# Table of Contents

1 **Altova Authentic 2019 Desktop Enterprise Edition** ................................................. 3

2 **User Guide and Reference** .................................................................................. 6

2.1 Interface and Environment .................................................................................... 7

2.1.1 The Graphical User Interface (GUI) ................................................................. 8

   - Main Window ........................................................................................................ 10
   - Project Window .................................................................................................... 12
   - Info Window ........................................................................................................ 14
   - Entry Helpers ....................................................................................................... 15
   - Output Window: Messages .......................................................... 16
   - Menu Bar, Toolbars, Status Bar ................................................................. 17

2.1.2 The Application Environment ........................................................................... 18

   - Settings and Customization ........................................................................ 19
   - Tutorials, Projects, Examples ................................................................. 20
   - Authentic Desktop Features and Help, and Altova Products ....................... 21

2.2 **Authentic View Tutorial** .................................................................................. 22

2.2.1 Opening an XML Document in Authentic View ........................................ 24

2.2.2 The Authentic View Interface ........................................................................ 26

2.2.3 Node Operations ............................................................................................. 29

2.2.4 Entering Data in Authentic View ................................................................. 32

2.2.5 Entering Attribute Values ........................................................................... 34

2.2.6 Adding Entities ............................................................................................... 35

2.2.7 Printing the Document ................................................................................... 36

2.3 **Authentic View Interface** .................................................................................. 37

2.3.1 Overview of the GUI ..................................................................................... 38

2.3.2 Authentic View Toolbar Icons ..................................................................... 40

2.3.3 Authentic View Main Window ....................................................................... 43

2.3.4 Authentic View Entry Helpers ..................................................................... 46

2.3.5 Authentic View Context Menus .................................................................... 51

2.4 **Editing in Authentic View** ............................................................................... 54

2.4.1 Basic Editing ................................................................................................... 55

2.4.2 Tables in Authentic View .............................................................................. 60

   - SPS Tables ........................................................................................................ 61

   - CALS/HTML Tables ......................................................................................... 63
2.4.3 Editing a DB ................................. 71
- Navigating a DB Table ......................... 72
- DB Queries ........................................ 73
- Modifying a DB Table ........................... 78
2.4.4 Working with Dates ......................... 80
- Date Picker ....................................... 81
- Text Entry ........................................ 82
2.4.5 Defining Entities ............................ 83
2.4.6 XML Signatures ............................. 85
2.4.7 Images in Authentic View ................ 87
2.4.8 Keystrokes in Authentic View .......... 88

2.5 Authentic Scripting ............................ 89

2.6 Browser View ................................... 91

2.7 Altova Global Resources ..................... 92

2.7.1 Defining Global Resources ............... 93
- Files ............................................... 96
- Folders .......................................... 101
- Databases ....................................... 103

2.7.2 Using Global Resources .................. 105
- Assigning Files and Folders ................. 106
- Changing the Active Configuration ....... 109

2.8 Source Control ................................. 110

2.8.1 Setting Up Source Control ............... 112

2.8.2 Supported Source Control Systems ....... 113

2.8.3 Local Workspace Folder ................. 115

2.8.4 Application Project ......................... 116

2.8.5 Add to Source Control ...................... 118

2.8.6 Working with Source Control ............ 120
- Add to, Remove from Source Control ...... 121
- Check Out, Check In ............................ 122
- Getting Files as Read-Only .................. 124
- Copying and Sharing from Source Control 126
- Changing Source Control .................... 129

2.8.7 Source Control with Git ................... 130
- Enabling Git Source Control with GIT SCC Plug-in 131
- Adding a Project to Git Source Control .... 132
- Cloning a Project from Git Source Control 134

2.9 Authentic Desktop in Visual Studio ...... 136

2.9.1 Installing the Authentic Desktop Plugin for Visual Studio 137

2.9.2 Differences with Standalone Version ... 138

2.10 Authentic Desktop in Eclipse .......... 139
2.10.1 Installing the Authentic Desktop Plugin for Eclipse ........................................ 140
2.10.2 Authentic Desktop Entry Points in Eclipse ................................................... 147
2.11...Menu Commands .......................................................................................... 150

2.11.1 File Menu ........................................................................................................ 151
  - New ....................................................................................................................... 152
  - Open ...................................................................................................................... 154
  - Reload .................................................................................................................. 160
  - Encoding ............................................................................................................... 161
  - Close, Close All, Close All But Active ................................................................. 162
  - Save, Save As, Save All........................................................................................ 163
  - Send by Mail ......................................................................................................... 169
  - Print ....................................................................................................................... 171
  - Print Preview, Print Setup .................................................................................... 172
  - Recent Files, Exit .................................................................................................. 173

2.11.2 Edit Menu ........................................................................................................ 174
  - Undo, Redo ............................................................................................................ 175
  - Cut, Copy, Paste, Delete ....................................................................................... 176
  - Select All ............................................................................................................... 177
  - Find, Find Next ..................................................................................................... 178
  - Replace .................................................................................................................. 179

2.11.3 Project Menu ................................................................................................... 180
  - New Project .......................................................................................................... 183
  - Open Project ......................................................................................................... 184
  - Reload Project ...................................................................................................... 185
  - Close Project ........................................................................................................ 186
  - Save Project, Save Project As .............................................................................. 187
  - Source Control ..................................................................................................... 188
    Open from Source Control .................................................................................... 188
    Enable Source Control ......................................................................................... 189
    Get Latest Version ................................................................................................. 190
    Get, Get Folders .................................................................................................... 190
    Check Out, Check In .............................................................................................. 191
    Undo Check Out .................................................................................................... 193
    Add to Source Control ......................................................................................... 194
    Remove from Source Control .............................................................................. 195
    Share from Source Control .................................................................................. 195
    Show History ......................................................................................................... 197
    Show Differences .................................................................................................. 198
    Show Properties .................................................................................................... 199
    Refresh Status ....................................................................................................... 200
    Source Control Manager ...................................................................................... 200
    Change Source Control ....................................................................................... 200
    – Add Files to Project ........................................................................................... 202
    – Add Global Resource to Project ....................................................................... 203
- Add URL to Project ................................................................. 204
- Add Active File to Project ..................................................... 205
- Add Active And Related Files to Project ............................ 206
- Add Project Folder to Project ............................................... 207
- Add External Folder to Project ............................................ 208
- Add External Web Folder to Project .................................... 211
- Script Settings ...................................................................... 215
- Properties ........................................................................... 216
- Most Recently Used Projects .............................................. 219

2.11.4 XML Menu .................................................................... 220
- Check Well-Formedness ....................................................... 221
- Validate XML ...................................................................... 223

2.11.5 XSL/XQuery Menu ............................................................ 225
- XSL Transformation ............................................................ 226
- XSL-FO Transformation ..................................................... 227
- XSL Parameters / XQuery Variables ................................. 229

2.11.6 Authentic Menu ............................................................. 233
- New Document ................................................................. 234
- Edit Database Data ............................................................ 235
- Edit StyleVision Stylesheet ............................................... 236
- Select New Row with XML Data for Editing ..................... 237
- XML Signature ..................................................................... 238
- Define XML Entities .......................................................... 240
- View Markup ...................................................................... 242
- RichEdit ............................................................................. 243
- Append/Insert/Duplicate/Delete Row ............................... 244
- Move Row Up/Down .......................................................... 245
- Generate HTML, RTF, PDF, Word 2007+ Document .......... 246
- Trusted Locations ............................................................. 247

2.11.7 View Menu ..................................................................... 248
- Authentic View ................................................................. 249
- Browser View .................................................................... 250

2.11.8 Browser Menu ............................................................... 251
- Back ................................................................................... 252
- Forward .............................................................................. 253
- Stop .................................................................................. 254
- Refresh .............................................................................. 255
- Fonts ................................................................................ 256
- Separate Window .............................................................. 257

2.11.9 Tools Menu .................................................................... 258
- Spelling .............................................................................. 259
- Spelling Options ............................................................... 262
- Scripting Editor ................................................................. 265
- Macros ............................................................................... 266
3 Programmers' Reference

2.11.10 Window Menu ......................................................... 299
  - Cascade ........................................................................ 300
  - Tile Horizontally ......................................................... 301
  - Tile Vertically ............................................................ 302
  - Project Window ........................................................... 303
  - Info Window ............................................................... 304
  - Entry Helpers ............................................................. 305
  - Output Windows .......................................................... 306
  - Project and Entry Helpers ........................................... 307
  - All On/Off .................................................................. 308
  - Currently Open Window List ....................................... 309

2.11.11 Help Menu ............................................................. 310
  - Table of Contents, Index, Search .................................. 311
  - Keyboard Map ............................................................ 312
  - Activation, Order Form, Registration, Updates ................. 313
  - Other Commands ......................................................... 317

2.11.12 Command Line ...................................................... 318
3.1 Scripting Editor ........................................................................................................................ 322

3.1.1 Overview ............................................................................................................................... 324
  – Scripting Projects in Authentic Desktop .................................................................................... 325
  – The Scripting Editor GUI .......................................................................................................... 327
  – Components of a Scripting Project ............................................................................................ 331

3.1.2 Creating a Scripting Project .................................................................................................... 333

3.1.3 Global Declarations .............................................................................................................. 335

3.1.4 Forms .................................................................................................................................... 337
  – Creating a New Form .................................................................................................................... 338
  – Form Design and FormObjects .................................................................................................. 340
  – FormEvents ................................................................................................................................. 343

3.1.5 Events .................................................................................................................................... 345

3.1.6 Macros .................................................................................................................................... 349
  – Creating and Editing a Macro ...................................................................................................... 350
  – Running a Macro ....................................................................................................................... 352
  – Debugging a Macro ...................................................................................................................... 355

3.1.7 Programming Points ............................................................................................................. 356
  – Built-in Commands .................................................................................................................... 358
    Form usage and commands ......................................................................................................... 365

3.1.8 Migrating to Scripting Editor 2010 and Later ........................................................................ 366

3.2 IDE Plugins ............................................................................................................................... 369

3.2.1 Registration of IDE PlugIns ..................................................................................................... 370

3.2.2 ActiveX Controls ..................................................................................................................... 371

3.2.3 Configuration XML .................................................................................................................. 372

3.2.4 ATL sample files ...................................................................................................................... 375
  – Interface description (IDL) .......................................................................................................... 376
  – Class definition ............................................................................................................................. 378
  – Implementation ............................................................................................................................ 379

3.2.5 IXMLSpyPlugIn ....................................................................................................................... 382
  – OnCommand ............................................................................................................................... 383
  – OnUpdateCommand .................................................................................................................... 384
  – OnEvent ..................................................................................................................................... 385
  – GetUIModifications ..................................................................................................................... 388
  – GetDescription .......................................................................................................................... 389

3.3 Application API .......................................................................................................................... 390

3.3.1 Overview ............................................................................................................................... 392
  – Object Model ............................................................................................................................... 393
  – Programming Languages ............................................................................................................ 394
    JSencript ....................................................................................................................................... 395
    
    Start Application .......................................................................................................................... 395
    Simple Document Access ........................................................................................................... 396
    Iteration ....................................................................................................................................... 397
    Error Handling ............................................................................................................................. 397
    Events .......................................................................................................................................... 398
3.3.2 Interfaces ............................................................................................................... 421
  Application .................................................................................................................. 422
  Events ......................................................................................................................... 423
    OnBeforeOpenDocument ......................................................................................... 423
    OnBeforeOpenProject ............................................................................................... 424
    OnDocumentOpened ................................................................................................. 424
    OnProjectOpened ...................................................................................................... 425
  ActiveDocument .......................................................................................................... 425
  AddMacroMenuItem ...................................................................................................... 425
  AddXSLT_XQParameter ............................................................................................... 426
  Application .................................................................................................................. 426
  ClearMacroMenu ......................................................................................................... 426
  CreateXMLSchemaFromDBStructure ............................................................................. 427
  CurrentProject ............................................................................................................ 427
  Dialogs ........................................................................................................................ 427
  Documents .................................................................................................................. 428
  Edition ........................................................................................................................... 428
  FindInFiles ................................................................................................................... 428
  GetDatabaseImportElementList .................................................................................... 428
  GetDatabaseSettings ................................................................................................... 429
  GetDatabaseTables ..................................................................................................... 429
  GetExportSettings ....................................................................................................... 430
  GetTextImportElementList .......................................................................................... 430
  GetTextImportExportSettings ..................................................................................... 431
  GetXSLT_XQParameterCount ...................................................................................... 432
  GetXSLT_XQParameterName ....................................................................................... 432
  GetXSLT_XQParameterXPath ...................................................................................... 432
  ImportFromDatabase ................................................................................................... 432
  ImportFromSchema ..................................................................................................... 433
  ImportFromText .......................................................................................................... 434
  ImportFromWord ......................................................................................................... 435
IsAPISupported
MajorVersion
MinorVersion
NewProject
OpenProject
Parent
Quit
ReloadSettings
RemoveXSLT_XQParameter
RunMacro
ScriptingEnvironment
ServicePackVersion
ShowApplication
ShowFindInFiles
ShowForm
Status
URLDelete
URLMakeDirectory
Visible
WarningNumber
WarningText

- AuthenticContextMenu
  CountItems
  DeleteItem
  GetItemText
  InsertItem
  SetItemText

- AuthenticDataTransfer
dropEffect
getdata
ownDrag
type

- AuthenticEventContext
  EvaluateXPath
  GetEventContextType
  GetNormalizedTextValue
  GetVariableValue
  GetXMLNode
  IsAvailable
  SetVariableValue

- AuthenticRange
  AppendRow
  Application
  CanPerformAction
  CanPerformActionWith
Clone ........................................................................................................ 452
CollapsToBegin ................................................................. 452
CollapsToEnd ........................................................................ 452
Copy ........................................................................................................ 453
Cut ........................................................................................................ 453
Delete .................................................................................................... 453
DeleteRow ......................................................................................... 454
DuplicateRow .................................................................................... 454
EvaluateXPath .................................................................................. 455
ExpandTo .......................................................................................... 455
FirstCursorPosition ....................................................................... 455
FirstXMLData ................................................................................ 456
FirstXMLDataOffset ........................................................................ 457
GetElementAttributeNames ......................................................... 458
GetElementAttributeValue ............................................................... 459
GetElementHierarchy ...................................................................... 459
GetEntityNames ............................................................................... 459
GetVariableValue ............................................................................ 460
Goto ..................................................................................................... 460
GotoNext .......................................................................................... 460
GotoNextCursorPosition ............................................................... 461
GotoPrevious ..................................................................................... 461
GotoPreviousCursorPosition .......................................................... 462
HasElementCursorPosition ............................................................ 462
InsertEntity ....................................................................................... 463
InsertRow .......................................................................................... 463
IsCopyEnabled ................................................................................. 464
IsCutEnabled ..................................................................................... 464
IsDeleteEnabled ............................................................................... 465
IsEmpty .............................................................................................. 465
IsEqual ................................................................................................. 465
IsFirstRow ......................................................................................... 465
IsInDynamicTable ............................................................................ 466
IsLastRow ........................................................................................ 466
IsPasteEnabled ............................................................................... 466
IsSelected .......................................................................................... 467
IsTextStateApplied .......................................................................... 467
LastCursorPosition ......................................................................... 467
LastXMLData .................................................................................. 468
LastXMLDataOffset ......................................................................... 469
MoveBegin ....................................................................................... 470
MoveEnd ............................................................................................ 470
MoveRowDown .................................................................................. 471
MoveRowUp ........................................................................................ 471
Parent .................................................................................................. 471
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paste</td>
<td>471</td>
</tr>
<tr>
<td>PerformAction</td>
<td>472</td>
</tr>
<tr>
<td>Select</td>
<td>473</td>
</tr>
<tr>
<td>SelectNext</td>
<td>473</td>
</tr>
<tr>
<td>SelectPrevious</td>
<td>474</td>
</tr>
<tr>
<td>SetElementAttributeValue</td>
<td>475</td>
</tr>
<tr>
<td>SetFromRange</td>
<td>476</td>
</tr>
<tr>
<td>SetVariableValue</td>
<td>476</td>
</tr>
<tr>
<td>Text</td>
<td>476</td>
</tr>
<tr>
<td><strong>Authenticate</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td></td>
</tr>
<tr>
<td>OnBeforeCopy</td>
<td>478</td>
</tr>
<tr>
<td>OnBeforeCut</td>
<td>479</td>
</tr>
<tr>
<td>OnBeforeDelete</td>
<td>479</td>
</tr>
<tr>
<td>OnBeforeDrop</td>
<td>480</td>
</tr>
<tr>
<td>OnBeforePaste</td>
<td>480</td>
</tr>
<tr>
<td>OnBeforeSave</td>
<td>481</td>
</tr>
<tr>
<td>OnDragOver</td>
<td>481</td>
</tr>
<tr>
<td>OnKeyboardEvent</td>
<td>482</td>
</tr>
<tr>
<td>OnLoad</td>
<td>482</td>
</tr>
<tr>
<td>OnMouseEvent</td>
<td>483</td>
</tr>
<tr>
<td>OnSelectionChanged</td>
<td>483</td>
</tr>
<tr>
<td>OnToolbarButtonClicked</td>
<td>484</td>
</tr>
<tr>
<td>OnToolbarButtonExecuted</td>
<td>485</td>
</tr>
<tr>
<td>OnUserAddedXMLNode</td>
<td>485</td>
</tr>
<tr>
<td>Application</td>
<td>486</td>
</tr>
<tr>
<td>AsXMLString</td>
<td>486</td>
</tr>
<tr>
<td>ContextMenu</td>
<td>486</td>
</tr>
<tr>
<td>CreateXMLNode</td>
<td>487</td>
</tr>
<tr>
<td>DisableAttributeEntryHelper</td>
<td>487</td>
</tr>
<tr>
<td>DisableElementEntryHelper</td>
<td>487</td>
</tr>
<tr>
<td>DisableEntityEntryHelper</td>
<td>487</td>
</tr>
<tr>
<td>DocumentBegin</td>
<td>488</td>
</tr>
<tr>
<td>DocumentEnd</td>
<td>488</td>
</tr>
<tr>
<td>DoNotPerformStandardAction</td>
<td>488</td>
</tr>
<tr>
<td>EvaluateXPath</td>
<td>488</td>
</tr>
<tr>
<td>Event</td>
<td>488</td>
</tr>
<tr>
<td>EventContext</td>
<td>489</td>
</tr>
<tr>
<td>GetToolbarButtonState</td>
<td>489</td>
</tr>
<tr>
<td>Goto</td>
<td>490</td>
</tr>
<tr>
<td>IsRedoEnabled</td>
<td>490</td>
</tr>
<tr>
<td>IsUndoEnabled</td>
<td>491</td>
</tr>
<tr>
<td>MarkupVisibility</td>
<td>491</td>
</tr>
<tr>
<td>Parent</td>
<td>491</td>
</tr>
<tr>
<td>Print</td>
<td>491</td>
</tr>
<tr>
<td>Redo</td>
<td>492</td>
</tr>
<tr>
<td>Selection</td>
<td>492</td>
</tr>
</tbody>
</table>
SetToolbarButtonState .......................................................... 493
Undo .............................................................................. 493
UpdateXMLInstanceEntities ................................................. 493
WholeDocument ............................................................... 494
XMLDataRoot ..................................................................... 494

- CodeGeneratorDlg .......................................................... 495
Application ........................................................................ 495
CPPSettings_DOMType ....................................................... 496
CPPSettings_GenerateVC6ProjectFile .................................. 496
CPPSettings_GenerateGCCMakefile ..................................... 496
CPPSettings_GenerateVSProjectFile ..................................... 497
CPPSettings_LibraryType .................................................... 497
CPPSettings_UseMFC ........................................................ 497
CSharpSettings_ProjectType ............................................... 497
OutputPath ........................................................................ 498
OutputPathDialogAction ................................................. 498
OutputResultDialogAction .............................................. 498
Parent ............................................................................. 499
ProgrammingLanguage .................................................... 499
PropertySheetDialogAction .............................................. 499
TemplateFileName .......................................................... 500

- DatabaseConnection ....................................................... 501
ADOConnection ............................................................... 501
AsAttributes .................................................................... 502
CommentIncluded ........................................................... 502
CreateMissingTables ....................................................... 502
CreateNew ....................................................................... 503
DatabaseKind ................................................................... 503
DatabaseSchema ............................................................. 503
ExcludeKeys ..................................................................... 503
File ................................................................................. 504
ForeignKeys ..................................................................... 504
ImportColumnsType ........................................................ 504
IncludeEmptyElements .................................................... 504
NullReplacement ............................................................ 505
NumberDateTimeFormat .................................................... 505
ODBCConnection ............................................................. 505
PrimaryKeys ..................................................................... 505
SchemaExtensionType ..................................................... 506
SchemaFormat .................................................................. 506
SQLSelect ......................................................................... 506
TextFieldLen .................................................................... 506
UniqueKeys ....................................................................... 507

