles 31
matrix 270
maximum hyphens 221
menu:helpid 491
menu:id 492
menu:label 492
meta information 35
meta information, example of 44
meta:delay 40
meta:target-frame-name 41
meta> element 31
metadata 31
metadata fields 166
metadata, user-defined 42
minimum denominator digits 84
minimum exponent digits 83
minimum label distance 136
minimum line height 216
minimum number of integer digits 82
minimum numerator digits 83
minimum width of a number 136
minor version 34
minutes 73
mirroring 102
modification date 38
month 69
moving content in a cell 250

N

name 94, 329
named expressions 289
named range 290
namespaces 27
native number system 80
next style 47
non-breaking blanks 126
non-breaking hyphens 126
number 65
number format 109
number format specification 110
number level style 135
number of cells repeated attribute 268
number of columns repeated attribute 262
number of columns spanned attribute 269
number of columns spanned by matrix attribute 270
number of rows repeated attribute 265
number of rows spanned attribute 269
number of rows spanned by matrix attribute 270
number style 64
number:automatic-order 79
number:calendar 84
number:country 67, 78
number:decimal-places 74, 82
number:decimal-replacement 83
number:display-factor 83
number:format-source 79

number:grouping 82
number:language 67, 77
number:min-denominator-digits 84
number:min-exponent-digits 83
number:min-integer-digits 82
number:min-numerator-digits 83
number:style 69
number:textual 69
number:title 78
number:transliteration-country 81
number:transliteration-format 80
number:transliteration-language 81
number:transliteration-style 81
number:truncate-on-overflow 80

O

office:author 275
office:automatic-update 130
office:chg-author 113
office:chg-date 113
office:create-date 275
office:dde-application 130
office:dde-item 130
office:dde-topic 130
office:display 276
office:layer-id 95, 96
office:name 95, 106
office:server-map 106
office:target-frame 105, 403, 429
office:target-frame-name 123
OpenOffice.org application settings 114
orphans 218
outline level style 141
outline numbering 140
outline style 141

P

package format 541
padding 100, 228
page and column breaks 224
page continuation text 156
page duration 388
page master 51, 59, 327
page name 329
page name. 58, 326
page number fields 155
page number format 52
page size 53
page style 327, 329
page styles and layout 51
page usage 53
page variable fields 176
page visibility 389
paragraph background color 225
paragraph background image 225
paragraph border 226
paragraph border line width 227
paragraph formatting properties 120
paragraph text 118
paragraph-only wrapping 101
parent style 47
percentage style 68
placeholders 161
plot area 474
point references 125
polygon 334
polyline 333
prefix and suffix 109
presentation notes 328
presentation page attributes 387
presentation shapes 347
presentation shapes, common attributes 347
presentation:animations 393
presentation:class
  presentation:placeholder
  presentation:user-transformed 348
presentation:endless 392
presentation:force-manual 392
presentation:full-screen 391
presentation:mouse-as-pen 393
presentation:mouse-visible 393
presentation:page-duration 389
presentation:pause 392
presentation:presentation-page-layout-name 328, 330
presentation:show 391
presentation:show-logo 392
presentation:sound 389
presentation:start-page 391
presentation:start-with-navigator 393
presentation:stay-on-top 394
presentation:style-name 348
presentation:transition-on-click 394
presentation:transition-speed 388
presentation:transition-style 388
presentation:transition-typ 387
presentation:visibility 389
print content 98
print date 38
printed by 37
protect 98

Q

quarter element 71

R

radius 336
range references 125
range usable as 291
rectangle 331
references 125
register true 223
related documentation 24
reload delay 40
reload URL 39
restart numbering 133
row element 265
row group element 264
row style attribute 265
ruby 201
run through wrapping mode 102

S

scenario table 258
scientific number 66
script: 538, 539
script:event 537
script:event-name 112, 432, 538
script:language 112, 538, 539
script:listener-event 539
script:listener-method 539
script:listener-param 539
script:listener-type 539
script:location 537, 539
script:macro-name 538, 539
scripts 110
seconds 73
section background 230
section columns 230
section formatting properties 229
sections 126
sender fields 157
sequence fields 190
sequence variables, declaring 189
series 481
server side image map 106
shadow 100, 228
shadow offset 375
shapes, 3D 349
shapes, drawings 331
shapes, presentations 347
simple variables, declaring 183
simple variables, displaying 185
simple variables, setting 184
soft hyphens 126
sort 300
sort by 301
sort groups 303
sound, in presentations 389
spacing and alignment 141
span 121
SQL database 298
statusbar:align 497
statusbar:autosize 497
statusbar:offset 498
statusbar:ownerdraw 498
statusbar:style 497
statusbar:width 498
stroke properties 363
style 95
style and conditional style 119
style family 47
style mapping, example 51
style mappings 49
style name 46
style:apply-style-name 50
style:auto-text-indent 224
style:auto-update 48
style:automatic 47
style:base-cell-address 51
style:border-line-width 227
style:break-inside 218
style:cell-protect 323
style:chain-next-name 86
style:char 219
style:column-width 319
style:condition 50
style:data-style-name 163, 185, 197
style:distance 222
style:distance-after-sep 61
style:distance-before-sep 61
style:editable 101
style:family 47
style:filter-name 226
style:first-page-number 56
style:font-charset 208
style:font-charset-asian 208
style:font-charset-complex 208
style:font-family-asian 206
style:font-family-complex 206
style:font-family-generic 206
style:font-family-generic-asian 206
style:font-family-generic-asian style:font-family-generic-complex 206
style:font-name 205
style:font-name-asian 205
style:font-name-complex 205
style:font-pitch 207
style:font-pitch-complex 207
style:font-pitchgv 207
style:font-size-rel 209
style:font-size-rel-asian 209
style:font-size-rel-complex 209
style:font-style-name 207
style:font-style-name-asian 207

style:font-style-name-complex 207
style:horizontal-align 62
style:horizontal-pos 99, 200
style:horizontal-rel 99, 200
style:justify-single-word 218
style:keep-with-next 228
style:leader-char 220, 444
style:length 62, 222
style:letter-kerning 212
style:line-break 216
style:line-height-at-least 216
style:line-spacing 217
style:lines 222
style:list-style-name 48
style:master-page-name 48
style:max-height 61
style:may-break-between-rows 318
style:min-row-height 320
style:mirror 103
style:name 46, 53, 77, 134
style:next-style-name 47, 59
style:num-format 110
style:num-letter-sync 110
style:num-prefix 109
style:num-suffix 109
style:number-wrapped-paragraphs 101
style:page-master-name 59
style:page-usage 53
style:paper-tray-name 54
style:parent-style-name 47
style:position 226, 444
style:print 56
style:print-content 98, 323
style:print-orientation 54
style:print-page-order 56
style:protect 98, 233
style:punctuation-wrap 215
style:register-true 223
style:register-truth-ref-style-name 56
style:rel-column-width 319
style:rel-width 231, 316
style:repeat 226, 368
style:rotation-align 323
style:rotation-angle 322, 481
style:row-height 320
style:ruby-align 202
style:ruby-position 202
style:run-through 102
style:scale-to 57
style:scale-to-pages 57
style:shadow 228
style:style 95
style:style-name 223
style:table-centering 57
style:text-align-last 217
style:text-align-source 321
style:text-autospace 214
style:text-background-color 213
style:text-blinking 213
style:text-combine 213
style:text-combine-end-char 214
style:text-combine-start-char 214
style:text-crossing-out 204
style:text-emphasize 214
style:text-outline 204
style:text-position 205
style:text-underline 211
style:text-underline-color 211
style:type 444
style:vertical-pos 100, 140, 200
style:vertical-rel 100, 140, 201
style:volatile 78
style:width 62, 316
style:wrap 101
style:wrap-contour 102
style:wrap-contour-mode 102
styles 31, 46
styles, examples of 32
styles, location of 32
styles:name 59
subject 36
subtable elements 284
subtotal field 304
subtotal rule 304
subtotal rules 302
svg:cx 336
svg:cy 336
svg:d 335
svg:end-angle 337
svg:height 97, 345
svg:r 336
svg:start-angle 337
svg:stroke-color 364
svg:stroke-dash 364
svg:stroke-linejoin 366
svg:stroke-opacity 366
svg:stroke-width 364
svg:viewbox 346
svg:width 97, 345
svg:x1 333
svg:x2 333
svg:y1 333
svg:y2 333