- Dialogs ........................................................................... 508
Application ........................................................................ 508
CodeGeneratorDlg........................................................................................................... 508
FileChooserDlg................................................................................................................ 509
Parent .................................................................................................................................. 509
SchemaDocumentationDlg.................................................................................................. 509
GenerateSampleXMLDlg...................................................................................................... 509
DTDSchemaGeneratorDlg.................................................................................................... 510
FindInFilesDlg.................................................................................................................... 510
WSDLDocumentationDlg.................................................................................................... 510
WSDL20DocumentationDlg.................................................................................................. 510
XBRLDocumentationDlg..................................................................................................... 511

- Document ......................................................................................................................... 512

Events .................................................................................................................................. 514
  OnBeforeSaveDocument..................................................................................................... 514
  OnBeforeCloseDocument.................................................................................................. 514
  OnBeforeValidate........................................................................................................... 515
  OnCloseDocument........................................................................................................... 516
  OnViewActivation.......................................................................................................... 516

Application ........................................................................................................................ 517
AssignDTD .......................................................................................................................... 517
AssignSchema.................................................................................................................... 517
AssignXSL .......................................................................................................................... 517
AssignXSLFO ...................................................................................................................... 518
AsXMLString ...................................................................................................................... 518
AuthenticView .................................................................................................................... 518
Close ................................................................................................................................... 519
ConvertDTDOrSchema ........................................................................................................ 519
ConvertDTDOrSchemaEx ...................................................................................................... 520
ConvertToWSDL20 ............................................................................................................. 520
CreateChild ....................................................................................................................... 521
CreateDBStructureFromXMLSchema .................................................................................. 521
CreateSchemaDiagram ....................................................................................................... 522
CurrentViewMode .............................................................................................................. 522
DataRoot ............................................................................................................................ 522
DocEditView ....................................................................................................................... 523
Encoding ............................................................................................................................. 523
EndChanges ....................................................................................................................... 524
ExecuteXQuery .................................................................................................................. 524
ExportToDatabase .............................................................................................................. 524
ExportToText ...................................................................................................................... 525
FlattenDTDOrSchema ........................................................................................................ 526
FullName ............................................................................................................................ 527
GenerateDTDOrSchema ........................................................................................................ 527
GenerateDTDOrSchemaEx ..................................................................................................... 528
GenerateProgramCode ........................................................................................................ 528
GenerateSampleXML ........................................................................................................... 528
GenerateSchemaDocumentation ......................................................................................... 529
GenerateWSDL20Documentation ................................................................. 529
GenerateWSDLDocumentation ................................................................. 529
GenerateXBRLDocumentation ................................................................. 530
GetDBStructureList ........................................................................... 530
GetExportElementList ....................................................................... 531
GetPathName (obsolete) ..................................................................... 531
GridView ......................................................................................... 531
IsModified ....................................................................................... 532
IsValid .............................................................................................. 532
IsValidEx ......................................................................................... 532
IsWellFormed ................................................................................... 533
Name ............................................................................................... 533
Parent ............................................................................................... 534
Path ................................................................................................... 534
RootElement ..................................................................................... 534
Save .................................................................................................. 535
SaveAs .............................................................................................. 535
Saved ............................................................................................... 535
SaveInString .................................................................................... 535
SaveToURL ....................................................................................... 536
SetActiveDocument .......................................................................... 536
SetEncoding (obsolete) ..................................................................... 536
SetExternals Valid ............................................................................ 538
SetPathName (obsolete) ..................................................................... 538
StartChanges .................................................................................. 538
Suggestions ...................................................................................... 539
SwitchViewMode .............................................................................. 539
TextView ......................................................................................... 539
Title .................................................................................................. 539
TransformXSL .................................................................................. 540
TransformXSLEx ............................................................................... 540
TransformXSLFO ............................................................................... 540
TreatXBRLInconsistenciesAsErrors ................................................... 541
UpdateViews .................................................................................... 541
UpdateXMLData ............................................................................... 541
Documents ....................................................................................... 542
Count ............................................................................................... 542
Item ................................................................................................. 543
NewAuthenticFile ............................................................................. 543
NewFile ............................................................................................ 543
NewFileFromText ............................................................................. 544
OpenAuthenticFile ............................................................................ 544
OpenFile ........................................................................................... 544
OpenURL .......................................................................................... 545
OpenURLDialog ............................................................................... 546
- DTDSchemaGeneratorDlg ........................................................................................................... 547
  Application .......................................................................................................................... 547
  AttributeTypeDefinition ........................................................................................................ 547
  DTDSchemaFormat ................................................................................................................ 548
  FrequentElements .................................................................................................................. 548
  GlobalAttributes ................................................................................................................... 548
  MaxEnumLength ...................................................................................................................... 548
  MergeAllEqualNamed ............................................................................................................. 549
  OnlyString Enums .................................................................................................................. 549
  OutputPath ................................................................................................................................ 549
  OutputPathDialogAction ........................................................................................................ 549
  Parent ....................................................................................................................................... 550
  ResolveEntities ..................................................................................................................... 550
  TypeDetection ....................................................................................................................... 550
  ValueList ................................................................................................................................ 550
- ElementList ................................................................................................................................ 551
  Count ........................................................................................................................................ 551
  Item ......................................................................................................................................... 551
  RemoveElement ...................................................................................................................... 551
- ElementListItem ..................................................................................................................... 553
  ElementKind .......................................................................................................................... 553
  FieldCount ............................................................................................................................ 553
  Name ....................................................................................................................................... 553
  RecordCount .......................................................................................................................... 553
- ExportSettings ......................................................................................................................... 555
  CreateKeys ............................................................................................................................. 555
  ElementList ............................................................................................................................ 555
  EntitiesToText ........................................................................................................................ 555
  ExportAllElements ................................................................................................................ 556
  ExportCompleteXML .............................................................................................................. 556
  FromAttributes ...................................................................................................................... 556
  FromSingleSubElements ....................................................................................................... 556
  FromTextValues ..................................................................................................................... 556
  IndependentPrimaryKey ......................................................................................................... 557
  Namespace ............................................................................................................................. 557
  StartFromElement .................................................................................................................. 557
  SubLevelLimit ......................................................................................................................... 557
- FileSelectionDlg ..................................................................................................................... 558
  Application ............................................................................................................................. 558
  DialogAction ........................................................................................................................ 558
  FullName .................................................................................................................................. 559
  Parent ...................................................................................................................................... 559
- FindInFilesDlg ........................................................................................................................ 560
  AdvancedXMLSearch ............................................................................................................. 560
  Application ............................................................................................................................. 561
ConsiderSampleValueHints.................................................................................. 572
ContentOfNillableElementsIsNonMandatory....................................................... 572
FillAttributesWithSampleData.......................................................................... 572
FillElementsWithSampleData........................................................................... 572
FillWithSampleData - obsolete.......................................................................... 573
LocalNameOfRootElement................................................................................... 573
NamespaceURIOfRootElement............................................................................ 573
NonMandatoryAttributes.................................................................................... 573
NonMandatoryElements.................................................................................... 573
Optimization - obsolete................................................................................... 574
OptionsDialogAction........................................................................................ 574
Parent .............................................................................................................. 574
RepeatCount...................................................................................................... 574
SampleValueHints............................................................................................. 575
SchemaOrDTDAssignment.................................................................................. 575
TakeFirstChoice - obsolete............................................................................... 575
TryToUseNonAbstractTypes............................................................................. 575
− GridView ...................................................................................................... 576
  Events ......................................................................................................... 576
    OnBeforeDrag........................................................................................... 576
    OnBeforeDrop........................................................................................... 576
    OnBeforeStartEditing.............................................................................. 577
    OnEditingFinished.................................................................................... 577
    OnFocusChanged...................................................................................... 578
CurrentFocus ................................................................................................... 578
Deselect .......................................................................................................... 579
IsVisible ........................................................................................................... 579
Select .............................................................................................................. 579
SetFocus ......................................................................................................... 579
− SchemaDocumentationDialog ....................................................................... 580
  AllDetails ..................................................................................................... 581
  Application................................................................................................... 581
  CreateDiagramsFolder.................................................................................. 581
  DiagramFormat............................................................................................. 582
  EmbedCSSInHTML....................................................................................... 582
  EmbedDiagrams.......................................................................................... 582
  GenerateRelativeLinks.................................................................................. 583
  IncludeAll .................................................................................................... 583
  IncludeAttributeGroups............................................................................... 583
  IncludeComplexTypes................................................................................... 583
  IncludeGlobalAttributes.............................................................................. 584
  IncludeGlobalElements............................................................................... 584
  IncludeGroups............................................................................................. 584
  IncludeIndex................................................................................................ 585
  IncludeLocalAttributes............................................................................... 585
  IncludeLocalElements................................................................................. 585
SpyProjectItems .................................................. 599
AddFile ............................................................... 599
AddFolder ............................................................. 599
AddURL ................................................................. 599
Count ................................................................. 600
Item .................................................................. 600
RemoveItem .......................................................... 600

TextImportExportSettings ..................................... 601
CommentIncluded .................................................. 601
DestinationFolder ................................................ 601
EnclosingCharacter .............................................. 601
Encoding ............................................................. 602
EncodingByteOrder .............................................. 602
FieldDelimiter .................................................... 602
FileExtension ..................................................... 602
HeaderRow ........................................................ 602
ImportFile .......................................................... 603
RemoveDelimiter ................................................ 603
RemoveNewline ................................................... 603

TextView ................................................................ 604
Events ................................................................ 604
OnBeforeShowSuggestions ..................................... 604
OnChar ................................................................. 605
Application ........................................................ 605
GetRangeText ...................................................... 606
GoToLineChar ...................................................... 606
Length ................................................................ 606
LineCount .......................................................... 606
LineFromPosition ............................................... 606
LineLength ........................................................ 607
MoveCaret .......................................................... 607
Parent ................................................................ 607
PositionFromLine ............................................... 607
ReplaceText ........................................................ 607
SelectionEnd ....................................................... 608
SelectionStart ..................................................... 608
SelectText .......................................................... 608
SetText ............................................................... 608
Text .................................................................. 609

WSDLDocumentationDlg ....................................... 610
AllDetails ........................................................... 611
Application ........................................................ 611
CreateDiagramsFolder ........................................ 611
DiagramFormat ................................................... 612
EmbedCSSInHTML ................................................ 612
EmbedDiagrams................................................................................................. 612
GlobalElementsAndTypesOnly........................................................................... 613
IncludeAll ......................................................................................................... 613
IncludeBinding................................................................................................. 613
IncludeImportedWSDLFiles............................................................................. 613
IncludeMessages............................................................................................... 614
IncludeOverview.............................................................................................. 614
IncludePortType............................................................................................... 614
IncludeService................................................................................................. 614
IncludeTypes.................................................................................................... 615
MultipleOutputFiles.......................................................................................... 615
OptionsDialogAction......................................................................................... 615
OutputFile......................................................................................................... 616
OutputFileDialogAction..................................................................................... 616
OutputFormat.................................................................................................... 616
Parent................................................................................................................ 617
SeparateSchemaDocument................................................................................. 617
ShowBindingDiagram....................................................................................... 617
ShowExtensibility............................................................................................. 618
ShowMessageParts........................................................................................... 618
ShowPort.......................................................................................................... 618
ShowPortTypeDiagram..................................................................................... 618
ShowPortTypeOperations.................................................................................. 619
ShowProgressBar............................................................................................. 619
ShowResult....................................................................................................... 619
ShowServiceDiagram...................................................................................... 620
ShowSourceCode............................................................................................. 620
ShowTypesDiagram......................................................................................... 620
ShowUsedBy..................................................................................................... 621
UseFixedDesign................................................................................................ 621
SPSFile.............................................................................................................. 621
- WSDL20DocumentationDlg............................................................................. 622
- AllDetails....................................................................................................... 623
- Application.................................................................................................... 623
- CreateDiagramsFolder................................................................................... 623
- DiagramFormat............................................................................................... 623
- EmbedCSSInHTML........................................................................................ 624
- EmbedDiagrams............................................................................................. 624
- GlobalElementsAndTypesOnly...................................................................... 624
- IncludeAll...................................................................................................... 624
- IncludeBinding.............................................................................................. 625
- IncludeImportedWSDLFiles.......................................................................... 625
- IncludeInterface........................................................................................... 625
- IncludeOverview........................................................................................... 626
- IncludeService.............................................................................................. 626
IncludeTypes................................................................. 626
MultipleOutputFiles......................................................... 627
OptionsDialogAction......................................................... 627
OutputFile ....................................................................... 627
OutputFileDialogAction..................................................... 628
OutputFormat...................................................................... 628
Parent .............................................................................. 628
SeparateSchemaDocument.................................................. 629
ShowBindingDiagram......................................................... 629
ShowEndpoint .................................................................. 629
ShowExtensibility.............................................................. 629
ShowFault ....................................................................... 630
ShowInterfaceDiagram....................................................... 630
ShowOperation ................................................................. 630
ShowProgressBar .............................................................. 631
ShowResult ..................................................................... 631
ShowServiceDiagram........................................................ 631
ShowSourceCode ............................................................. 632
ShowTypesDiagram........................................................... 632
ShowUsedBy .................................................................... 632
SPSFile .......................................................................... 633
UseFixedDesign ................................................................ 633
- XBRLDocumentationDlg.................................................. 634
AllDetails ........................................................................ 635
Application ...................................................................... 635
CreateDiagramsFolder....................................................... 635
DiagramFormat .................................................................. 636
EmbedCSSInHTML............................................................. 636
EmbedDiagrams................................................................. 636
IncludeAll ........................................................................ 637
IncludeCalculationLinkroles ............................................. 637
IncludeDefinitionLinkroles .............................................. 637
IncludeGlobalElements .................................................... 637
IncludeNamespacePrefixes .............................................. 638
IncludeOverview ............................................................. 638
IncludePresentationLinkroles ........................................... 638
OptionsDialogAction......................................................... 638
OutputFile ....................................................................... 639
OutputFileDialogAction..................................................... 639
OutputFormat...................................................................... 640
Parent .............................................................................. 640
ShortQualifiedNam e ........................................................... 640
ShowAbstract .................................................................. 640
ShowBalance .................................................................... 641
ShowDiagram ................................................................... 641
3.3.3 Enumerations ........................................................................................................ 658
   - ENUMApplicationStatus ...................................................................................... 659
   - SPYAttributeTypeDefinition ............................................................................... 660
   - SPYAuthenticActions ......................................................................................... 661
   - SPYAuthenticDocumentGetPosition ..................................................................... 662
3.4 ActiveX Integration

3.4.1 Prerequisites .......................................................... 704
3.4.2 Adding the ActiveX Controls to the Toolbox ............. 706
3.4.3 Integration at Application Level ............................ 708
3.4.4 Integration at Document Level ..................................................... 711
3.4.5 ActiveX Integration Examples ..................................................... 715
  - C# ........................................................................................................ 716
    Running the Sample C# Solution.......................................................... 716
  - HTML .................................................................................................... 720
    HTML Integration at Application Level................................................. 720
      Instantiate the Control....................................................................... 720
      Add Button to Open Default Document ............................................ 720
      Connect to Custom Events.................................................................. 721
    HTML Integration at Document Level................................................ 721
      Instantiate the AuthenticDesktopControl......................................... 722
      Create Editor Window ....................................................................... 722
      Create Project Window ..................................................................... 722
      Create Placeholder for Helper Windows ......................................... 722
  - Java ...................................................................................................... 724
    Example Java Project........................................................................... 725
    Creating the ActiveX Controls............................................................. 727
    Loading Data in the Controls............................................................... 728
    Basic Event Handling ........................................................................ 728
    Menus ................................................................................................... 729
    UI Update Event Handling .................................................................. 731
    Creating an XML Tree ........................................................................ 731
3.4.6 Command Reference ..................................................................... 734
  - "File" Menu ......................................................................................... 735
  - "Edit" Menu ....................................................................................... 736
  - "Project" Menu .................................................................................. 737
  - "XML" Menu ...................................................................................... 738
  - "XSL/XQuery" Menu .......................................................................... 739
  - "Authentic" Menu ............................................................................. 740
  - "View" Menu ...................................................................................... 742
  - "Browser" Menu ................................................................................. 743
  - "Tools" Menu ..................................................................................... 744
  - "Window" Menu ............................................................................... 745
  - "Help" Menu ...................................................................................... 746
3.4.7 Object Reference ........................................................................... 747
  - Authentic DesktopCommand ............................................................. 748
    Accelerator ......................................................................................... 748
    ID .......................................................................................................... 748
    IsSeparator ......................................................................................... 749
    Label ..................................................................................................... 749
    Name .................................................................................................... 749
    StatusText .......................................................................................... 749
    SubCommands .................................................................................... 749
    ToolTip ................................................................................................. 749
  - Authentic DesktopCommands ............................................................ 751
    Count .................................................................................................... 751
<table>
<thead>
<tr>
<th>Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>- AuthenticDesktopControl</td>
<td>752</td>
</tr>
<tr>
<td>Properties</td>
<td>752</td>
</tr>
<tr>
<td>Appearance</td>
<td>752</td>
</tr>
<tr>
<td>Application</td>
<td>753</td>
</tr>
<tr>
<td>BorderStyle</td>
<td>753</td>
</tr>
<tr>
<td>CommandsList</td>
<td>753</td>
</tr>
<tr>
<td>EnableUserPrompts</td>
<td>754</td>
</tr>
<tr>
<td>IntegrationLevel</td>
<td>754</td>
</tr>
<tr>
<td>MainMenu</td>
<td>754</td>
</tr>
<tr>
<td>Toolbars</td>
<td>755</td>
</tr>
<tr>
<td>Methods</td>
<td>756</td>
</tr>
<tr>
<td>Exec</td>
<td>756</td>
</tr>
<tr>
<td>Open</td>
<td>756</td>
</tr>
<tr>
<td>QueryStatus</td>
<td>756</td>
</tr>
<tr>
<td>Events</td>
<td>757</td>
</tr>
<tr>
<td>OnCloseEditingWindow</td>
<td>757</td>
</tr>
<tr>
<td>OnContextMenuChanged</td>
<td>757</td>
</tr>
<tr>
<td>OnDocumentOpened</td>
<td>757</td>
</tr>
<tr>
<td>OnFileChangedAlert</td>
<td>758</td>
</tr>
<tr>
<td>OnLicenseProblem</td>
<td>758</td>
</tr>
<tr>
<td>OnOpenedOrFocused</td>
<td>758</td>
</tr>
<tr>
<td>OnToolWindowUpdated</td>
<td>759</td>
</tr>
<tr>
<td>OnUpdateCmdUI</td>
<td>759</td>
</tr>
<tr>
<td>OnValidationWindowUpdated</td>
<td>759</td>
</tr>
<tr>
<td>- AuthenticDesktopControlDocument</td>
<td>760</td>
</tr>
<tr>
<td>Properties</td>
<td>760</td>
</tr>
<tr>
<td>Appearance</td>
<td>760</td>
</tr>
<tr>
<td>BorderStyle</td>
<td>761</td>
</tr>
<tr>
<td>Document</td>
<td>761</td>
</tr>
<tr>
<td>IsModified</td>
<td>761</td>
</tr>
<tr>
<td>Path</td>
<td>761</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>761</td>
</tr>
<tr>
<td>Methods</td>
<td>762</td>
</tr>
<tr>
<td>Exec</td>
<td>762</td>
</tr>
<tr>
<td>New</td>
<td>762</td>
</tr>
<tr>
<td>Open</td>
<td>762</td>
</tr>
<tr>
<td>QueryStatus</td>
<td>763</td>
</tr>
<tr>
<td>Reload</td>
<td>763</td>
</tr>
<tr>
<td>Save</td>
<td>763</td>
</tr>
<tr>
<td>SaveAs</td>
<td>763</td>
</tr>
<tr>
<td>Events</td>
<td>764</td>
</tr>
<tr>
<td>OnActivate</td>
<td>764</td>
</tr>
<tr>
<td>OnContextMenuChanged</td>
<td>764</td>
</tr>
<tr>
<td>OnDocumentClosed</td>
<td>764</td>
</tr>
<tr>
<td>OnDocumentOpened</td>
<td>764</td>
</tr>
<tr>
<td>OnFileChangedAlert</td>
<td>765</td>
</tr>
<tr>
<td>OnModifiedFlagChanged</td>
<td>765</td>
</tr>
<tr>
<td>OnSetEditorTitle</td>
<td>766</td>
</tr>
<tr>
<td>- AuthenticDesktopControlPlaceHolder</td>
<td>766</td>
</tr>
<tr>
<td>Properties</td>
<td>766</td>
</tr>
</tbody>
</table>
4 Appendices 772

4.1 Technical Data ........................................................................................................ 773
  4.1.1 OS and Memory Requirements ................................................................. 774
  4.1.2 Altova Engines ....................................................................................... 775
  4.1.3 Unicode Support ................................................................................... 776
  4.1.4 Internet Usage ....................................................................................... 777

4.2 License Information ................................................................................................ 778
  4.2.1 Electronic Software Distribution ............................................................ 779
  4.2.2 Altova End-User License Agreement for Authentic ............................... 780

Index
Chapter 1

Altova Authentic 2019 Desktop Enterprise Edition
Altova Authentic 2019 Desktop Enterprise Edition

Altova Authentic 2019 Desktop Enterprise Edition is an innovative visual approach to authoring XML documents that shields the end-user from having to deal with the technical aspects of XML. Authentic Desktop runs on Windows 7 SP1 with Platform Update, Windows 8, Windows 10, and Windows Server 2008 R2 SP1 with Platform Update or newer. Authentic Desktop Enterprise Edition is available for 64-bit and 32-bit machines.