T

tab position 219
tab stops 120, 218
tab type 219
tabbing navigation order 428
table alignment 316

table cell content validations 280
table cell element 267
table cell formatting properties 320
table element 255
table filter 292
table filter element 292
table formatting properties 313
table margins 317
table name attribute 255
table of contents 451
table row formatting properties 319
table style attribute 255
table width 316
table:acceptance-state 254
table:algorithm 301
table:align 316
table:allow-empty-cell 282
table:application-data 306
table:automatic-find-labels 242
table:base-cell-address 281
table:base-change-position 248
table:bind-styles-to-conten 303
table:bind-styles-to-content 300
table:boolean-value 272
table:border-color 259
table:border-model 318
table:buttons 307
table:case-sensitive 241, 294, 301, 303
table:cell-address 252, 253
table:cell-range-address 277, 309
table:column
  table:row
  table:table 251
table:column, table:row, table:table 253
table:comment 260
table:condition 280
table:condition-source 293
table:condition-source-range-address 293
table:contains-error 277
table:contains-header 297
table:content-validation-name 269
table:conversion-mode 315
table:copy-back 259
table:copy-formulas 259
table:copy-styles 259
table:count 246
table:country 301
table:currency 273
table:data-cell-range-address 289
table:data-type 294, 303
table:database-name 298
table:date-value 242, 272
table:default-cell-style-name 263, 266
table:direction 277
table:display 261
table:display-border 259
table:display-duplicates 293
table:display-filter-buttons 298
table:end-cell-address 345
table:end-x 345
table:end-y 345
table:execute 284
table:expression 292
table:field-number 294, 304
table:filter-name 257
table:filter-options 258
table:formula 270
table:function 305, 310
table:grand-total 307
table:group-by-field-number 304
table:has-persistent-data 297
table:id 254
table:identify-categories 307
table:ignore-empty-rows 307
table:index 278
table:is-active 260
table:is-data-layout-field 310
table:is-selection 296
table:label-cell-range-address 289
table:language 301
table:marked-invalid 277
table:matrix-covered 252
table:maximum-difference 243
table:message-type 283
table:mode 257
table:name 255, 278
table:null-year 242
table:number-columns-repeated 262, 268
table:number-columns-spanned 269
table:number-matrix-columns-spanned 271
table:number-matrix-rows-spanned 271
table:number-rows-repeated 265
table:number-rows-spanned 269
table:on-update-keep-size 297
table:on-update-keep-styles 297
table:operator 295
table:order 302, 304
table:orientation 289, 297, 311
table:page-breaks-on-group-change 303
table:page-style-name 318
table:parse-sql-statement 299
table:position 248, 249
Table:position 246
table:precision-as-shown 241
table:previous 253
table:print-ranges 256
table:protected 256
table:protection-key 244, 256
table:query-name 300
table:range-usable-as 291
table:refresh-delay 258, 298
table:rejecting-change-id 255

table:scenario-ranges 260
table:search-criteria-must-apply-to-whole-cell 241
table:show-empty 311
table:source-field-name 310
table:sql-statement 299
table:start-column
table:end-column
table:start-row
table:end-row
table:start-table
table:end-table 251
table:start-position; table:end-position; table:position 249
table:status 243
table:steps 243
table:string-value 272
table:structure-protected
table:protection-key 240
table:style-name 256, 263, 266, 269
table:table 247, 248
table:table-background 345
table:table-name 258
table:target-range-address 292, 298, 307
table:target-range-address
table:source-range-address 250
table:time-value 272
table:title 282
table:track-changes 244
table:type 246, 248
table:used-hierarchy 311
table:value 271, 295
table:value-type 242, 271
table:visibility 263, 266
target frame 40
template 38
template location 39
template modification date 39
template title 39
terminology 25
text align 217
text align of last line 217
text background color 213
text blinking 212
text box 85
text content 76
text decoration word mode 212
text formatting properties 121, 203
text headings 117
text indent 223
text input fields 192
text outline 204
text paragraphs 117
text position 204
text section 126
text shadow 210
text style 75
text styles 121
text transformations 203
text:active 177
text:alphabetical-separators 464
text:anchor-page-number 201
text:anchor-type 199
text:automatic-update 180
text:bibliography-data-field 443
text:bibliography-type 468
text:bullet-char 138
text:bullet-relative-size 138
text:capitalize-entries 466
text:caption-sequence-name 456
text:citation-body-style-name 147
text:citation-style 146
text:column-name 163
text:combine-entries 465
text:comma-separated 466
text:cond-style-name 119
text:condition 128, 164, 171, 172, 173
text:consecutive-numbering 135
text:continue-numbering 132
text:copy-outline-levels 462
text:count-empty-lines 144
text:count-in-floating-frames 144
text:database-column-name 163
text:database-name 163, 164, 165, 166
text:date-adjust 154
text:date-value 154
text:dde-application 179
text:dde-item 179
text:dde-topic 179
text:default-style 147
text:description 196
text:display 127, 174, 175, 185, 196
text:display-levels 137
text:display-outline-level 189
text:filter-name 129
text:fixed 195
text:footnotes-position 148
text:formula 184, 197
text:id 150
text:ignore-case 464
text:increment 143, 145
text:index-name 449
text:index-scope 440
text:key1 450
text:key2 450
text:label 150
text:level 119, 135
text:line-number 145
text:main-entry 450
text:main-entry-style-name 464
text:master-page-name 147
text:min-label-distance 136
text:min-label-width 136
text:name 123, 127, 178, 180, 184, 196