Altova website: XML content editing, XML authoring

Last updated: 28 March 2019
Chapter 2
User Guide and Reference
User Guide and Reference

This User Manual contains a tutorial and explanation of the various Authentic View features to get you started. It also contains a comprehensive reference section that describes the features of the interface. It consists of the following sections:

- An introduction that describes the GUI and the Authentic Desktop environment.
- A tutorial to get you started using Authentic Desktop.
- A description of Authentic View, which is a WYSIWYG view of an XML document. Authentic View enables users to write and edit XML documents as if they were simple text documents or interactive forms. The XML markup is hidden from users, thus enabling them to concentrate on document content. Authentic View is the main view of Authentic Desktop.
- A description of Browser View, in which the XML document is transformed on the fly and presented in a browser window.
- An explanation of Altova's Global Resources feature, which enables resources to be quickly switched from one to the other.
- Explanations of how Authentic Desktop can be used in Visual Studio and Eclipse.
- A User Reference that contains a description of all windows and menu commands available in Authentic Desktop.

File paths in Windows 7, Windows 8, and Windows 10

File paths given in this documentation will not be the same for all operating systems. You should note the following correspondences:

- (My) Documents folder: Located by default at the following locations. Example files are located in a sub-folder of this folder.

  | Windows 7/8/10 | C:\Users\<username>\Documents |

- Application folder: The Application folder is the folder where your Altova application is located. The path to the Application folder is, by default, the following.

  | Windows 7/8/10 | C:\Program Files\Altova\ |
  | 32-bit version on 64-bit OS | C:\Program Files (x86)\Altova\ |

Note: Authentic Desktop is also supported on Windows Server 2008 R2 SP1 with Platform Update or newer.
1 Interface and Environment

This section describes:

- The application GUI, and
- The application environment.

The GUI section starts off by presenting an overview of the GUI and then goes on to describe each of the various GUI windows in detail. It also shows you how to re-size, move, and otherwise work with the windows and the GUI.

The Application Environment section points out the various settings that control how files are displayed and can be edited. It also explains how and where you can customize your application. In this section, you will learn where important example and tutorial files have been installed on your machine, and, later in the section, you are linked to the Altova website, where you can explore the feature matrix of your application, learn about the multiple formats of your user manual, find out about the various support options available to you, and discover other products in the Altova range.
1.1 The Graphical User Interface (GUI)

The Graphical User Interface (GUI) consists of a Main Window and several sidebars (see illustration below). By default, the sidebars are located around the Main Window and are organized into the following groups:

- Project Window
- Info Window
- Entry Helpers: Elements, Attributes, Entities, etc (depending upon the type of document currently active)
- Output Windows: Messages

The main window and sidebars are described in the sub-sections of this section.

Switching on and off the display of sidebars

Sidebar groups (Project Window, Info Window, Entry Helpers, Output Windows) can be displayed or hidden by toggling them on and off via the commands in the Window menu. A displayed sidebar (or a group of tabbed sidebars) can also be hidden by right-clicking the title bar of the displayed sidebar (or tabbed-sidebar group) and selecting the command Hide.

Floating and docking the sidebars

An individual sidebar window can either float free of the GUI or be docked within the GUI. When a floating window is docked, it docks into its last docked position. A window can also be docked as
a tab within another window.

A window can be made to float or dock using one of the following methods:

- Right-click the title bar of a window and choose the required command (Floating or Docking).
- Double-click the title bar of the window. If docked, the window will now float. If floating, the window will now dock in the last position in which it was docked.
- Drag the window (using its title bar as a handle) out of its docked position so that it floats. Drag a floating window (by its title bar) to the location where it is to be docked. Two sets of blue arrows appear. The outer set of four arrows enables docking relative to the application window (along the top, right, bottom, or left edge of the GUI). The inner set of arrows enables docking relative to the window over which the cursor is currently placed. Dropping a dragged window on the button in the center of the inner set of arrows (or on the title bar of a window) docks the dragged window as a tabbed window within the window in which it is dropped.

To float a tabbed window, double-click its tab. To drag a tabbed window out of a group of tabbed windows, drag its tab.

**Auto-hiding sidebars**

The Auto-hide feature enables you to minimize docked sidebars to buttons along the edges of the application window. This gives you more screen space for the Main Window and other sidebars. Scrolling over a minimized sidebar rolls out that sidebar.

To auto-hide and restore sidebars click the drawing pin icon in the title bar of the sidebar window (or right-click the title bar and select **Auto-Hide**).
1.1.1 Main Window

The Main Window (screenshot below) is where you view and edit documents.

Files in the Main Window

- Any number of files can be opened and edited at once.
- Each open document has its own window and a tab (containing the document's file name) at the bottom of the Main Window. To make an open document active, click its tab.
- If several files are open, some document tabs might not be visible for lack of space in the document tabs bar. Document tabs can be brought into view by: (i) using the scroll buttons at the right of the document tab bar, or (ii) selecting the required document from the list at the bottom of the Window menu.
- When the active document is maximized, its Minimize, Restore, and Close buttons are located at the right side of the Menu Bar. When a document is cascaded, tiled, or minimized, the Maximize, Restore, and Close buttons are located in the title bar of the document window.
- When you maximize one file, all open files are maximized.
- Open files can be cascaded or tiled using commands in the Window menu.
- You can also activate open files in the sequence in which they were opened by using Ctrl + Tab or Ctrl+F6.
- Right-clicking a document tab opens a context-menu with a selection of File commands, such as Print and Close.

Views in the Main Window

The active document can be displayed and edited in multiple views. The available views are displayed in a bar above the document tabs (see illustration above), and the active view is highlighted. A view is selected by clicking the required view button or by using the commands in the View menu.

The available views are either editing or browser views:

- **Authentic View**: For editing XML documents that are based on StyleVision Power Stylesheets in a graphical interface.
- **Browser View**: An integrated browser view that supports both CSS and XSL stylesheets.
**Note:** The default view for individual file extensions can be customized in the **Tools | Options** dialog: in the Default View pane of the File Types tab.
1.1.2 Project Window

A project is a collection of files that are related to each other in some way you determine. For example, in the screenshot below, a project named Examples collects the files for various examples in separate example folders, each of which can be further organized into sub-folders. Within the Examples project, for instance, the OrgChart example folder is further organized into sub-folders for XML, XSL, and Schema files.

Projects thus enable you to gather together files that are used together and to access them quicker. Additionally, you can define schemas and XSLT files for individual folders, thus enabling the batch processing of files in a folder.

Project operations

Commands for folder operations are available in the Project menu, and some commands are available in the context menus of the project and its folders (right-click to access).

- One project is open at a time in the Project Window. When a new project is created or an existing project opened, it replaces the project currently open in the Project Window.
- After changes have been made to a project, the project must be saved (by clicking the Project | Save Project command). A project with unsaved changes is indicated with an asterisk next to its name (see screenshot above).
- The project has a tree structure composed of folders, files, and other resources. Such resources can be added at any level and to an unlimited depth.
- Project folders are semantic folders that represent a logical grouping of files. They do not need to correspond to any hierarchical organization of files on your hard disk.
- Folders can correspond to, and have a direct relationship to, physical directories on your file system. We call such folders external folders, and they are indicated in the Project Window by a yellow folder icon (as opposed to normal project folders, which are green).
External project folders must be explicitly synchronized by using the Refresh command.

- A folder can contain an arbitrary mix of file-types. Alternatively, you can define file-type extensions for each folder (in the Properties dialog of that folder) to keep common files in one convenient place. When a file is added to the parent folder, it is automatically added to the sub-folder that has been defined to contain files of that file extension.
- In the Project Window, a folder can be dragged to another folder or to another location within the same folder, while a file can be dragged to another folder but cannot be moved within the same folder (within which files are arranged alphabetically). Additionally, files and folders can be dragged from Windows File Explorer to the Project Window.
- Each folder has a set of properties that are defined in the Properties dialog of that folder. These properties include file extensions for the folder, the schema by which to validate XML files, the XSLT file with which to transform XML files, etc.
- Batch processing of files in a folder is done by right-clicking the folder and selecting the relevant command from the context menu (for example, Validate XML or Check Well-Formedness).

**Note:** The display of the Project Window can be turned on and off in the Window menu.
1.1.3 Info Window

The Info Window (screenshot below) shows information about the element or attribute in which the cursor is currently positioned.

The display of the Info Window can be turned on and off in the **Window** menu.
1.1.4 Entry Helpers

Entry helpers are an intelligent editing feature that helps you to create valid XML documents quickly. When you are editing a document, the entry helpers display structural editing options according to the current location of the cursor. The entry helpers get the required information from the underlying DTD, XML Schema, and/or StyleVision Power Stylesheet, etc. If, for example, you are editing an XML data document, then the elements, attributes, and entities that can be inserted at the current cursor position are displayed in the relevant entry helpers windows.

Note the following:

- You can turn the display of entry helpers on or off with the menu option Window | Entry Helpers.
1.1.5 Output Window: Messages

The Messages Window displays messages about actions carried out in Authentic Desktop as well as errors and other output. For example, if an XML document is validated and is valid, a successful validation message is displayed in the Messages Window. Otherwise, a message that describes the error is displayed. Notice that there are links (black link text) to nodes and node content in the XML document, as well as links (blue link text) to the sections in the relevant specification on the Internet that describe the rule in question.

Validating folders and files in the Project window

The Validate command (in the XML menu) is normally applied to the active document. But you can also apply the command to a file, folder, or group of files in the active project. Select the required file or folder in the Project Window (by clicking on it), and click XML | Validate XML or F8. Invalid files in a project will be opened and made active in the Main Window, and the File is not valid error message will be displayed.

Note: You can also carry out the Well-Formedness check (Check Well-Formedness or F7) in the Project window.
1.1.6 Menu Bar, Toolbars, Status Bar

Menu Bar

The menu bar (see illustration) contains the various application menus. The following conventions apply:

- If commands in a menu are not applicable in a view or at a particular location in the document, they are unavailable.
- Some menu commands pop up a submenu with a list of additional options. Menu commands with submenus are indicated with a right-pointing arrowhead to the right of the command name.
- Some menu commands pop up a dialog that prompts you for further information required to carry out the selected command. Such commands are indicated with an ellipsis (…) after the name of the command.
- To access a menu command, click the menu name and then the command. If a submenu is indicated for a menu item, the submenu opens when you mouseover the menu item. Click the required sub-menu item.
- A menu can be opened from the keyboard by pressing the appropriate key combination. The key combination for each menu is Alt+KEY, where KEY is the underlined letter in the menu name. For example, the key combination for the File menu is Alt+F.
- A menu command (that is, a command in a menu) can be selected by sequentially selecting (i) the menu with its key combination (see previous point), and then (ii) the key combination for the specific command (Alt+KEY, where KEY is the underlined letter in the command name). For example, to create a new file (File | New), press Alt+F and then Alt+N.
- Some menu commands can be selected directly by pressing a special shortcut key or key combination (Ctrl+KEY). Commands which have shortcuts associated with them are indicated with the shortcut key or key combination listed to the right of the command. For example, you can use the shortcut key combination Ctrl+N to create a new file; the shortcut key F8 to validate an XML file. You can create your own shortcuts in the Keyboard tab of the Customize dialog (Tools | Customize).

Toolbars

The toolbars (see illustration) contain icons that are shortcuts for selecting menu commands. The name of the command appears when you place your mouse pointer over the icon. To execute the command, click the icon.

Toolbar buttons are arranged in groups. In the Tools | Customize | Toolbars dialog, you can specify which toolbar groups are to be displayed. These settings apply to the current view. To make a setting for another view, change to that view and then make the setting in the Tools | Customize | Toolbars. In the GUI, you can also drag toolbar groups by their handles (or title bars) to alternative locations on the screen. Double-clicking the handle causes the toolbar to undock and to float; double-clicking its title bar causes the toolbar to dock at its previous location.

Status Bar

The Status Bar is located at the bottom of the application window (see illustration) and displays (i) status information about the loading of files, and (ii) information about menu commands and command shortcuts in the toolbars when the mouse cursor is placed over these. If you are using the 64-bit version of Authentic Desktop, this is indicated in the status bar with the suffix (x64) after the application name. There is no suffix for the 32-bit version.
1.2 The Application Environment

In this section we describe various aspects of the application that are important for getting started. Reading through this section will help you familiarize yourself with Authentic Desktop and get you off to a confident start. It contains important information about settings and customization, which you should read for a general idea of the range of settings and customization options available to you and how these can be changed.

This section is organized as follows:

- **Settings and Customization**: Describes how and where important settings and customization options can be defined.
- **Tutorials, Projects, Examples**: Notes the location of the various non-program files included in the application package.
- **Product features and documentation, and Altova products**: Provides links to the Altova website, where you can find information about product features, additional Help formats, and other Altova products.
1.2.1 Settings and Customization

This section is organized into the following parts.

- Settings
- Customization

Settings

Several important Authentic Desktop settings are defined in different tabs in the Options dialog. You should look through the various options to familiarize yourself with what's available.

Customization

You can also customize various aspects of Authentic Desktop, including the appearance of the GUI. These customization options are available in the Customize dialog (accessed via the menu command Tools | Customize).

The various customization options are described in the User Reference section.
1.2.2 Tutorials, Projects, Examples

The Authentic Desktop installation package contains tutorials, projects, and example files.

Location of tutorials, projects, and example files
The Authentic Desktop tutorials, projects, and example files are installed in the folder:

C:\Documents and Settings\<username>\My Documents\Altova\Authentic2019\AuthenticExamples\

The My Documents\Altova\Authentic2019 folder will be installed for each user registered on a PC within that user's <username> folder. Under this installation system, therefore, each user will have his or her own AuthenticExamples folder in a separate working area.

Note about the master Authentic Desktop folder
When Authentic Desktop is installed on a machine, a master Altova\Authentic2019 folder is created at the following folder location:

C:\Documents and Settings\All Users\Application Data\

When a user on that machine starts Authentic Desktop for the first time, Authentic Desktop creates a copy of this master folder in the user's <username>\My Documents\ folder. It is therefore important not to use the master folder when working with tutorial or example files, otherwise these edited files will be copied to the user folder of a user who subsequently uses Authentic Desktop for the first time.

Location of tutorial, project, and examples files
All tutorial, project, and example files are located in the AuthenticExamples folder.
1.2.3 **Authentic Desktop Features and Help, and Altova Products**

The Altova website, [www.altova.com](http://www.altova.com), has a wealth of Authentic Desktop-related information and resources. Among these are the following.

**Authentic Desktop feature listing**
The Altova website carries a list of Authentic Desktop features.

**Authentic Desktop Help**
This documentation is the Altova-supplied Help for Authentic Desktop. It is available as the built-in Help system of Authentic Desktop, which is accessible via the Help menu or by pressing F1. Additionally, the user manuals for all Altova products are available in the following formats:

- **Online HTML manuals**, accessed via the Support page at the Altova website
- **Printable PDFs**, which you can download from the Altova website and print locally
- **Printed books** that you can buy via a link at the Altova website

**Support options**
If you require additional information to what is available in the user manual (this documentation) or have a query about Altova products, visit our Support Center at the Altova website. Here you will find:

- Links to our FAQ pages
- Discussion forums on Altova products and general XML subjects
- Online Support Forms that enable you to make support requests, should you have a support package. Your support request will be processed by our support team.

**Altova products**
For a list of all Altova products, see the [Altova website](http://www.altova.com).
2 Authentic View Tutorial

In Authentic View, you can edit XML documents in a graphical WYSIWYG interface (screenshot below), just like in word-processor applications such as Microsoft Word. In fact, all you need to do is enter data. You do not have to concern yourself with the formatting of the document, since the formatting is already defined in the stylesheet that controls the Authentic View of the XML document. The stylesheet (StyleVision Power Stylesheet, shortened to SPS in this tutorial) is created by a stylesheet designer using Altova’s StyleVision product.

Nanonull, Inc.

Location: US

| Street: 119 Oakstreet, Suite 4876 | Phone: +1 (321) 555 5155 0 |
| City: Vereno | Fax: +1 (321) 555 5155 4 |
| State & Zip: DC 29213 | E-mail: office@nanonull.com |

Vereno Office Summary: 4 departments, 15 employees.

The company was established in Vereno in 1995 as a privately held software company. Since 1996, Nanonull has been actively involved in developing nanoelectronic software technologies. It released the first version of its acclaimed NanoSoft Development Suite in February 1999. Also in 1999, Nanonull increased its capital base with investment from a consortium of private investment firms. The company has been expanding rapidly ever since.

Editing an XML document in Authentic View involves two user actions: (i) editing the structure of the document (for example, adding or deleting document parts, such as paragraphs and headlines); and (ii) entering data (the content of document parts).

This tutorial takes you through the following steps:

- Opening an XML document in Authentic View. The key requirement for Authentic View editing is that the XML document be associated with an SPS file.
- A look at the Authentic View interface and a broad description of the central editing mechanisms.
- Editing document structure by inserting and deleting nodes.
- Entering data in the XML document.
- Entering (i) attribute values via the Attributes entry helper, and (ii) entity values.
- Printing the document.

Remember that this tutorial is intended to get you started, and has intentionally been kept simple. You will find additional reference material and feature descriptions in the Authentic View interface.
section.

Tutorial requirements

All the files you need for the tutorial are in the Examples folder of your Altova application folder. These files are:

- NanonullOrg.xml (the XML document you will open)
- NanonullOrg.sps (the StyleVision Power Stylesheet to which the XML document is linked)
- NanonullOrg.xsd (the XML Schema on which the XML document and StyleVision Power Stylesheet are based, and to which they are linked)
- nanonull.gif and Altova_right_300.gif (two image files used in the tutorial)

Please note: At some points in the tutorial, we ask you to look at the XML text of the XML document (as opposed to the Authentic View of the document). If the Altova product edition you are using does not include a Text View (as with Authentic Desktop and Authentic Browser), then use a plain text editor like Wordpad or Notepad to view the text of the XML document.

Caution: We recommend that you use a copy of NanonullOrg.xml for the tutorial, so that you can always retrieve the original should the need arise.
2.1 Opening an XML Document in Authentic View

In Authentic View, you can edit an existing XML document or create and edit a new XML document. In this tutorial, you will open an existing XML document in Authentic View (described in this section) and learn how you can edit it (subsequent sections). Additionally, in this section is a description of how a new XML document can be created for editing in Authentic View.

Opening an existing XML document

The file you will open is `NanonullOrg.xml`. It is in the Examples folder of your Altova application. You can open `NanonullOrg.xml` in one of two ways:

- Click File | Open in your Altova product, then browse for `NanonullOrg.xml` in the dialog that appears, and click Open.
- Use Windows Explorer to locate the file, right-click, and select your Altova product as the application with which to open the file.

The file `NanonullOrg.xml` opens directly in Authentic View (screenshot below).

![NanonullOrg.xml](nanonull.gif)

Remember: It is the SPS that defines and controls how an XML document is displayed in Authentic View. Without an SPS, there can be no Authentic View of the document.

Creating a new XML document based on an SPS

You can also create a new XML document that is based on an SPS. You can do this in two ways: via the File | New menu command and via the Authentic | New Document menu command. In both cases an SPS is selected.

Via File | New

1. Select File | New, and, in the Create a New Document dialog, select XML as the new file type to create.
2. Click Select a STYLEVISION Stylesheet, and browse for the desired SPS.
Via Authentic | New Document

1. Select **Authentic | New Document**.
2. In the Create a New Document dialog, browse for the desired SPS.

If a Template XML File has been assigned to the SPS, then the data in the Template XML File is used as the starting data of the XML document template created in Authentic View.
2.2 The Authentic View Interface

The Authentic View editing interface consists of a main window in which you enter and edit the document data, and three entry helpers. Editing a document is simple. If you wish to see the markup of the document, switch on the markup tags. Then start typing in the content of your document. To modify the document structure, you can use either the context menu or the Elements entry helper.

Displaying XML node tags (document markup)

An XML document is essentially a hierarchy of nodes. For example:

```xml
<DocumentRoot>
  <Person id="ABC001">
    <Name>Alpha Beta</Name>
    <Address>Some Address</Address>
    <Tel>1234567</Tel>
  </Person>
</DocumentRoot>
```

By default, the node tags are not displayed in Authentic View. You can switch on the node tags by selecting the menu item Authentic | Show Large Markup (or the toolbar icon). Large markup tags contain the names of the respective nodes. Alternatively, you can select small markup (no node names in tags) and mixed markup (a mixture of large, small, and no markup tags, which is defined by the designer of the stylesheet; the default mixed markup for the document is no markup).

You can view the text of the XML document in the Text View of your Altova product or in a text editor.

Entry helpers

There are three entry helpers in the interface (screenshot below), located by default along the right edge of the application window. These are the Elements, Attributes, and Entity entry helpers.
Elements entry helper: The Elements entry helper displays elements that can be inserted and removed with reference to the current location of the cursor or selection in the Main Window. Note that the entry helper is context-sensitive; its content changes according to the location of the cursor or selection. The content of the entry helper can be changed in one other way: when another node is selected in the XML tree of the Elements entry helper, the elements relevant to that node are displayed in the entry helper. The Elements entry helper can be expanded to show the XML tree by checking the Show XML Tree check box at the top of the entry helper (see screenshot above). The XML tree shows the hierarchy of nodes from the top-level element node all the way down to the node selected in the Main Window.

Attributes entry helper: The Attributes entry helper displays the attributes of the element selected in the Main Window, and the values of these attributes. Attribute values can be entered or edited in the Attributes entry helper. Element nodes from the top-level element down to the selected element are available for selection in the combo box of the Attributes entry helper. Selecting an element from the dropdown list of the combo box causes that element’s attributes to be displayed in the entry helper, where they can then be edited.

Entities entry helper: The Entities entry helper is not context-sensitive, and displays all the
entities declared for the document. Double-clicking an entity inserts it at the cursor location. How to add entities for a document is described in the section Authentic View interface.

Context menu
Right-clicking at a location in the Authentic View document pops up a context menu relevant to that (node) location. The context menu provides commands that enable you to:

- Insert nodes at that location or before or after the selected node. Submenus display lists of nodes that are allowed at the respective insert locations.
- Remove the selected node (if this allowed by the schema) or any removable ancestor element. The nodes that may be removed (according to the schema) are listed in a submenu.
- Insert entities and CDATA sections. The entities declared for the document are listed in a submenu. CDATA sections can only be inserted within text.
- Cut, copy, paste (including pasting as XML or text), and delete document content.

**Note:** For more details about the interface, see Authentic View interface
2.3 Node Operations

There are two major types of nodes you will encounter in an Authentic View XML document: element nodes and attribute nodes. These nodes are marked up with tags, which you can switch on. There are also other nodes in the document, such as text nodes (which are not marked up) and CDATA section nodes (which are marked up, in order to delimit them from surrounding text).

The node operations described in this section refer only to element nodes and attribute nodes. When trying out the operations described in this section, it is best to have large markup switched on.

Note: It is important to remember that only same- or higher-level elements can be inserted before or after the selected element. Same-level elements are siblings. Siblings of a paragraph element would be other paragraph elements, but could also be lists, a table, an image, etc. Siblings could occur before or after an element. Higher-level elements are ancestor elements and siblings of ancestors. For a paragraph element, ancestor elements could be a section, chapter, article, etc. A paragraph in a valid XML file would already have ancestors. Therefore, adding a higher-level element in Authentic View, creates the new element as a sibling of the relevant ancestor. For example, if a section element is inserted after a paragraph, it is created as a sibling of the section that contains the current paragraph element.

Carrying out node operations

Node operations can be carried out by selecting a command in the context menu or by clicking the node operation entry in the Elements entry helper. In some cases, an element or attribute can be added by clicking the Add Node link in the Authentic View of the document. In the special cases of elements defined as paragraphs or list items, pressing the Enter key when within such an element creates a new sibling element of that kind. This section also describes how nodes can be created and deleted by using the Apply Element, Remove Node, and Clear Element mechanisms.

Inserting elements

Elements can be inserted at the following locations:

- The cursor location within an element node. The elements available for insertion at that location are listed in a submenu of the context menu's Insert command. In the Elements entry helper, elements that can be inserted at a location are indicated with the icon.
  In the NonnullOrg.xml document, place the cursor inside the para element, and create bold and italic elements using both the context menu and Elements entry helper.
- Before or after the selected element or any of its ancestors, if allowed by the schema. Select the required element from the submenu/s that roll out. In the Elements entry helper, elements that can be inserted before or after the selected element are indicated with the and icons, respectively. Note that in the Elements entry helper, you can insert elements before/after the selected element only; you cannot insert before/after an ancestor element. Try out this command, by first placing the cursor inside the para element and then inside the table listing the employees.
Add Node link

If an element or attribute is included in the document design, and is not present in the XML document, an Add Node link is displayed at the location in the document where that node is specified. To see this link, in the line with the text, Location of logo, select the @href node within the CompanyLogo element and delete it (by pressing the Delete key). The \texttt{add @href} link appears within the CompanyLogo element that was edited (\textit{screenshot below}). Clicking the link adds the @href node to the XML document. The text box within the @href tags appears because the design specifies that the @href node be added like this. You still have to enter the value (or content) of the @href node. Enter the text nanonull.gif.

If the content model of an element is ambiguous, for example, if it specifies that a sequence of child elements may appear in any order, then the \texttt{add...} link appears. Note that no node name is specified. Clicking the link will pop up a list of elements that may validly be inserted.

\textbf{Note:} The Add Node link appears directly in the document template; there is no corresponding entry in the context menu or Elements entry helper.

Creating new elements with the Enter key

In cases where an element has been formatted as a paragraph or list item (by the stylesheet designer), pressing the Enter key when inside such a node causes a new node of that kind to be inserted after the current node. You can try this mechanism in the NanonullOrg.xml document by going to the end of a para node (just before its end tag) and pressing Enter.

Applying elements

In elements of mixed content (those which contain both text and child elements), some text content can be selected and an allowed child element be applied to it. The selected text becomes the content of the applied element. To apply elements, in the context menu, select Apply and then select from among the applicable elements. (If no elements can be applied to the selected text, then the Apply command does not appear in the context menu.) In the Elements entry helper, elements that can be applied for a selection are indicated with the \texttt{bold} and \texttt{italic} icons. In the NanonullOrg.xml document, select text inside the mixed content para element and experiment with applying the \texttt{bold} and \texttt{italic} elements.

The stylesheet designer might also have created a toolbar icon to apply an element. In the NanonullOrg.xml document, the \texttt{bold} and \texttt{italic} elements can be applied by clicking the bold and italic icons in the application's Authentic toolbar.
Removing nodes
A node can be removed if its removal does not render the document invalid. Removing a node causes a node and all its contents to be deleted. A node can be removed using the **Remove** command in the context menu. When the Remove command is highlighted, a submenu pops up which contains all nodes that may be removed, starting from the selected node and going up to the document's top-level node. To select a node for removal, the cursor can be placed within the node, or the node (or part of it) can be highlighted. In the Elements entry helper, nodes that can be removed are indicated with the [icon]. A removable node can also be removed by selecting it and pressing the **Delete** key. In the `NanonullOrg.xml` document, experiment with removing a few nodes using the mechanisms described. You can undo your changes with **Ctrl+Z**.

Clearing elements
Element nodes that are children of elements with mixed content (both text and element children) can be cleared. The entire element can be cleared when the node is selected or when the cursor is placed inside the node as an insertion point. A text fragment within the element can be cleared of the element markup by highlighting the text fragment. With the selection made, select **Clear** in the context menu and then the element to clear. In the Elements entry helper, elements that can be cleared for a particular selection are indicated with the [icon] (insertion point selection) and [icon] (range selection). In the `NanonullOrg.xml` document, try the clearing mechanism with the **bold** and **italic** child elements of `para` (which has mixed content).

Tables and table structure
There are two types of Authentic View table:

- **SPS tables (static and dynamic)**. The broad structure of SPS table is determined by the stylesheet designer. Within this broad structure, the only structural changes you are allowed are content-driven. For example, you could add new rows to a dynamic SPS table.
- **XML tables**, in which you decide to present the contents of a particular node (say, one for person-specific details) as a table. If the stylesheet designer has enabled the creation of this node as an XML table, then you can determine the structure of the table and edit its contents. XML tables are discussed in detail in the **Tables in Authentic View** section.
2.4 Entering Data in Authentic View

Data is entered into the XML document directly in the main window of Authentic View. Additionally for attributes, data (the value of the attribute) can be entered in the Attributes entry helper. Data is entered (i) directly as text, or (ii) by selecting an option in a data-entry device, which is then mapped to a predefined text entry.

Adding text content

You can enter element content and attribute values directly as text in the main window of Authentic View. To insert content, place the cursor at the location where you want to insert the text, and type. You can also copy text from the clipboard into the document. Content can also be edited using standard editing mechanisms, such as the Caps and Delete keys. For example, you can highlight the text to be edited and type in the replacement text with the Caps key on.

For example, to change the name of the company, in the Name field of Office, place the cursor after Nanonull, and type in USA to change the name from Nanonull, Inc. to Nanonull USA, Inc.

Nanonull USA, Inc.
Location: US

If text is editable, you will be able to place your cursor in it and highlight it, otherwise you will not be able to. Try changing any of the field names (not the field values), such as "Street", "City", or "State/Zip," in the address block. You are not able to place the cursor in this text because such text is not XML content; it is derived from the StyleVision Power Stylesheet.

Inserting special characters and entities

When entering data, the following type of content is handled in a special way:

- **Special characters that are used for XML markup** (ampersand, apostrophe, greater than, less than, and quotes). These characters are available as built-in entities and can be entered in the document by double-clicking the respective entity in the Entities entry helper. If these characters occur frequently (for example, in program code listings), then they can be entered within CDATA sections. To insert a CDATA section, right-click at the location where you wish to enter the CDATA section, and select Insert CDATA Section from the context menu. The XML processor ignores all markup characters within CDATA sections. This also means that if you want a special character inside a CDATA section, you should enter that character and not its entity reference.

- **Special characters that cannot be entered via the keyboard** should be entered by copying them from the character map of your system to the required location in the document.

- **A frequently used text string** can be defined as an entity, which appears in the Entities entry helper. The entity is inserted at the required locations by placing the cursor at each required location and double-clicking the entity in the entry helper. This is useful for maintenance because the value of the text string is held in one location; if the value needs to be changed, then all that needs to be done is to change the entity definition.

Note: When markup is hidden in Authentic View, an empty element can easily be overlooked. To make sure that you are not overlooking an empty element, switch large or small.
Try using each type of text content described above.

Adding content via a data-entry device

In the content editing you have learned above, content is added by directly typing in text as content. There is one other way that element content (or attribute values) can be entered in Authentic View: via data-entry devices.

Given below is a list of data-entry devices in Authentic View, together with an explanation of how data is entered in the XML file for each device.

<table>
<thead>
<tr>
<th>Data-Entry Device</th>
<th>Data in XML File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Field (Text Box)</td>
<td>Text entered by user</td>
</tr>
<tr>
<td>Multiline Input Field</td>
<td>Text entered by user</td>
</tr>
<tr>
<td>Combo box</td>
<td>User selection mapped to value</td>
</tr>
<tr>
<td>Check box</td>
<td>User selection mapped to value</td>
</tr>
<tr>
<td>Radio button</td>
<td>User selection mapped to value</td>
</tr>
<tr>
<td>Button</td>
<td>User selection mapped to value</td>
</tr>
</tbody>
</table>

In the static table containing the address fields (shown below), there are two data-entry devices: an input field for the Zip field and a combo-box for the State field. The values that you enter in the text fields are entered directly as the XML content of the respective elements. For other data-entry devices, your selection is mapped to a value.

For the Authentic View shown above, here is the corresponding XML text:

```xml
<Address>
  <ipo:street>119 Oakstreet, Suite 4876</ipo:street>
  <ipo:city>Vereno</ipo:city>
  <ipo:state>DC</ipo:state>
  <ipo:zip>29213</ipo:zip>
</Address>
```

Notice that the combo-box selection DC is mapped to a value of DC. The value of the Zip field is entered directly as content of the ipo:zip element.
2.5 Entering Attribute Values

An attribute is a property of an element, and an element can have any number of attributes. Attributes have values. You may sometimes be required to enter XML data as an attribute value. In Authentic View, you enter attribute values in two ways:

- As content in the main window if the attribute has been created to accept its value in this way
- In the Attributes entry helper

Attribute values in the main window

Attribute values can be entered as normal text or as text in an input field, or as a user selection that will be mapped to an XML value. They are entered in the same way that element content is entered: see Entering Data in Authentic View. In such cases, the distinction between element content and attribute value is made by the StyleVision Power Stylesheet and the data is handled appropriately.

Attribute values in the Attributes Entry Helper

If you wish to enter or change an attribute value, you can also do this in the Attributes Entry Helper. First, the attribute node is selected in Authentic View, then the value of the attribute is entered or edited in the Attributes entry helper. In the NanonullOrg.xml document, the location of the logo is stored as the value of the href attribute of the CompanyLogo element. To change the logo to be used:

1. Select the CompanyLogo element by clicking a CompanyLogo tag. The attributes of the CompanyLogo element are displayed in the Attributes Entry Helper.
2. In the Attributes Entry Helper, change the value of the href attribute from nanonull.gif to Altova_right_300.gif (an image in the Examples folder).

This causes the Nanonull logo to be replaced by the Altova logo.

Note: Entities cannot be entered in the Attributes entry helper.
2.6 Adding Entities

An entity in Authentic View is typically XML data (but not necessarily), such as a single character; a text string; and even a fragment of an XML document. An entity can also be a binary file, such as an image file. All the entities available for a particular document are displayed in the Entities Entry Helper (screenshot below). To insert an entity, place the cursor at the location in the document where you want to insert it, and then double-click the entity in the Entities entry helper. Note that you cannot enter entities in the Attributes entry helper.

The ampersand character (&) has special significance in XML (as have the apostrophe, less than and greater than symbols, and the double quote). To insert these characters, entities are used so that they are not confused with XML-significant characters. These characters are available as entities in Authentic View.

In NanonullOrg.xml, change the title of Joe Martin (in Marketing) to Marketing Manager Europe & Asia. Do this as follows:

1. Place the cursor where the ampersand is to be inserted.
2. Double-click the entity listed as "amp". This inserts an ampersand (screenshot below).

Note: The Entities Entry Helper is not context-sensitive. All available entities are displayed no matter where the cursor is positioned. This does not mean that an entity can be inserted at all locations in the document. If you are not sure, then validate the document after inserting the entity: XML | Validate (F8).

Defining your own entities

As a document editor, you can define your own document entities. How to do this is described in the section Defining Entities in Authentic View.
2.7 Printing the Document

A printout from Authentic View of an XML document preserves the formatting seen in Authentic View.

To print NanonullOrg.xml, do the following:

1. Switch to Hide Markup mode if you are not already in it. You must do this if you do not want markup to be printed.
2. Select File | Print Preview to see a preview of all pages. Shown below is part of a print preview page, reduced by 50%.

Notice that the formatting of the page is the same as that in Authentic View.

3. To print the file, click File | Print.

Note that you can also print a version of the document that displays markup. To do this, switch Authentic View to Show small markup mode or Show large markup mode, and then print.
3  Authentic View Interface

Authentic View is enabled by clicking the Authentic tab of the active document. If no SPS has been assigned to the XML document, you are prompted to assign one.

This section provides:

- An overview of the interface
- A description of the toolbar icons specific to Authentic View
- A description of viewing modes available in the main Authentic View window
- A description of the Entry Helpers and how they are to be used
- A description of the context menus available at various points in the Authentic View of the XML document

Additional sources of Authentic View information are:

- An Authentic View Tutorial, which shows you how to use the Authentic View interface. This tutorial is available in the documentation of the Altova XMLSpy and Altova Authentic Desktop products (see the Tutorials section), as well as online.
- For a detailed description of Authentic View menu commands, see the User Reference section of your product documentation.

Altova website: XML content editing, XML authoring
3.1 Overview of the GUI

Authentic View has a menu bar and toolbar running across the top of the window, and three areas that cover the rest of the interface: the Project Window, Main Window, and Entry Helpers Window. These areas are shown below.

Menu bar
The menus available in the menu bar are described in detail in the User Reference section of your product documentation.

Toolbar
The symbols and icons displayed in the toolbar are described in the section, Authentic View toolbar icons.

Project window
You can group XML, XSL, XML schema, and Entity files together in a project. To create and modify the list of project files, use the commands in the Project menu (described in the User Reference section of your product documentation). The list of project files is displayed in the Project window. A file in the Project window can be accessed by double-clicking it.

Main window
This is the window in which the XML document is displayed and edited. It is described in the section, Authentic View main window.

Entry helpers
There are three entry helper windows in this area: Elements, Attributes, and Entities. What entries appear in these windows (Elements and Attributes Entry Helpers) are context-sensitive, i.e. it depends on where in the document the cursor is. You can enter an element or entity into the document by double-clicking its entry helper. The value of an attribute is entered into the value
field of that attribute in the Attributes Entry Helper. See the section Authentic View Entry Helpers for details.

**Status Bar**
The Status Bar displays the XPath to the currently selected node.

**Context menus**
These are the menus that appear when you right-click in the Main Window. The available commands are context-sensitive editing commands, i.e. they allow you to manipulate structure and content relevant to the selected node. Such manipulations include inserting, appending, or deleting a node, adding entities, or cutting and pasting content.
3.2 Authentic View Toolbar Icons

Icons in the Authentic View toolbar are command shortcuts. Some icons will already be familiar to you from other Windows applications or Altova products, others might be new to you. This section describes icons unique to Authentic View. In the description below, related icons are grouped together.

**Show/hide XML markup**

In Authentic View, the tags for all, some, or none of the XML elements or attributes can be displayed, either with their names (large markup) or without names (small markup). The four markup icons appear in the toolbar, and the corresponding commands are available in the Authentic menu.

- Hide markup. All XML tags are hidden except those which have been collapsed. Double-clicking on a collapsed tag (which is the usual way to expand it) in Hide markup mode will cause the node's content to be displayed and the tags to be hidden.
- Show small markup. XML element/attribute tags are shown without names.
- Show large markup. XML element/attribute tags are shown with names.
- Show mixed markup. In the StyleVision Power Stylesheet, each XML element or attribute can be specified to display (as either large or small markup), or not to display at all. This is called mixed markup mode since some elements can be specified to be displayed with markup and some without markup. In mixed markup mode, therefore, the Authentic View user sees a customized markup. Note, however, that this customization is created by the person who has designed the StyleVision Power Stylesheet. It cannot be defined by the Authentic View user.

**Editing dynamic table structures**

Rows in a dynamic SPS table are repetitions of a data structure. Each row represents an occurrence of a single element. Each row, therefore, has the same XML substructure as the next.

The dynamic table editing commands manipulate the rows of a dynamic SPS table. That is, you can modify the number and order of the element occurrences. You cannot, however, edit the columns of a dynamic SPS table, since this would entail changing the substructure of individual element occurrences.

The icons for dynamic table editing commands appear in the toolbar, and are also available in the Authentic menu.

- Append row to table
Insert row in table

Duplicate current table row (i.e. cell contents are duplicated)

Move current row up by one row

Move current row down by one row

Delete the current row

Please note: These commands apply only to dynamic SPS tables. They should not be used inside static SPS tables. The various types of tables used in Authentic View are described in the Using Tables in Authentic View section of this documentation.