text:num-format 166, 198
text:num-letter-sync 166, 198
text:number-lines 143, 145
text:offset 144
text:outline-level 175, 439
text:page-adjust 156, 177
text:placeholder-type 161
text:position 144
text:protected 440
text:protection-key 234
text:ref-name 181, 191
text:reference-format 181
text:relative-tab-stop-position 440
text:restart-numbering 133
text:restart-on-page 144
text:row-number 165
text:section-name 129
text:select-page 156, 157
text:separation-character 190
text:sequence-format 456
text:space-before 136
text:start-numbering-at 148
text:start-value 133, 137, 147
text:string-value 157, 172
text:string-value-if-false 171
text:string-value-if-true 171
text:style-name 119, 124, 127, 132, 137, 439, 440
text:style-name 143
text:table-name 163, 164, 165, 166
text:table-type 163, 164, 165, 166
text:time-adjust 155
text:time-value 155
text:track-changes 234
text:use-caption 456
text:use-chart-objects 459
text:use-draw-objects 459
text:use-floating-frames 461
text:use-graphics 461
text:use-index-marks 453, 461
text:use-index-source-styles 453
text:use-keys-as-entries 465
text:use-math-objects 459
text:use-objects 461
text:use-other-objects 459
text:use-spreadsheet-objects 459
text:use-tables 461
text:value 166
text:value-type 184, 194
text:visited-style-name 124
textual representation attribute 69
thumbnail 343
tickmark properties 480
tile reference point 369
tile translation 369
time adjustment 155
time fields 154
time style 72
time value 155
time value truncation 80
title 36, 78
toolbar:align 505
toolbar:bitmap 502
toolbar:dockinglines 505
toolbar:floating 504
toolbar:floatinglines 505
toolbar:floatingposleft 504
toolbar:floatingpostop 504
toolbar:helpid 502
toolbar:id 504
toolbar:style 501, 506
toolbar:text 501
toolbar:userdefined 501
toolbar:visible 500, 505
toolbar:width 501
top and bottom margins for frames 98
top and bottom margins for paragraphs 224
transition speed 388
transition style 388
transition type 387
transparency gradient 360
tristate 524

U

underlining 211
unnamed styles 44
user variable input fields 188
user variables, declaring 187
user variables, displaying 187
user-defined metadata 42

V

validation 34
value-step 529, 530, 531, 532
variable fields 183
variable input fields 186
version attribute 30
version attribute, function of 34
versions 34
vertical position 100
vertical relation 100
volatility 78

W

wall 476
week of year element 71
white-space characters 33, 118, 120
widows 218

wrap through 102
wrapping 101

X

xlink:actuate 257
xlink:href 123, 129, 139, 257, 403, 429, 495, 496, 500, 509, 512
xlink:show 129
xlink:type 129, 257

Y

year 70

Z

Z index 96

Change Log

Changes from specification document version 1 to 2

- Some erroneous DTD examples have been corrected.
- Some deprecated and unused element descriptions have been removed (for instance `<style:use-styles>` and `<office:a-map>`.
- Some minor errors in specification text have been corrected.
- Chapter about dialogs added.