Creating and editing XML tables
You can insert your own tables should you want to present your data as a table. Such tables are inserted as XML tables. You can modify the structure of an XML table, and format the table. The icons for creating and editing XML tables are available in the toolbar, and are shown below. They are described in the section XML table editing icons.

The commands corresponding to these icons are not available as menu items. Note also that for you to be able to use XML tables, this function must be enabled and suitably configured in the StyleVision Power Stylesheet.

A detailed description of the types of tables used in Authentic View and of how XML tables are to be created and edited is given in Using Tables in Authentic View.

Text formatting icons
Text in Authentic View is formatted by applying to it an XML element or attribute that has the required formatting. If such formatting has been defined, the designer of the StyleVision Power Stylesheet can provide icons in the Authentic View toolbar to apply the formatting. To apply text formatting using a text formatting icon, highlight the text you want to format, and click the appropriate icon.

DB Row Navigation icons
The arrow icons are, from left to right, Go to First Record in the DB; Go to Previous Record; Open Go to Record # dialog; Go to Next Record; and Go to Last Record.

This icon opens the Edit Database Query dialog in which you can enter a query.
Authentic View displays the queried record/s.

XML database editing
The Select New Row with XML Data for Editing command enables you to select a new row from the relevant table in an XML DB, such as IBM DB2. This row appears in Authentic View, can be edited there, and then saved back to the DB.

Portable XML Form (PXF) toolbar buttons
The following PXF toolbar buttons are available in the Authentic View of XMLSpy and Authentic Desktop:

Clicking the individual buttons generates HTML, RTF, PDF, and/or DocX output.

These buttons are enabled when a PXF file is opened in Authentic View. Individual buttons are enabled if the PXF file was configured to contain the XSLT stylesheet for that specific output format. For example, if the PXF file was configured to contain the XSLT stylesheets for HTML and RTF, then only the toolbar buttons for HTML and RTF output will be enabled while those for PDF and DocX (Word 2007+) output will be disabled.
3.3 Authentic View Main Window

There are four viewing modes in Authentic View: Large Markup; Small Markup; Mixed Markup; and Hide All Markup. These modes enable you to view the document with varying levels of markup information. To switch between modes, use the commands in the Authentic menu or the icons in the toolbar (see the previous section, Authentic View toolbar icons).

Large markup

This shows the start and end tags of elements and attributes with the element/attribute names in the tags:

```
<table>
<thead>
<tr>
<th>Department</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the department</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Person</td>
</tr>
</tbody>
</table>
```

The element Name in the figure above is expanded, i.e. the start and end tags, as well as the content of the element, are shown. An element/attribute can be contracted by double-clicking either its start or end tag. To expand the contracted element/attribute, double-click the contracted tag.

```
|Department|Name|Person|
```

In large markup, attributes are recognized by the equals-to symbol in the start and end tags of the attribute:

```
=country|USA|=country
```

Small markup

This shows the start and end tags of elements/attributes without names:
Notice that start tags have a symbol inside it while end tags are empty. Also, element tags have an angular-brackets symbol while attribute tags have and equals sign as its symbol (see screenshot below).

To collapse or expand an element/attribute, double-click the appropriate tag. The example below shows a collapsed element (highlighted in blue). Notice the shape of the tag of the collapsed element and that of the start tag of the expanded element to its left.

**Mixed markup**
Mixed markup shows a customized level of markup. The person who has designed the StyleVision Power Stylesheet can specify either large markup, small markup, or no markup for individual elements/attributes in the document. The Authentic View user sees this customized markup in mixed markup viewing mode.

**Hide all markup**
All XML markup is hidden. Since the formatting seen in Authentic View is the formatting of the printed document, this viewing mode is a WYSIWYG view of the document.
Content display
In Authentic View, content is displayed in two ways:

- Plain text. You type in the text, and this text becomes the content of the element or the value of the attribute.

  ![Plain Text Example]

- Data-entry devices. The display contains either an input field (text box), a multiline input field, combo box, check box, or radio button. In the case of input fields and multiline input fields, the text you enter in the field becomes the XML content of the element or the value of the attribute.

  ![Data-Entry Devices Example]

In the case of the other data-entry devices, your selection produces a corresponding XML value, which is specified in the StyleVision Power Stylesheet. Thus, in a combo box, a selection of, say, "approved" (which would be available in the dropdown list of the combo box) could map to an XML value of "1", or to "approved", or anything else; while "not approved" could map to "0", or "not approved", or anything else.

Optional nodes
When an element or attribute is optional (according to the referenced schema), a prompt of type add [element/attribute] is displayed:

  ![Optional Node Prompt]

Clicking the prompt adds the element, and places the cursor for data entry. If there are multiple optional nodes, the prompt add... is displayed. Clicking the prompt displays a menu of the optional nodes.
3.4 Authentic View Entry Helpers

There are three entry helpers in Authentic View: for Elements, Attributes, and Entities. They are displayed as windows down the right side of the Authentic View interface (see screenshot below).

The Elements and Attributes Entry Helpers are context-sensitive, i.e. what appears in the entry helper depends on where the cursor is in the document. The entities displayed in the Entities Entry Helper are not context-sensitive; all entities allowed for the document are displayed no matter where the cursor is.

Each of the entry helpers is described separately below.

Elements Entry Helper

The Elements Entry Helper consists of two parts:

- The upper part, containing an XML tree that can be toggled on and off using the Show XML tree check box. The XML tree shows the ancestors up to the document’s root element for the current element. When you click on an element in the XML tree, elements
corresponding to that element (as described in the next item in this list) appear in the lower part of the Elements Entry Helper.

- The lower part, containing a list of the nodes that can be inserted within, before, and after; applied to or cleared from the selected element or text range in Authentic View. What you can do with an element listed in the Entry Helper is indicated by the icon to the left of the element name in the Entry Helper. The icons that occur in the Elements Entry Helper are listed below, together with an explanation of what they mean.

To use a node from the Entry Helper, click its icon.

- **Insert After Element**
  The element in the Entry Helper is inserted after the selected element. Note that it is appended at the correct hierarchical level. For example, if your cursor is inside a `//sect1/para` element, and you append a `sect1` element, then the new `sect1` element will be appended not as a following sibling of `//sect1/para` but as a following sibling of the `sect1` element that is the parent of that `para` element.

- **Insert Before Element**
  The element in the Entry Helper is inserted before the selected element. Note that, just as with the Insert After Element command, the element is inserted at the correct hierarchical level.

- **Remove Element**
  Removes the element and its content.

- **Insert Element**
  An element from the Entry Helper can also be inserted within an element. When the cursor is placed within an element, then the allowed child elements of that element can be inserted. Note that allowed child elements can be part of an elements-only content model as well as a mixed content model (text plus child elements).

  An allowed child element can be inserted either when a text range is selected or when the cursor is placed as an insertion point within the text.

  - When a text range is selected and an element inserted, the text range becomes the content of the inserted element.
  - When an element is inserted at an insertion point, the element is inserted at that point.

  After an element has been inserted, it can be cleared by clicking either of the two Clear Element icons that appear (in the Elements Entry Helper) for these inline elements. Which of the two icons appears depends on whether you select a text range or place the cursor in the text as an insertion point (see below).

- **Apply Element**
  If you select an element in your document (by clicking either its start or end tag in the Show large markup view) and that element can be replaced by another element (for example, in a mixed content element such as `para`, an `italic` element can be replaced by the `bold` element), this icon indicates that the
element in the Entry Helper can be applied to the selected (original) element. The **Apply Element** command can also be applied to a text range within an element of mixed content; the text range will be created as content of the applied element.

- If the applied element has a **child element with the same name** as a child of the original element and an instance of this child element exists in the original element, then the child element of the original is retained in the new element's content.
- If the applied element has **no child element with the same name** as that of an instantiated child of the original element, then the instantiated child of the original element is appended as a sibling of any child element or elements that the new element may have.
- If the applied element has a **child element for which no equivalent exists** in the original element's content model, then this child element is not created directly but Authentic View offers you the option of inserting it.

If a text range is selected rather than an element, applying an element to the selection will create the applied element at that location with the selected text range as its content. Applying an element when the cursor is an insertion point is not allowed.

**Clear Element (when range selected)**
This icon appears when text within an element of mixed content is selected. Clicking the icon clears the element from around the selected text range.

**Clear Element (when insertion point selected)**
This icon appears when the cursor is placed within an element that is a child of a mixed-content element. Clicking the icon clears the inline element.

**Attributes Entry Helper**
The Attributes Entry Helper consists of a drop-down combo box and a list of attributes. The element that you have selected (you can click the start or end tag, or place the cursor anywhere in the element content to select it) appears in the combo box.

The Attributes Entry Helper shown in the figures below has a **para** element in the combo box. Clicking the arrow in the combo box drops down a list of all the **para** element's ancestors up to the document's root element, which in this case is **OrgChart**.

![Attribute Entry Helper](image)

Below the combo box, a list of valid attributes for that element is displayed, in this case for **para**.
If an attribute is mandatory on a given element, then it appears in bold. (In the example below, there are no mandatory attributes except the built-in attribute `xsi:type`.)

To enter a value for an attribute, click in the value field of the attribute and enter the value. This creates the attribute and its value in the XML document.

In the case of the `xsi:nil` attribute, which appears in the Attributes Entry Helper when a nillable element has been selected, the value of the `xsi:nil` attribute can only be entered by selecting one of the allowed values (`true` or `false`) from the dropdown list for the attribute’s value.

The `xsi:type` attribute can be changed by clicking in the value field of the attribute and then selecting, from the dropdown list that appears, one of the listed values. The listed values are the available abstract types defined in the XML Schema on which the Authentic View document is based.

**Entities Entry Helper**

The Entities Entry Helper allows you to insert an entity in your document. Entities can be used to insert special characters or text fragments that occur often in a document (such as the name of a company). To insert an entity, place the cursor at the point in the text where you want to have the entity inserted, then double-click the entity in the Entities Entry Helper.

**Note:** An internal entity is one that has its value defined within the DTD. An external entity is one that has its value contained in an external source, e.g. another XML file. Both internal and external entities are listed in the Entities Entry Helper. When you insert an entity, whether internal or external, the entity—not its value—is inserted into the XML text. If the entity is an internal entity, Authentic View displays the value of the entity. If the entity is an external entity, Authentic View displays the entity—and not its value. This means, for example, that an XML file that is an external entity will be shown in the Authentic View display as an entity; its content does not replace the entity in the Authentic View display.

You can also define your own entities in Authentic View and these will also be displayed in the entry helper: see Define Entities in the Editing in Authentic View section.
3.5 Authentic View Context Menus

Right-clicking on some selected document content or node pops up a context menu with commands relevant to the selection or cursor location.

Inserting elements

The figure below shows the Insert submenu, which is a list of all elements that can be inserted at that current cursor location. The Insert Before submenu lists all elements that can be inserted before the current element. The Insert After submenu lists all elements that can be inserted after the current element. In the figure below, the current element is the para element. The bold and italic elements can be inserted within the current para element.

As can be seen below, the para and Office elements can be inserted before the current para element.

The node insertion, replacement (Apply), and markup removal (Clear) commands that are available in the context menu are also available in the Authentic View entry helpers and are fully described in that section.

Insert entity

Positioning the cursor over the Insert Entity command rolls out a submenu containing a list of all declared entities. Clicking an entity inserts it at the selection. See Define Entities for a description of how to define entities for the document.

Insert CDATA Section

This command is enabled when the cursor is placed within text. Clicking it inserts a CDATA section at the cursor insertion point. The CDATA section is delimited by start and end tags; to see these tags you should switch on large or small markup. Within CDATA sections, XML markup and parsing is ignored. XML markup characters (the ampersand, apostrophe, greater than, less than, and quote characters) are not treated as markup, but as literals. So CDATA sections are useful for text such as program code listings, which have XML markup characters.

Remove node

Positioning the mouse cursor over the Remove command pops up a menu list consisting of the selected node and all its removable ancestors (those that would not invalidate the document) up to the document element. Click the element to be removed. This is a quick way to delete an element or any removable ancestor. Note that clicking an ancestor element will remove all its descendants, including the selected element.
Clear
The **Clear** command clears the element markup from around the selection. If the entire node is selected, then the element markup is cleared for the entire node. If a text segment is selected, then the element markup is cleared from around that text segment only.

Apply
The **Apply** command applies a selected element to your selection in the main Window. For more details, see Authentic View entry helpers.

Copy, Cut, Paste
These are the standard Windows commands. Note, however, that the **Paste** command pastes copied text either as XML or as Text, depending on what the designer of the stylesheet has specified for the SPS as a whole. For information about how the **Copy as XML** and **Copy as Text** commands work, see the description of the **Paste As** command immediately below.

Paste As
The **Paste As** command offers the option of pasting as XML or as text an Authentic View XML fragment (which was copied to the clipboard). If the copied fragment is pasted as XML it is pasted together with its XML markup. If it is pasted as text, then only the text content of the copied fragment is pasted (not the XML markup, if any). The following situations are possible:

- **An entire node together with its markup tags** is highlighted in Authentic View and copied to the clipboard. (i) The node can be pasted as XML to any location where this node may validly be placed. It will not be pasted to an invalid location. (ii) If the node is pasted as text, then only the node's text content will be pasted (not the markup); the text content can be pasted to any location in the XML document where text may be pasted.

- **A text fragment** is highlighted in Authentic View and copied to the clipboard. (i) If this fragment is pasted as XML, then the XML markup tags of the text—even though these were not explicitly copied with the text fragment—will be pasted along with the text, but only if the XML node is valid at the location where the fragment is pasted. (ii) If the fragment is pasted as text, then it can be pasted to any location in the XML document where text may be pasted.

**Note:** Text will be copied to nodes where text is allowed, so it is up to you to ensure that the copied text does not invalidate the document. The copied text should therefore be:

(i) lexically valid in the new location (for example, non-numeric characters in a numeric node would be invalid), and
(ii) not otherwise invalidate the node (for example, four digits in a node that accepts only three-digit numbers would invalidate the node).

If the pasted text does in any way invalidate the document, this will be indicated by the text being displayed in red.

Delete
The **Delete** command removes the selected node and its contents. A node is considered to be selected for this purpose by placing the cursor within the the node or by clicking either the start or
end tag of the node.
4 Editing in Authentic View

This section describes important features of Authentic View in detail. Features have been included in this section either because they are frequently used or because the mechanisms or concepts involved require explanation.

The section explains the following:

- There are three distinct types of tables used in Authentic View. The section Using tables in Authentic View explains the three types of tables (static SPS, dynamic SPS, and XML), and when and how to use them. It starts with the broad, conceptual picture and moves to the details of usage.
- The Date Picker is a graphical calendar that enters dates in the correct XML format when you click a date. See Date Picker.
- An entity is shorthand for a special character or text string. You can define your own entities, which allows you to insert these special characters or text strings by inserting the corresponding entities. See Defining Entities for details.
- In the Enterprise and Professional editions of Altova products, Authentic View users can sign XML documents with digital XML signatures and verify these signatures.
- What image formats can be displayed in Authentic View.

Altova website: XML content editing, XML authoring
4.1 Basic Editing

When you edit in Authentic View, you are editing an XML document. Authentic View, however, can hide the structural XML markup of the document, thus displaying only the content of the document (first screenshot below). You are therefore not exposed to the technicalities of XML, and can edit the document as you would a normal text document. If you wish, you could switch on the markup at any time while editing (second screenshot below).

Inserting nodes

Very often you will need to add a new node to the Authentic XML document. For example, a new Person element might need to be added to an address book type of document. In such cases the XML Schema would allow the addition of the new element. All you need to do is right-click the node in the Authentic View document before which or after which you wish to add the new node. In the context menu that appears, select Insert Before or Insert After as required. The nodes available for insertion at that point in the document are listed in a submenu. Click the required
node to insert it. The node will be inserted. All mandatory descendant nodes are also inserted. If a descendant node is optional, a clickable link, **Add NodeName**, appears to enable you to add the optional node if you wish to.

If the node being added is an element with an abstract type, then a dialog (**something like in the screenshot below**) appears containing a list of derived types that are available in the XML Schema.

The screenshot above pops up when a Publication element is added. The Publication element is of type PublicationType, which is an abstract complex type. The two complex types BookType and MagazineType are derived from the abstract PublicationType. Therefore, when a Publication element is added to the XML document, one of these two concrete types derived from Publication's abstract type must be specified. The new Publication element will be added with an xsi:type attribute:

```xml
<Publication xsi:type="BookType"> ... </Publication>
<Publication xsi:type="MagazineType"> ... </Publication>
...
<Publication xsi:type="MagazineType"> ... </Publication>
```

Selecting one of the available derived types and clicking **OK** does the following:

- Sets the selected derived type as the value of the xsi:type attribute of the element
- Inserts the element together with the descendant nodes defined in the content model of the selected derived type.

The selected derived type can be changed subsequently by changing the value of the element’s xsi:type attribute in the Attributes Entry Helper. When the element’s type is changed in this way, all nodes of the previous type’s content model are removed and nodes of the new type’s content model are inserted.
Text editing
An Authentic View document will essentially consist of text and images. To edit the text in the
document, place the cursor at the location where you wish to insert text, and type. You can copy,
move, and delete text using familiar keystrokes (such as the Delete key) and drag-and-drop
mechanisms. One exception is the Enter key. Since the Authentic View document is pre-
formatted, you do not—and cannot—add extra lines or space between items. The Enter key in
Authentic View therefore serves to append another instance of the element currently being edited,
and should be used exclusively for this purpose.

Copy as XML or as text
Text can be copied and pasted as XML or as text.

- If text is pasted as XML, then the XML markup is pasted together with the text content of
  nodes. The XML markup is pasted even if only part of a node's contents has been copied.
  For the markup to be pasted it must be allowed, according to the schema, at the location
  where it is pasted.
- If text is pasted as text, XML markup is not pasted.

To paste as XML or text, first copy the text (Ctrl+C), right-click at the location where the text is to
be pasted, and select the context menu command Paste As | XML or Paste As | Text. If the
shortcut Ctrl+V is used, the text will be pasted in the default Paste Mode of the SPS. The default
Paste Mode will have been specified by the designer of the SPS. For more details, see the
section Context Menus.

Alternatively, highlighted text can be dragged to the location where it is to be pasted. When the
text is dropped, a pop-up appears asking whether the text is to be pasted as text or XML. Select
the desired option.

Text formatting
A fundamental principle of XML document systems is that content be kept separate from
presentation. The XML document contains the content, while the stylesheet contains the
presentation (formatting). In Authentic View, the XML document is presented via the stylesheet.
This means that all the formatting you see in Authentic View is produced by the stylesheet. If you
see bold text, that bold formatting has been provided by the stylesheet. If you see a list or a table,
that list format or table format has been provided by the stylesheet. The XML document, which
you edit in Authentic View contains only the content; it contains no formatting whatsoever. The
formatting is contained in the stylesheet. What this means for you, the Authentic View user, is
that you do not have to—nor can you—format any of the text you edit. You are editing content.
The formatting that is automatically applied to the content you edit is linked to the semantic and/
or structural value of the data you are editing. For example, an email address (which could be
considered a semantic unit) will be formatted automatically in a certain way because it is an
email. In the same way, a headline must occur at a particular location in the document (both a
structural and semantic unit) and will be formatted automatically in the way the stylesheet
designer has specified that headlines be formatted. You cannot change the formatting of either
email address or headline. All that you do is edit the content of the email address or headline.

In some cases, content might need to be specially presented; for example, a text string that must
be presented in boldface. In all such cases, the presentation must be tied in with a structural
element of the document. For example, a text string that must be presented in boldface, will be
structurally separated from surrounding content by markup that the stylesheet designer will format in boldface. If you, as the Authentic View user, need to use such a text string, you would need to enclose the text string within the appropriate element markup. For information about how to do this, see the Insert Element command in the Elements Entry Helper section of the documentation.

**Using RichEdit in Authentic View**

In Authentic View, when the cursor is placed inside an element that has been created as a RichEdit component, the buttons and controls in the RichEdit toolbar (screenshot below) become enabled. Otherwise they are grayed out.

![RichEdit toolbar screenshot](image)

Select the text you wish to style and specify the styling you wish to apply via the buttons and controls of the RichEdit toolbar. RichEdit enables the Authentic View user to specify the font, font-weight, font-style, font-decoration, font-size, color, background color and alignment of text. The text that has been styled will be enclosed in the tags of the styling element.

**Inserting entities**

In XML documents, some characters are reserved for markup and cannot be used in normal text. These are the ampersand (&), apostrophe (‘), less than (<), greater than (>), and quote (“) characters. If you wish to use these characters in your data, you must insert them as entity references, via the Entities Entry Helper (screenshot below).

![Entities Entry Helper screenshot](image)

XML also offers the opportunity to create custom entities. These could be: (i) special characters that are not available on your keyboard, (ii) text strings that you wish to re-use in your document content, (iii) XML data fragments, or (iv) other resources, such as images. You can define your own entities within the Authentic View application. Once defined, these entities appear in the Entities Entry Helper and can then be inserted as in the document.

**Inserting CDATA sections**

CDATA sections are sections of text in an XML document that the XML parser does not process as XML data. They can be used to escape large sections of text if replacing special characters by entity references is undesirable; this could be the case, for example, with program code or an XML fragment that is to be reproduced with its markup tags. CDATA sections can occur within element content and are delimited by `<![CDATA[` and `]]>`. Consequently the text string `]]>` should not occur within a CDATA section as it would prematurely signify the end of the section. In this case, the greater than character should be escaped by its entity reference (`&gt;`). To insert a CDATA section within an element, place the
cursor at the desired location, right-click, and select **Insert CDATA Section** from the context menu. To see the CDATA section tags in Authentic View, switch on the markup display. Alternatively, you could highlight the text that is to be enclosed in a CDATA section, and then select the **Insert CDATA section** command.

**Note:** CDATA sections cannot be inserted into input fields (that is, in text boxes and multiline text boxes). CDATA sections can only be entered within elements that are displayed in Authentic View as text content components.

**Editing and following links**
A hyperlink consists of two parts: the link text and the target of the link. You can edit the link text by clicking in the text and editing. But you cannot edit the target of the link. (The target of the link is set by the designer of the stylesheet (either by typing in a static target address or by deriving the target address from data contained in the XML document).) From Authentic View, you can go to the target of the link by pressing **Ctrl** and clicking the link text. (Remember: merely clicking the link will set you up for editing the link text.)
4.2 Tables in Authentic View

The three table types fall into two categories: SPS tables (static and dynamic) and CALS/HTML Tables.

SPS tables are of two types: static and dynamic. SPS tables are designed by the designer of the StyleVision Power Stylesheet to which your XML document is linked. You yourself cannot insert an SPS table into the XML document, but you can enter data into SPS table fields and add and delete the rows of dynamic SPS tables. The section on SPS tables below explains the features of these tables.

CALS/HTML tables are inserted by you, the user of Authentic View. Their purpose is to enable you to insert tables at any allowed location in the document hierarchy should you wish to do so. The editing features of CALS/HTML Tables and the CALS/HTML Table editing icons are described below.
4.2.1 SPS Tables

Two types of SPS tables are used in Authentic View: static tables and dynamic tables.

Static tables are fixed in their structure and in the content-type of cells. You, as the user of Authentic View, can enter data into the table cells but you cannot change the structure of these tables (i.e. add rows or columns, etc) or change the content-type of a cell. You enter data either by typing in text, or by selecting from options presented in the form of check-box or radio button alternatives or as a list in a combo-box. After you enter data, you can edit it.

Please note: The icons or commands for editing dynamic tables must not be used to edit static tables.

Dynamic tables have rows that represent a repeating data structure, i.e. each row has an identical data structure (not the case with static tables). Therefore, you can perform row operations: append row, insert row, move row up, move row down, and delete row. These commands are available under the Authentic menu and as icons in the toolbar (shown below).

To use these commands, place the cursor anywhere in the appropriate row, and then select the required command.

<table>
<thead>
<tr>
<th>Nanonull, Inc.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Street:</strong> 119 Oakstreet, Suite 4876</td>
<td><strong>Phone:</strong> +1 (321) 555 5155</td>
</tr>
<tr>
<td><strong>City:</strong> Vereno</td>
<td><strong>Fax:</strong> +1 (321) 555 5155 - 9</td>
</tr>
<tr>
<td><strong>State &amp; Zip:</strong> DC 29213</td>
<td><strong>E-mail:</strong> <a href="mailto:office@nanonull.com">office@nanonull.com</a></td>
</tr>
</tbody>
</table>

Please note: The icons or commands for editing dynamic tables must not be used to edit static tables.

<table>
<thead>
<tr>
<th>Administration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First</strong></td>
<td><strong>Last</strong></td>
</tr>
<tr>
<td>Vernon</td>
<td>Callaby</td>
</tr>
<tr>
<td>Frank</td>
<td>Further</td>
</tr>
<tr>
<td>Loby</td>
<td>Matise</td>
</tr>
<tr>
<td><strong>Employees: 3 (20% of Office, 9% of Company)</strong></td>
<td><strong>Shares: 1500 (13% of Office, 6% of Company)</strong></td>
</tr>
<tr>
<td>Non-Shareholders: Frank Further, Loby Matise.</td>
<td></td>
</tr>
</tbody>
</table>
To move among cells in the table, use the Up, Down, Left, and Right arrow keys. To move forward from one cell to the next, use the Tab key. Pressing the Tab key in the last cell of the last row creates a new row.
4.2.2 CALS/HTML Tables

CALS/HTML tables can be inserted by you, the user of Authentic View, for certain XML data structures that have been specified to show a table format. There are three steps involved when working with CALS/HTML tables: inserting the table; formatting it; and entering data. The commands for working with CALS/HTML tables are available as icons in the toolbar (see CALS/HTML table editing icons).

Inserting tables

To insert a CALS/HTML table do the following:

1. Place your cursor where you wish to insert the table, and click the icon. (Note that where you can insert tables is determined by the schema.) The Insert Table dialog appears. This dialog lists all the XML element data-structures for which a table structure has been defined. For example, in the screenshot below, the informaltable element and table element have each been defined as both a CALS table as well as an HTML table.

2. Select the entry containing the element and table model you wish to insert, and click OK.
3. In the next dialog, select the number of columns and rows, and specify whether a header and/or footer is to be added to the table and whether the table is to extend over the entire available width. Click OK when done.
For the specifications given in the dialog box shown above, the following table is created.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

By using the **Table** menu commands, you can add and delete columns, and create row and column joins and splits. But to start with, you must create the broad structure.

**Formatting tables and entering data**

The table formatting will already have been assigned in the document design. However, you might, under certain circumstances, be able to modify the table formatting. These circumstances are as follows:

- The elements corresponding to the various table structure elements must have the relevant CALS or HTML table properties defined as attributes (in the underlying XML Schema). Only those attributes that are defined will be available for formatting. If, in the design, values have been set for these attributes, then you can override these values in Authentic View.
- In the design, no **style** attribute containing CSS styles must have been set. If a style attribute containing CSS styles has been specified for an element, the **style** attribute has precedence over any other formatting attribute set on that element. As a result, any formatting specified in Authentic View will be overridden.

To format a table, row, column, or cell, do the following:

1. Place the cursor anywhere in the table and click the **(Table Properties)** icon. This opens the Table Properties dialog (see **screenshot**), where you specify formatting for the table, or for a row, column, or cell.
2. Set the cellspacing and cellpadding properties to "0". Your table will now look like this:

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

3. Place the cursor in the first row to format it, and click the (Table Properties) icon. Click the Row tab.

![Table Properties](image)

Since the first row will be the header row, set a background color to differentiate this row from the other rows. Note the Row properties that have been set in the figure above. Then enter the column header text. Your table will now look like this:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Telephone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Notice that the alignment is centered as specified.

4. Now, say you want to divide the "Telephone" column into the sub-columns "Office" and "Home", in which case you would need to split the horizontal width of the Telephone column into two columns. First, however, we will split the vertical extent of the header cell to make a sub-header row. Place the cursor in the "Telephone" cell, and click the (Split vertically) icon. Your table will look like this:
5. Now place the cursor in the cell below the cell containing "Telephone", and click the (Split horizontally) icon. Then type in the column headers "Office" and "Home". Your table will now look like this:

<table>
<thead>
<tr>
<th>Name</th>
<th>Telephone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now you will have to split the horizontal width of each cell in the "Telephone" column.

You can also add and delete columns and rows, and vertically align cell content, using the table-editing icons. The CALS/HTML table editing icons are described in the section titled, CALS/HTML Table Editing Icons.

**Moving among cells in the table**
To move among cells in the CALS/HTML table, use the Up, Down, Right, and Left arrow keys.

**Entering data in a cell**
To enter data in a cell, place the cursor in the cell, and type in the data.

**Formatting text**
Text in a CALS/HTML table, as with other text in the XML document, must be formatted using XML elements or attributes. To add an element, highlight the text and double-click the required element in the Elements Entry Helper. To specify an attribute value, place the cursor within the text fragment and enter the required attribute value in the Attributes Entry Helper. After formatting the header text bold, your table will look like this.

<table>
<thead>
<tr>
<th>Name</th>
<th>Telephone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The text above was formatted by highlighting the text, and double-clicking the element **strong**, for which a global template exists that specifies bold as the font-weight. The text formatting becomes immediately visible.

**Please note:** For text formatting to be displayed in Authentic View, a global template with the required text formatting must have been created in StyleVision for the element in question.
4.2.3 CALS/HTML Table Editing Icons

The commands required to edit CALS/HTML tables are available as icons in the toolbar, and are listed below. Note that no corresponding menu commands exist for these icons.

For a full description of when and how CALS/HTML Tables are to be used, see CALS/HTML Tables.

Insert table

The "Insert Table" command inserts a CALS/HTML table at the current cursor position.

Delete table

The "Delete table" command deletes the currently active table.

Append row

The "Append row" command appends a row to the end of the currently active table.

Append column

The "Append column" command appends a column to the end of the currently active table.

Insert row

The "Insert row" command inserts a row above the current cursor position in the currently active table.

Insert column

The "Insert column" command inserts a column to the left of the current cursor position in the currently active table.

Join cell left

The "Join cell left" command joins the current cell (current cursor position) with the cell to the left. The tags of both cells remain in the new cell, the column headers remain unchanged and are concatenated.
Join cell right

The "Join cell right" command joins the current cell (current cursor position) with the cell to the right. The contents of both cells are concatenated in the new cell.

Join cell below

The "Join cell below" command joins the current cell (current cursor position) with the cell below. The contents of both cells are concatenated in the new cell.

Join cell above

The "Join cell above" command joins the current cell (current cursor position) with the cell above. The contents of both cells are concatenated in the new cell.

Split cell horizontally

The "Split cell Horizontally" command creates a new cell to the right of the currently active cell. The size of both cells, is now the same as the original cell.

Split cell vertically

The "Split cell Vertically" command creates a new cell below the currently active cell.

Align top

This command aligns the cell contents to the top of the cell.

Center vertically

This command centers the cell contents.

Align bottom

This command aligns the cell contents to the bottom of the cell.
Table properties

The "Table properties" command opens the Table Properties dialog box. This icon is only made active for HTML tables, it cannot be clicked for CALS tables.
4.3 Editing a DB

In Authentic View, you can edit database (DB) tables and save data back to a DB. This section contains a full description of interface features available to you when editing a DB table. The following general points need to be noted:

- The number of records in a DB table that are displayed in Authentic View may have been deliberately restricted by the designer of the StyleVision Power Stylesheet in order to make the design more compact. In such cases, only that limited number of records is initially loaded into Authentic View. Using the DB table row navigation icons (see Navigating a DB Table), you can load and display the other records in the DB table.
- You can query the DB to display certain records.
- You can add, modify, and delete DB records, and save your changes back to the DB. See Modifying a DB Table.

To open a DB-based StyleVision Power Stylesheet in Authentic View:
- Click Authentic | Edit Database Data, and browse for the required StyleVision Power Stylesheet.

Note: In Authentic View, data coming from a SQLite database is not editable. When you attempt to save SQLite data from the Authentic view, a message box will inform you of this known limitation.
4.3.1 **Navigating a DB Table**

The commands to navigate DB table rows are available as buttons in the Authentic View document. Typically, one navigation panel with either four or five buttons accompanies each DB table.

![Navigation icons](image)

The arrow icons are, from left to right, Go to First Record in the DB Table; Go to Previous Record; Open the Go to Record dialog (see screenshot); Go to Next Record; and Go to Last Record.

![Go To Record dialog](image)

To navigate a DB table, click the required button.

**XML Databases**

In the case of XML DBs, such as IBM DB2, one cell (or row) contains a single XML document, and therefore a single row is loaded into Authentic View at a time. To load an XML document that is in another row, use the `Authentic | Select New Row with XML Data for Editing` menu command.
4.3.2 DB Queries

A DB query enables you to query the records of a table displayed in Authentic View. A query is made for an individual table, and only one query can be made for each table. You can make a query at any time while editing. If you have unsaved changes in your Authentic View document at the time you submit the query, you will be prompted about whether you wish to save all changes made in the document or discard all changes. Note that even changes made in other tables will be saved/discard. After you submit the query, the table is reloaded using the query conditions.

Please note: If you get a message saying that too many tables are open, then you can reduce the number of tables that are open by using a query to filter out some tables.

To create and submit a query:

1. Click the Query button for the required table in order to open the Edit Database Query dialog (see screenshot). This button typically appears at the top of each DB table or below it. If a Query button is not present for any table, the designer of the StyleVision Power Stylesheet has not enabled the DB Query feature for that table.

2. Click the Append AND or Append OR button. This appends an empty criterion for the query (shown below).

![Edit Database Query screenshot](image)
4. Enter the expression for the criterion. An expression consists of: (i) a field name (available from the associated combo-box); (ii) an operator (available from the associated combo-box); and (iii) a value (to be entered directly). For details of how to construct expressions see the Expressions in criteria section.

5. If you wish to add another criterion, click the Append AND or Append OR button according to which logical operator (AND or OR) you wish to use to join the two criteria. Then add the new criterion. For details about the logical operators, see the section Re-ordering criteria in DB Queries.

Expressions in criteria
Expressions in DB Query criteria consist of a field name, an operator, and a value. The available field names are the child elements of the selected top-level data table; the names of these fields are listed in a combo-box (see screenshot above). The operators you can use are listed below:

- `=` Equal to
- `<>` Not equal to
- `<` Less than
- `<=` Less than or equal to
- `>` Greater than
- `>=` Greater than or equal to
- `LIKE` Phonetically alike
- `NOT LIKE` Phonetically not alike
- `IS NULL` Is empty
- `NOT NULL` Is not empty

If IS NULL or NOT NULL is selected, the Value field is disabled. Values must be entered without quotes (or any other delimiter). Values must also have the same formatting as that of the corresponding DB field; otherwise the expression will evaluate to FALSE. For example, if a criterion for a field of the date datatype in an MS Access DB has an expression StartDate=25/05/2004, the expression will evaluate to FALSE because the date datatype in
an MS Access DB has a format of YYYY-MM-DD.

Using parameters with DB Queries

You can enter the name of a parameter as the value of an expression when creating queries. Parameters are variables that can be used instead of literal values in queries. When you enter it in an expression, its value is used in the expression. Parameters that are available have been defined by the SPS designer in the SPS and can be viewed in the View Parameters dialog (see screenshot below). Parameters have been assigned a default value in the SPS, which can be overridden by passing a value to the parameter via the command line (if and when the output document is compiled via the command line).

To view the parameters defined for the SPS, click the Parameters button in the Edit Database Query dialog. This opens the View Parameters dialog (see screenshot).

The View Parameters dialog contains all the parameters that have been defined for the stylesheet in the SPS and parameters must be edited in the stylesheet design.

Re-ordering criteria in DB Queries

The logical structure of the DB Query and the relationship between any two criteria or sets of criteria is indicated graphically. Each level of the logical structure is indicated by a square bracket. Two adjacent criteria or sets of criteria indicate the AND operator, whereas if two criteria are separated by the word OR then the OR operator is indicated. The criteria are also appropriately indented to provide a clear overview of the logical structure of the DB Query.
The DB Query shown in the screenshot above may be represented in text as:

\[
\text{State}=\text{CA} \text{ AND (City}=\text{Los Angeles OR City}=\text{San Diego OR (City}=\text{San Francisco AND CustomerNr}=25))
\]

You can re-order the DB Query by moving a criterion or set of criteria up or down relative to the other criteria in the DB Query. To move a criterion or set of criteria, do the following:

1. Select the criterion by clicking on it, or select an entire level by clicking on the bracket that represents that level.
2. Click the Up or Down arrow button in the dialog.

The following points should be noted:

- If the adjacent criterion in the direction of movement is at the same level, the two criteria exchange places.
- A set of criteria (i.e. criterion within a bracket) changes position within the same level; it does not change levels.
- An individual criterion changes position within the same level. If the adjacent criterion is further outward/inward (i.e. not on the same level), then the selected criterion will move outward/inward, **one level at a time**.

To delete a criterion in a DB Query, select the criterion and click **Delete**.

**Modifying a DB Query**

To modify a DB Query:

1. Click the Query button \[\text{Query} \]. The Edit Database Query dialog box opens. You can now edit the expressions in any of the listed criteria, add new criteria, re-order criteria, or delete criteria in the DB Query.
2. Click **OK**. The data from the DB is automatically re-loaded into Authentic View so as to reflect the modifications to the DB Query.
4.3.3 Modifying a DB Table

Adding a record
To add a record to a DB table:

1. Place the cursor in the DB table row and click the icon (to append a row) or the icon (to insert a row). This creates a new record in the temporary XML file.
2. Click the File | Save command to add the new record in the DB. In Authentic View a row for the new record is appended to the DB table display. The AltovaRowStatus for this record is set to A (for Added).

When you enter data for the new record it is entered in bold and is underlined. This enables you to differentiate added records from existing records—if existing records have not been formatted with these text formatting properties. Datatype errors are flagged by being displayed in red.

The new record is added to the DB when you click File | Save. After a new record is saved to the DB, its AltovaRowStatus field is initialized (indicated with ---) and the record is displayed in Authentic View as a regular record.

Modifying a record
To modify a record, place the cursor at the required point in the DB table and edit the record as required. If the number of displayed records is limited, you may need to navigate to the required record (see Navigating a DB Table).

When you modify a record, entries in all fields of the record are underlined and the AltovaRowStatus of all primary instances of this record is set to U (for Updated). All secondary instances of this record have their AltovaRowStatus set to u (lowercase). Primary and secondary instances of a record are defined by the structure of the DB—and correspondingly of the XML Schema generated from it. For example, if an Address table is included in a Customer table, then the Address table can occur in the Design Document in two types of instantiations: as the Address table itself and within instantiations of the Customer table. Whichever of these two types is modified is the type that has been primarily modified. Other types—there may be more than one other type—are secondary types. Datatype errors are flagged by being displayed in red.

The modifications are saved to the DB by clicking File | Save. After a modified record is saved to the DB, its AltovaRowStatus field is initialized (indicated with ---) and the record is displayed in Authentic View as a regular record.

Please note:

- If even a single field of a record is modified in Authentic View, the entire record is updated when the data is saved to the DB.
- The date value 0001-01-01 is defined as a NULL value for some DBs, and could result in an error message.

Deleting a record

To delete a record:

1. Place the cursor in the row representing the record to be deleted and click the icon. The record to be deleted is marked with a strikethrough. The AltovaRowStatus is set as follows: primary instances of the record are set to D; secondary instances to d; and
records indirectly deleted to x. Indirectly deleted records are fields in the deleted record that are held in a separate table. For example, an Address table might be included in a Customer table. If a Customer record were to be deleted, then its corresponding Address record would be indirectly deleted. If an Address record in the Customer table were deleted, then the Address record in the Customer table would be primarily deleted, but the same record would be secondarily deleted in an independent Address table if this were instantiated.

2. Click **File | Save** to save the modifications to the DB.

**Please note:** Saving data to the DB resets the Undo command, so you cannot undo actions that were carried out prior to the save.
4.4 Working with Dates

There are two ways in which dates can be edited in Authentic View:

- Dates are entered or modified using the Date Picker.
- Dates are entered or modified by typing in the value.

The method the Authentic View user will use is defined in the SPS. Both methods are described in the two sub-sections of this section.

Note on date formats

In the XML document, dates can be stored in one of several date datatypes. Each of these datatypes requires that the date be stored in a particular lexical format in order for the XML document to be valid. For example, the xs:date datatype requires a lexical format of YYYY-MM-DD. If the date in an xs:date node is entered in anything other than this format, then the XML document will be invalid.

In order to ensure that the date is entered in the correct format, the SPS designer can include the graphical Date Picker in the design. This would ensure that the date selected in the Date Picker is entered in the correct lexical format. If there is no Date Picker, the Authentic View should take care to enter the date in the correct lexical format. Validating the XML document could provide useful tips about the required lexical format.
4.4.1 Date Picker

The Date Picker is a graphical calendar used to enter dates in a standard format into the XML document. Having a standard format is important for the processing of data in the document. The Date Picker icon appears near the date field it modifies (see screenshot).

To display the Date Picker (see screenshot), click the Date Picker icon.

To select a date, click on the desired date, month, or year. The date is entered in the XML document, and the date in the display is modified accordingly. You can also enter a time zone if this is required.
4.4.2 Text Entry

For date fields that do not have a Date Picker (see screenshot), you can edit the date directly by typing in the new value.

Please note: When editing a date, you must not change its format.

If you edit a date and change it such that it is out of the valid range for dates, the date turns red to alert you to the error. If you place the mouse cursor over the invalid date, an error message appears (see screenshot).

If you try to change the format of the date, the date turns red to alert you to the error (see screenshot).
4.5 Defining Entities

You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the Insert Entity submenu of the context menu. When you double-click on an entity in the Entities Entry Helper, that entity is inserted at the cursor insertion point.

An entity is useful if you will be using a text string, XML fragment, or some other external resource in multiple locations in your document. You define the entity, which is basically a short name that stands in for the required data, in the Define Entities dialog. After defining an entity you can use it at multiple locations in your document. This helps you save time and greatly enhances maintenance.

There are two broad types of entities you can use in your document: a parsed entity, which is XML data (either a text string or a fragment of an XML document), or an unparsed entity, which is non-XML data such as a binary file (usually a graphic, sound, or multimedia object). Each entity has a name and a value. In the case of parsed entities the entity is a placeholder for the XML data. The value of the entity is either the XML data itself or a URI that points to a .xml file that contains the XML data. In the case of unparsed entities, the value of the entity is a URI that points to the non-XML data file.

To define an entity:

1. Click Authentic | Define XML Entities... This opens the Define Entities dialog (screenshot below).

2. Enter the name of your entity in the Name field. This is the name that will appear in the Entities Entry Helper.
3. Enter the type of entity from the drop-down list in the Type field. The following types are possible: An Internal entity is one for which the text to be used is stored in the XML document itself. Selecting Public or System specifies that the resource is located outside the XML file, and will be located with the use of a public identifier or a system identifier, respectively. A system identifier is a URI that gives the location of the resource. A public identifier is a location-independent identifier, which enables some processors to identify the resource. If you specify both a public and system identifier, the public identifier resolves to the system identifier, and the system identifier is used.
4. If you have selected Public as the Type, enter the public identifier of your resource in the PUBLIC field. If you have selected Internal or System as your Type, the PUBLIC
field is disabled.

5. In the Value/Path field, you can enter any one of the following:

- If the entity type is Internal, enter the text string you want as the value of your entity. Do not enter quotes to delimit the entry. Any quotes that you enter will be treated as part of the text string.
- If the entity type is SYSTEM, enter the URI of the resource or select a resource on your local network by using the Browse button. If the resource contains parsed data, it must be an XML file (i.e., it must have a .xml extension). Alternatively, the resource can be a binary file, such as a GIF file.
- If the entity type is PUBLIC, you must additionally enter a system identifier in this field.

6. The NDATA entry tells the processor that this entity is not to be parsed but to be sent to the appropriate processor. The NDATA field must therefore contain some value to indicate that the entity is an unparsed entity.

**Dialog features**

You can do the following in the Define Entities dialog:

- Append entities
- Insert entities
- Delete entities
- Sort entities by the alphabetical value of any column by clicking the column header; clicking once sorts in ascending order, twice in descending order.
- Resize the dialog box and the width of columns.
- Locking. Once an entity is used in the XML document, it is locked and cannot be edited in the Define Entities dialog. Locked entities are indicated by a lock symbol in the first column. Locking an entity ensures that the XML document valid with respect to entities. (The document would be invalid if an entity is referenced but not defined.)
- Duplicate entities are flagged.

**Limitations of entities**

- An entity contained within another entity is not resolved, either in the dialog, Authentic View, or XSLT output, and the ampersand character of such an entity is displayed in its escaped form, i.e. &
- External unparsed entities that are not image files are not resolved in Authentic View. If an image in the design is defined to read an external unparsed entity and has its URI set to be an entity name (for example: 'logo'), then this entity name can be defined in the Define Entities dialog (see screenshot above) as an external unparsed entity with a value that resolves to the URI of the image file (as has been done for the logo entity in the screenshot above).
4.6 XML Signatures

An SPS can be designed with an XML signature configured for Authentic View. When XML signatures are enabled in the SPS, the Authentic View user can digitally sign the Authentic XML file with the enabled signature. After the document has been signed, any modification to it will cause the verification of the signature to fail. Whenever a signed Authentic XML document is opened in the Authentic View of any Altova product, the verification process will be run on the document and the result of the verification will be displayed in a window.

Note: XML signatures can be used, and will be verified, in the Authentic View of Enterprise and Professional editions of the following Altova products: Authentic Desktop, Authentic Browser, XMLSpy, and StyleVision.

XML signature actions
The following Authentic View user actions for signatures are possible:

- **Choosing the certificate/password:** Signatures are authenticated with either a certificate or a password. The authentication object (certificate or password) is required when the signature is created and again when it is verified. If an Authentic XML document has a signature-enabled SPS assigned to it, the SPS might specify a default certificate or password for the signature. Whether a default certificate or password has been specified or not, the signature can be configured to allow the Authentic View user to select an own certificate/password. The Authentic View user can do this at any time in the XML Signature dialog ([screenshot below](#)). Selecting an own certificate/password overrides the default certificate/password. The own certificate/password is stored in memory and is used for the current session. If, after an own certificate/password has been selected, the Authentic View user closes the file or the application, the SPS reverts to its default setting for the certificate/password.

- **Signing the document:** The Authentic XML document can be signed either automatically or manually. Automatic signing will have been specified in the signature configuration by the SPS designer and causes the Authentic XML document to be signed automatically when it is saved. If the automatic-signing option has not been activated, the document can be signed manually. This is done by clicking the XML Signature toolbar icon or the **Authentic | XML Signature** command, and, in the XML Signature dialog that then pops up ([screenshot above](#)), clicking the **Sign Document** button. Note that signing the document with an embedded signature would require the schema to allow the **Signature** element as the last child element of the root (document) element. Otherwise the document will be invalid against the schema. When signing the document, the authentication object and the placement of the signature are determined according to the signature configuration. You must ensure that you have access to the authentication information. For more information about this, consult your SPS designer.

- **Verifying the Authentic XML document:** If an SPS has XML Signatures enabled, the verification process will be run on the signature each time the Authentic View XML document is loaded. If the password or certificate key information is not saved with the SPS and signature, respectively, the Authentic View user will be prompted to enter the password or select a certificate for verification. Note that if an embedded signature is generated, it will be saved with the XML file when the XML file is saved. The generated signature must be explicitly removed (via the **Remove Signature** button of the XML Signature dialog; see [screenshot above](#)) if you do not wish to save it with the XML file.
Similarly, if a detached signature is generated, it too must be explicitly removed if it is not required.
Images in Authentic View

Authentic View allows you to specify images that will be used in the final output document (HTML, RTF, PDF and Word 2007). You should note that some image formats might not be supported in some formats or by some applications. For example, the SVG format is supported in PDF, but not in RTF and would require a browser add-on for it to be viewed in HTML. So, when selecting an image format, be sure to select a format that is supported in the output formats of your document. Most image formats are supported across all the output formats (see list below).

Authentic View is based on Internet Explorer, and is able to display most of the image formats that your version of Internet Explorer can display. The following commonly used image formats are supported:

- GIF
- JPG
- PNG
- BMP
- WMF (Microsoft Windows Metafile)
- EMF (Enhanced Metafile)
- SVG (for PDF output only)

Relative paths

Relative paths are resolved relative to the SPS file.
4.8 Keystrokes in Authentic View

The Enter key

In Authentic View the Enter key is used to append additional elements when it is in certain cursor locations. For example, if the chapter of a book may (according to the schema) contain several paragraphs, then pressing Enter inside the text of the paragraph causes a new paragraph to be appended immediately after the current paragraph. If a chapter can contain one title and several paragraphs, pressing Enter inside the chapter but outside any paragraph element (including within the title element) causes a new chapter to be appended after the current chapter (assuming that multiple chapters are allowed by the schema).

Please note: The Enter key does not insert a new line. This is the case even when the cursor is inside a text node, such as paragraph.

Using the keyboard

The keyboard can be used in the standard way, for typing and navigating. Note the following special points:

- The Tab key moves the cursor forward, stopping before and after nodes, and highlighting node contents; it steps over static content.
- The add... and add Node hyperlinks are considered node contents and are highlighted when tabbed. They can be activated by pressing either the spacebar or the Enter key.
5  Authentic Scripting

The Authentic Scripting feature provides more flexibility and interactivity to SPS designs. These designs can be created or edited in StyleVision Enterprise and Professional editions, and can be viewed in the Authentic View of the Enterprise and Professional editions of Altova products.

A complete listing of support for this feature in Altova products is given in the table below. Note, however, that in the trusted version of Authentic Browser plug-in, internal scripting is turned off because of security concerns.

<table>
<thead>
<tr>
<th>Altova Product</th>
<th>Authentic Scripts Creation</th>
<th>Authentic Scripts Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleVision Enterprise</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>StyleVision Professional</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>StyleVision Basic *</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>XMLSpy Enterprise</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>XMLSpy Professional</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>AuthenticDesktop Enterprise</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Authentic Browser Plug-in</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Authentic Browser Plug-in Enterprise Trusted **</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Authentic Browser Plug-in Enterprise Untrusted</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*   No AuthenticView
** Scripted designs displayed. No internal macro execution or event handling. External events fired.

Authentic Scripts behave in the same way in all Altova products, so no product-specific code or settings are required.

Authentic Script Warning Dialog

If a PXF file, or an XML file linked to an SPS, contains a script and the file is opened or switched to Authentic View, then a warning dialog (screenshot below) pops up.

You can choose one of the following options:

- Click Yes to add the folder containing the file to the Trusted Locations list for Authentic
scripts. Subsequently, all files in the trusted folder will be opened in Authentic View without this warning dialog being displayed first. The Trusted Locations list can be accessed via the menu command Authentic | Trusted Locations, and modified.

- Click No to not add the folder containing the file to the Trusted Locations list. The file will be displayed in Authentic View with scripts disabled. The Authentic Script Warning dialog will appear each time this file is opened in Authentic View. To add the file's folder to the Trusted Locations list subsequently, open the Trusted locations dialog via the menu command Authentic | Trusted Locations, and add the folder or modify as required.

For a description of the Trusted Locations dialog, see the description of the Authentic | Trusted Locations menu command in the User Reference.

**Note:** When Authentic Desktop is accessed via its COM interface (see Programmers' Reference to see how this can be done), the security check is not done and the Authentic Script Warning dialog is not displayed.

### How Authentic Scripting works

The designer of the SPS design can use Authentic Scripting in two ways to make Authentic documents interactive:

- By assigning scripts for user-defined actions (macros) to design elements, toolbar buttons, and context menu items.
- By adding to the design event handlers that react to Authentic View events.

All the scripting that is required for making Authentic documents interactive is done within the StyleVision GUI (Enterprise and Professional editions). Forms, macros and event handlers are created within the Scripting Editor interface of StyleVision and these scripts are saved with the SPS. Then, in the Design View of StyleVision, the saved scripts are assigned to design elements, toolbar buttons, and context menus. When an XML document based on the SPS is opened in an Altova product that supports Authentic Scripting (see table above), the document will have the additional flexibility and interactivity that has been created for it.

### Documentation for Authentic Scripting

The documentation for Authentic Scripting is available in the documentation of StyleVision. It can be viewed online via the Product Documentation page of the Altova website.
6 Browser View

Browser View is typically used to view:

- XML files that have an associated XSLT file. When you switch to Browser View, the XML file is transformed on the fly using the associated XSLT stylesheet and the result is displayed directly in Browser View.
- HTML files which are either created directly as HTML or created via an XSLT transformation of an XML file.

To view XML and HTML files in Browser View, click the Browser tab.

Note about Microsoft Internet Explorer and XSLT

Browser View requires Microsoft's Internet Explorer 5.0 or later. If you wish to use Browser View for viewing XML files transformed by an XSLT stylesheet, we strongly recommend Internet Explorer 6.0 or later, which uses MSXML 3.0, an XML parser that fully supports the XSLT 1.0 standard. You might also wish to install MSXML 4.0. Please see our Download Center for more details. (Note that support for XSLT in IE 5 is not 100% compatible with the official XSLT Recommendation. So if you encounter problems in Browser View with IE 5, you should upgrade to IE 6 or later.) You should also check the support for XSLT of your version of Internet Explorer.

Browser View features

The following features are available in Browser View. They can be accessed via the Browser menu, File menu, and Edit menu.

- **Open in separate window**: When Browser View is a separate window, it can be positioned side-by-side with an editing view of the same document. This command is in the Browser menu and is a toggle-command that can be used to return a separate Browser View window as a tabbed view. In the View tab of the Options dialog, you can set whether Browser View should, by default, be a separate window.
- **Forward and Back**: The common browser commands to navigate through pages that were loaded in Browser View. These commands are in the Browser menu.
- **Font size**: Can be adjusted via the Browser menu command.
- **Stop, Refresh, Print**: More standard browser commands, these can be found in the Browser and File menus.
- **Find**: Enables searches for text strings, this command is in the Edit menu.
7 Altova Global Resources

Altova Global Resources is a collection of aliases for file, folder, and database resources. Each alias can have multiple configurations, and each configuration maps to a single resource (see screenshot below). Therefore, when a global resource is used as an input, the global resource can be switched among its configurations. This is done easily via controls in the GUI that let you select the active configuration.

Using Altova Global Resources involves two processes:

- **Defining Global Resources**: Resources are defined and the definitions are stored in an XML file. These resources can be shared across multiple Altova applications.
- **Using Global Resources**: Within Authentic Desktop, files can be located via a global resource instead of via a file path. The advantage is that the resource can be switched by changing the active configuration in Authentic Desktop.

**Global resources in other Altova products**

Currently, global resources can be defined and used in the following individual Altova products: XMLSpy, StyleVision, MapForce, Authentic Desktop, MobileTogether Designer, and DatabaseSpy.
7.1 Defining Global Resources

Altova Global Resources are defined in the Manage Global Resources dialog, which can be accessed in two ways:

- Click the menu command **Tools | Global Resources**.
- Click the **Manage Global Resources** icon in the Global Resources toolbar (screenshot below).

The Global Resources Definitions file

Information about global resources is stored in an XML file called the Global Resources Definitions file. This file is created when the first global resource is defined in the Manage Global Resources dialog (screenshot below) and saved.

When you open the Manage Global Resources dialog for the first time, the default location and name of the Global Resources Definitions file is specified in the **Definitions File** text box (see screenshot above):

```
C:\Users\<username>\My Documents\Altova\GlobalResources.xml
```

This file is set as the default Global Resources Definitions file for all Altova applications. So a global resource can be saved from any Altova application to this file and will be immediately available to all other Altova applications as a global resource. To define and save a global resource to the Global Resources Definitions file, add the global resource in the Manage Global Resources dialog and click **OK** to save.

To select an already existing Global Resources Definitions file to be the active definitions file of a particular Altova application, browse for it via the **Browse** button of the **Definitions File** text box (see screenshot above).
**Note:** You can name the Global Resources Definitions file anything you like and save it to any location accessible to your Altova applications. All you need to do in each application, is specify this file as the Global Resources Definitions file for that application (in the *Definitions File* text box). The resources become global across Altova products when you use a single definitions file across all Altova products.

**Note:** You can also create multiple Global Resources Definitions files. However, only one of these can be active at any time in a given Altova application, and only the definitions contained in this file will be available to the application. The availability of resources can therefore be restricted or made to overlap across products as required.

Managing global resources: adding, editing, deleting, saving

In the Manage Global Resources dialog (screenshot above), you can add a global resource to the selected Global Resources Definitions file, or edit or delete a selected global resource. The Global Resources Definitions file organizes the global resources you add into groups: of files, folders, and databases (see screenshot above).

To **add a global resource**, click the **Add** button and define the global resource in the appropriate **Global Resource** dialog that pops up (see the descriptions of *files*, *folders*, and *databases* in the sub-sections of this section). After you define a global resource and save it (by clicking **OK** in the Manage Global Resources dialog), the global resource is added to the library of global definitions in the selected Global Resources Definitions file. The global resource will be identified by an alias.

To **edit a global resource**, select it and click **Edit**. This pops up the relevant **Global Resource** dialog, in which you can make the necessary changes (see the descriptions of *files*, *folders*, and *databases* in the sub-sections of this section).

To **delete a global resource**, select it and click **Delete**.

After you finish adding, editing, or deleting, make sure to click **OK** in the Manage Global Resources dialog to **save your modifications** to the Global Resources Definitions file.

Relating global resources to alias names via configurations

Defining a global resource involves mapping an alias name to a resource (file, folder, or database). A single alias name can be mapped to multiple resources. Each mapping is called a configuration. A single alias name can therefore be associated with several resources via different configurations (screenshot below).
In an Altova application, you can then assign aliases instead of files. For each alias you can switch between the resources mapped to that alias simply by changing the application's active Global Resource configuration (active configuration). For example, in Altova's XMLSpy application, if you wish to run an XSLT transformation on the XML document `MyXML.xml`, you can assign the alias `MyXSLT` to it as the global resource to be used for XSLT transformations. In XMLSpy you can then change the active configuration to use different XSLT files. If `Configuration-1` maps `First.xslt` to `MyXSLT` and `Configuration-1` is selected as the active configuration, then `First.xslt` will be used for the transformation. In this way multiple configurations can be used to access multiple resources via a single alias. This mechanism can be useful when testing and comparing resources. Furthermore, since global resources can be used across Altova products, resources can be tested and compared across multiple Altova products as well.
7.1.1 Files

The Global Resource dialog for Files (screenshot below) is accessed via the **Add | Files** command in the *Manage Global Resources dialog*. In this dialog, you can define configurations of the alias that is named in the *Resource Alias* text box. After specifying the properties of the configurations as explained below, save the alias definition by clicking **OK**.

After saving an alias definition, you can add another alias by repeating the steps given above (starting with the **Add | Files** command in the *Manage Global Resources dialog*).

**Global Resource dialog**

An alias is defined in the Global Resource dialog (screenshot below).

**Global Resource dialog icons**

- **Add Configuration**: Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.
Add Configuration as Copy: Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.

Delete: Deletes the selected configuration.

Open: Browse for the file to be created as the global resource.

Defining the alias

Define the alias (its name and configurations) as follows:

1. Give the alias a name: Enter the alias name in the Resource Alias text box.
2. Add configurations: The Configurations pane will have, by default, a configuration named Default (see screenshot above), which cannot be deleted or renamed. You can add as many additional configurations as you like by: (i) clicking the Add Configuration or Add Configuration as Copy icons, and (ii) giving the configuration a name in the dialog that pops up. Each added configuration will be shown in the Configurations list. In the screenshot above, two additional configurations, named Long and Short, have been added to the Configurations list. The Add Configuration as Copy command enables you to copy the selected configuration and then modify it.
3. Select a resource type for each configuration: Select a configuration from the Configurations list, and, in the Settings for Configuration pane, specify a resource for the configuration: (i) File, (ii) Output of an Altova MapForce transformation, or (iii) Output of an Altova StyleVision transformation. Select the appropriate radio button. If a MapForce or StyleVision transformation option is selected, then a transformation is carried out by MapForce or StyleVision using, respectively, the .mfd or .sps file and the respective input file. The result of the transformation will be the resource.
4. Select a file for the resource type: If the resource is a directly selected file, browse for the file in the Resource File Selection text box. If the resource is the result of a transformation, in the File Selection text box, browse for the .mfd file (for MapForce transformations) or the .sps file (for StyleVision transformations). Where multiple inputs or outputs for the transformation are possible, a selection of the options will be presented. For example, the output options of a StyleVision transformation are displayed according to what edition of StyleVision is installed (the screenshot below shows the outputs for Enterprise Edition).

Select the radio button of the desired option (in the screenshot above, 'HTML output' is selected). If the resource is the result of a transformation, then the output can be saved as a file or itself as a global resource. Click the icon and select, respectively, Global Resource (for saving the output as a global resource) or Browse (for saving the output as a file). If neither of these two saving options is selected, the transformation result will be loaded as a temporary file when the global resource is invoked.
5. **Define multiple configurations if required:** You can add more configurations and specify a resource for each. Do this by repeating Steps 3 and 4 above for each configuration. You can add a new configuration to the alias definition at any time.

6. **Save the alias definition:** Click **OK** to save the alias and all its configurations as a global resource. The global resource will be listed under Files in the Manage Global Resources dialog.

### Result of MapForce transformation

Altova MapForce maps one or more (existing) input document schemas to one or more (new) output document schemas. This mapping, which is created by a MapForce user, is known as a MapForce Design (MFD). XML files, text files, databases, etc., that correspond to the input schema/s can be used as data sources. MapForce generates output data files that correspond to the output document schema. This output document is the *Result of MapForce Transformation* file that will become a global resource.

If you wish to set a MapForce-generated data file as a global resource, the following must be specified in the Global Resource dialog (see screenshot below):

- **A `.mfd` (MapForce Design) file.** You must specify this file in the Resource will point to generated output of text box (see screenshot above).
- **One or more input data files.** After the MFD file has been specified, it is analyzed and, based on the input schema information in it, default data file/s are entered in the Inputs pane (see screenshot above). You can modify the default file selection for each input schema by specifying another file.
• **An output file.** If the MFD document has multiple output schemas, all these are listed in the *Outputs* pane (see screenshot above) and you must select one of them. If the output file location of an individual output schema is specified in the MFD document, then this file location is entered for that output schema in the *Outputs* pane. From the screenshot above we can see that the MFD document specifies that the *Customers* output schema has a default XML data file (*CustomersOut.xml*), while the *Text file* output schema does not have a file association in the MFD file. You can use the default file location in the *Outputs* pane or specify one yourself. The result of the MapForce transformation will be saved to the file location of the selected output schema. This is the file that will be used as the global resource.

**Note:** The advantage of this option (Result of MapForce transformation) is that the transformation is carried out at the time the global resource is invoked. This means that the global resource will contain the most up-to-date data (from the input file/s).

**Note:** Since MapForce is used to run the transformation, you must have Altova MapForce installed for this functionality to work.

**Result of StyleVision transformation**

Altova StyleVision is used to create StyleVision Power Stylesheet (SPS) files. These SPS files generate XSLT stylesheets that are used to transform XML documents into output documents in various formats (HTML, PDF, RTF, Word 2007+, etc). If you select the option *Result of StyleVision Transformation*, the output document created by StyleVision will be the global resource associated with the selected configuration.

For the *StyleVision Transformation* option in the Global Resource dialog (see screenshot below), the following files must be specified.
A .sps (SPS) file. You must specify this file in the Resource will point to generated output of text box (see screenshot above).

Input file/s. The input file might already be specified in the SPS file. If it is, it will appear automatically in the Inputs pane once the SPS file is selected. You can change this entry. If there is no entry, you must add one.

Output file/s. Select the output format in the Outputs pane, and specify an output file location for that format.

Note: The advantage of this option (Result of StyleVision transformation) is that the transformation is carried out when the global resource is invoked. This means that the global resource will contain the most up-to-date data (from the input file/s).

Note: Since StyleVision is used to run the transformation, you must have Altova StyleVision installed for this functionality to work.
7.1.2 Folders

In the Global Resource dialog for Folders (screenshot below), add a folder resource as described below.

Global Resource dialog icons

- Add Configuration: Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.
- Add Configuration as Copy: Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.
- Delete: Deletes the selected configuration.
- Open: Browse for the folder to be created as the global resource.

Defining the alias

Define the alias (its name and configurations) as follows:

1. Give the alias a name: Enter the alias name in the Resource Alias text box.
2. Add configurations: The Configurations pane will have a configuration named Default (see screenshot above). This Default configuration cannot be deleted nor have its name changed. You can enter as many additional configurations for the selected alias as you like. Add a configuration by clicking the Add Configuration or Add Configuration as Copy icons. In the dialog which pops up, enter the configuration name. Click OK. The new configuration will be listed in the Configurations pane. Repeat for as many configurations as you want.
3. **Select a folder as the resource of a configuration:** Select one of the configurations in the Configurations pane and browse for the folder you wish to create as a global resource. If security credentials are required to access a folder, then specify these in the *Username* and *Password* fields.

4. **Define multiple configurations if required:** Specify a folder resource for each configuration you have created (that is, repeat Step 3 above for the various configurations you have created). You can add a new configuration to the alias definition at any time.

5. **Save the alias definition:** Click **OK** in the Global Resource dialog to save the alias and all its configurations as a global resource. The global resource will be listed under Folders in the [Manage Global Resources dialog](#).
7.1.3 Databases

In the Global Resource dialog for Databases (screenshot below), you can add a database resource as follows:

Global Resource dialog icons

- **Add Configuration**: Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.
- **Add Configuration as Copy**: Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.
- **Delete**: Deletes the selected configuration.
Defining the alias

Define the alias (its name and configurations) as follows:

1. **Give the alias a name:** Enter the alias name in the *Resource Alias* text box.
2. **Add configurations:** The Configurations pane will have a configuration named Default (see screenshot above). This Default configuration cannot be deleted nor have its name changed. You can enter as many additional configurations for the selected alias as you like. Add a configuration by clicking the **Add Configuration** or **Add Configuration as Copy** icons. In the dialog which pops up, enter the configuration name. Click **OK**. The new configuration will be listed in the Configurations pane. Repeat for as many configurations as you want.

3. **Start selection of a database as the resource of a configuration:** Select one of the configurations in the Configurations pane and click the **Choose Database** icon. This pops up the Create Global Resources Connection dialog.

4. **Connect to the database:** Select whether you wish to create a connection to the database using the Connection Wizard, an existing connection, an ADO Connection, an ODBC Connection, or JDBC Connection.

5. **Select the root object:** If you connect to a database server where a root object can be selected, you will be prompted, in the Choose Root Object dialog (screenshot below), to select a root object on the server. Select the root object and click **Set Root Object**. The root object you select will be the root object that is loaded when this configuration is used.

If you choose not to select a root object (by clicking the **Skip** button), then you can select the root object at the time the global resource is loaded.

6. **Define multiple configurations if required:** Specify a database resource for any other configuration you have created (that is, repeat Steps 3 to 5 above for the various configurations you have created). You can add a new configuration to the alias definition at any time.

7. **Save the alias definition:** Click **OK** in the Global Resource dialog to save the alias and all its configurations as a global resource. The global resource will be listed under databases in the Manage Global Resources dialog.
7.2 Using Global Resources

There are several types of global resources (file-type, folder-type, and database-type). Some scenarios in which you can use global resources in Authentic Desktop are listed here: Files and Folders.

Selections that determine which resource is used

There are two application-wide selections that determine what global resources can be used and which global resources are actually used at any given time:

- **The active Global Resources XML File** is selected in the Global Resource dialog. The global-resource definitions that are present in the active Global Resources XML File are available to all files that are open in the application. Only the definitions in the active Global Resources XML File are available. The active Global Resources XML File can be changed at any time, and the global-resource definitions in the new active file will immediately replace those of the previously active file. The active Global Resources XML File therefore determines: (i) what global resources can be assigned, and (ii) what global resources are available for look-up (for example, if a global resource in one Global Resource XML File is assigned but there is no global resource of that name in the currently active Global Resources XML File, then the assigned global resource (alias) cannot be looked up).

- **The active configuration** is selected via the menu item Tools | Active Configuration or via the Global Resources toolbar. Clicking this command (or drop-down list in the toolbar) pops up a list of configurations across all aliases. Selecting a configuration makes that configuration active application-wide. This means that wherever a global resource (or alias) is used, the resource corresponding to the active configuration of each used alias will be loaded. The active configuration is applied to all used aliases. If an alias does not have a configuration with the name of the active configuration, then the default configuration of that alias will be used. The active configuration is not relevant when assigning resources; it is significant only when the resources are actually used.
7.2.1 Assigning Files and Folders

File-type and folder-type global resources are assigned differently. In any one of the usage scenarios below, clicking the **Switch to Global Resources** button displays the Open Global Resource dialog (screenshot below).

![Open Global Resource dialog](image)

**Manage Global Resources:** Displays the Manage Global Resources dialog.

Selecting a *file-type global resource* assigns the file. Selecting a *folder-type global resource* causes an Open dialog to open, in which you can browse for the required file. The path to the selected file is entered relative to the folder resource. So if a folder-type global resource were to have two configurations, each pointing to different folders, files having the same name but in different folders could be targeted via the two configurations. This could be useful for testing purposes.

You can switch to the file dialog or the URL dialog by clicking the respective button at the bottom of the dialog. The **Manage Global Resources** icon in the top right-hand corner pops up the Manage Global Resources dialog.

**Usage scenarios**

File-type and folder-type global resources can be used in the following scenarios:

- Opening global resources
- Saving as global resource
- XSLT transformation
- Assigning an SPS
Opening global resources

A global resource can be opened in Authentic Desktop with the File | Open (Switch to Global Resource) command and can be edited. In the case of a file-type global resource, the file is opened directly. In the case of a folder-type global resource, an Open dialog pops up with the associated folder selected. You can then browse for the required file in descendant folders. One advantage of addressing files for editing via global resources is that related files can be saved under different configurations of a single global resource and accessed merely by changing configurations. Any editing changes would have to be saved before changing the configuration.

Saving as global resource

A newly created file can be saved as a global resource. Also, an already existing file can be opened and then saved as a global resource. When you click the File | Save or File | Save As commands, the Save dialog appears. Click the Switch to Global Resource button to access the available global resources (screenshot below), which are the aliases defined in the current Global Resources XML File.

Select an alias and then click Save. If the alias is a file alias, the file will be saved directly. If the alias is a folder alias, a dialog will appear that prompts for the name of the file under which the file is to be saved. In either case the file will be saved to the location that was defined for the currently active configuration.

Note: Each configuration points to a specific file location, which is specified in the definition of that configuration. If the file you are saving as a global resource does not have the same filetype extension as the file at the current file location of the configuration, then there might be editing and validation errors when this global resource is opened in Authentic Desktop. This is because Authentic Desktop will open the file assuming the filetype
specified in the definition of the configuration.

**XSLT transformations**

Clicking the command **XSL/XQuery | XSL Transformation** or **XSL/XQuery | XSL:FO Transformation** pops up a dialog in which you can browse for the required XSLT or XML file. Click the **Browse** button and then the **Switch to Global Resource** button to pop up the Open Global Resource dialog (screenshot at top of section). The file that is associated with the currently active configuration of the selected global resource is used for the transformation.

**Assigning an SPS**

When assigning a StyleVision stylesheet to an XML file (**Authentic | Assign StyleVision Stylesheet**), you can select a global resource to locate the stylesheet. Click the **Browse** button and then the **Switch to Global Resource** button to pop up the Open Global Resource dialog (screenshot at top of section). With a global resource selected as the assignment, the Authentic View of the XML document can be changed merely by changing the active configuration in Authentic Desktop.
7.2.2 Changing the Active Configuration

One configuration of a global resource can be active at any time. This configuration is called the active configuration, and it is active application-wide. This means that the active configuration is active for all global resources aliases in all currently open files and data source connections. If an alias does not have a configuration with the name of the active configuration, then the default configuration of that alias will be used. As an example of how to change configurations, consider the case in which a file has been assigned via a global resource with multiple configurations. Each configuration maps to a different file. So, which file is selected depends on which configuration is selected as the application's active configuration.

Switching the active configuration can be done in the following ways:

- Via the menu command Tools | Active Configuration. Select the configuration from the command's submenu.
- In the combo box of the Global Resources toolbar (screenshot below), select the required configuration.

![Screenshot](default.png)

In this way, by changing the active configuration, you can change source files that are assigned via a global resource.
8 Source Control

The source control support in Authentic Desktop is available through the Microsoft Source Control Plug-in API (formerly known as the MSSCCI API), versions 1.1, 1.2 and 1.3. This enables you to run source control commands such as “Check in” or “Check out” directly from Authentic Desktop to virtually any source control system that lets native or third-party clients connect to it through the Microsoft Source Control Plug-in API.

You can use as your source control provider any commercial or non-commercial plug-in that supports the Microsoft Source Control Plug-in API, and can connect to a compatible version control system. For the list of source control systems and plug-ins tested by Altova, see Supported Source Control Systems.

Installing and configuring the source control provider

To view the source control providers available on your system, do the following:

1. On the Tools menu, click Options.
2. Click the Source Control tab.

Any source control plug-ins compatible with the Microsoft Source Code Control Plug-in API are displayed in the Current source control plug-in drop-down list.

If a compatible plug-in cannot be found on your system, the following message is displayed:

"Registration of installed source control providers could not be found or is incomplete."

Some source control systems might not install the source control plug-in automatically, in which case you will need to install it separately. For further instructions, refer to the documentation of the respective source control system. A plug-in (provider) compatible with the Microsoft Source Code Control Plug-in API is expected to be registered under the following registry entry on your
operating system:

HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders

Upon correct installation, the plug-in becomes available automatically in the list of plug-ins available to Authentic Desktop.

Accessing the source control commands
The commands related to source control are available in the Project | Source Control menu.

Resource / Speed issues
Very large source control databases might be introducing a speed/resource penalty when automatically performing background status updates.

You might be able to speed up your system by disabling (or increasing the interval of) the Perform background status updates every ... seconds option in the Source Control tab accessed through Tools | Options.

Note: The 64-bit version of your Altova application automatically supports any of the supported 32-bit source control programs listed in this documentation. When using a 64-bit Altova application with a 32-bit source control program, the Perform background status updates every ... seconds option is automatically grayed-out and cannot be selected.

Differencing with Altova DiffDog
You can configure many source control systems (including Git and TortoiseSVN) so that they use Altova DiffDog as their differencing tool. For more information about DiffDog, see https://www.altova.com/diffdog. For DiffDog documentation, see https://www.altova.com/documentation.html.
8.1 Setting Up Source Control

The mechanism for setting up source control and placing files in a Authentic Desktop project under source control is as follows:

1. If this hasn't been done already, install the source control system (see Supported Source Control Systems) and set up the source control database (repository) to which you wish to save your work.

2. Create a local workspace folder that will contain the working files that you wish to place under source control. The folder that contains all your workspace folders and files is called the local folder, and the path to the local folder is referred to as the local path. This local folder will be bound to a particular folder in the repository.

3. In your Altova application, create an application project folder to which you must add the files you wish to place under source control. This organization of files in an application project is abstract. The files in a project reference physical files saved locally, preferably in one folder (with sub-folders if required) for each project.

4. In the source control system's database (also referred to as source control or repository), a folder is created that is bound to the local folder. This folder (called the bound folder) will replicate the structure of the local folder so that all files to be placed under source control are correctly located hierarchically within the bound folder. The bound folder is usually created when you add a file or an application project to source control for the first time. See the section, Application Project, for information about the repository’s folder structure.

5. Project files are added to source control using the command Project | Source Control | Add to Source Control. When you add a project or a file in a project for the first time to source control, the correct bindings and folder structure will be created in the repository.

6. Source control actions, such as the checking in and out of files, and the removing of files from source control, can be carried out via commands in the Project | Source Control submenu. These commands are described in the Project menu subsection of the User Reference.

Note: If you wish to change the current source control provider, this can be done in one of two ways: (i) via the Source Control options (Tools | Options | Source Control), or (ii) in the Change Source Control dialog (Project | Source Control | Change Source Control).
8.2 **Supported Source Control Systems**

The list below shows the Source Control Servers (SCSs) supported by Authentic Desktop, together with their respective Source Control Clients (SCCs). The list is organized alphabetically by SCS. Note the following:

- Altova has implemented the Microsoft Source Control Plug-in API (versions 1.1, 1.2, and 1.3) in Authentic Desktop, and has tested support for the listed drivers and revision control systems. It is expected that Authentic Desktop will continue to support these products if, and when, they are updated.
- Source Code Control clients not listed below, but which implement the Microsoft Source Control Plug-in API, should also work with Authentic Desktop.

<table>
<thead>
<tr>
<th>Source Control System</th>
<th>Source Code Control Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccuRev 4.7.0 Windows</td>
<td>AccuBridge for Microsoft SCC 2008.2</td>
</tr>
<tr>
<td>Bazaar 1.9 Windows</td>
<td>Aigenta Unified SCC 1.0.6</td>
</tr>
<tr>
<td>Borland StarTeam 2008</td>
<td>Borland StarTeam Cross-Platform Client 2008 R2</td>
</tr>
<tr>
<td>Codice Software Plastic SCM Professional 2.7.127.10 (Server)</td>
<td>Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)</td>
</tr>
</tbody>
</table>
| Collabnet Subversion 1.5.4                    | • Aigenta Unified SCC 1.0.6  
  • PushOK SVN SCC 1.5.1.1  
  • PushOK SVN SCC x64 version 1.6.3.1  
  • TamTam SVN SCC 1.2.24 |
| ComponentSoftware CS-RCS (PRO) 5.1            | ComponentSoftware CS-RCS (PRO) 5.1                                                           |
| Dynamsoft SourceAnywhere for VSS 5.3.2 Standard/Professional Server | Dynamsoft SourceAnywhere for VSS 5.3.2 Client                                                 |
| Dynamsoft SourceAnywhere Hosted               | Dynamsoft SourceAnywhere Hosted Client (22252)                                              |
| Dynamsoft SourceAnywhere Standalone 2.2 Server | Dynamsoft SourceAnywhere Standalone 2.2 Client                                                |
| Git                                           | PushOK GIT SCC plug-in (see Source Control with Git )                                       |
| IBM Rational ClearCase 7.0.1 (LT)             | IBM Rational ClearCase 7.0.1 (LT)                                                            |
| March-Hare CVSNT 2.5 (2.5.03.2382)            | Aigenta Unified SCC 1.0.6                                                                   |
| March-Hare CVS Suite 2008                     | • Jalindi Igloo 1.0.3  
  • March-Hare CVS Suite Client 2008 (3321)  
  • PushOK CVS SCC NT 2.1.2.5  
  • PushOK CVS SCC x64 version 2.2.0.4  
  • TamTam CVS SCC 1.2.40 |
<p>| Mercurial 1.0.2 for Windows                    | Sergey Antonov HgSCC 1.0.1                                                                   |
| Microsoft SourceSafe 2005 with CTP            | Microsoft SourceSafe 2005 with CTP                                                           |</p>
<table>
<thead>
<tr>
<th>Source Control System</th>
<th>Source Code Control Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>PureCM Server 2008/3a</td>
<td>PureCM Client 2008/3a</td>
</tr>
<tr>
<td>QSC Team Coherence Server 7.2.1.35</td>
<td>QSC Team Coherence Client 7.2.1.35</td>
</tr>
<tr>
<td>Reliable Software Code Co-Op 5.1a</td>
<td>Reliable Software Code Co-Op 5.1a</td>
</tr>
<tr>
<td>Seapine Surround SCM Client/Server for Windows 2009.0.0</td>
<td>Seapine Surround SCM Client 2009.0.0</td>
</tr>
<tr>
<td>Serena Dimensions Express/CM 10.1.3 for Win32 Server</td>
<td>Serena Dimensions 10.1.3 for Win32 Client</td>
</tr>
<tr>
<td>Softimage Alienbrain Server 8.1.0.7300</td>
<td>Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300</td>
</tr>
<tr>
<td>SourceGear Fortress 1.1.4 Server</td>
<td>SourceGear Fortress 1.1.4 Client</td>
</tr>
<tr>
<td>SourceGear SourceOffsite Server 4.2.0</td>
<td>SourceGear SourceOffsite Client 4.2.0 (Windows)</td>
</tr>
<tr>
<td>SourceGear Vault 4.1.4 Server</td>
<td>SourceGear Vault 4.1.4 Client</td>
</tr>
<tr>
<td>VisualSVN Server 1.6</td>
<td>- Aigenta Unified SCC 1.0.6</td>
</tr>
<tr>
<td></td>
<td>- PushOK SVN SCC 1.5.1.1</td>
</tr>
<tr>
<td></td>
<td>- PushOK SVN SCC x64 version 1.6.3.1</td>
</tr>
<tr>
<td></td>
<td>- TamTam SVN SCC 1.2.24</td>
</tr>
</tbody>
</table>
8.3 Local Workspace Folder

The files you will be working with should be saved in a hierarchy inside a local workspace folder (see diagram below).

```
Local Workspace Folder
|-- MyProject.spp
|-- QuickStart
| |-- QuickStart.css
| |-- QuickStart.xml
| |-- QuickStart.xsd
|-- Grouping
| |-- Persons
| | |-- Persons.xml
```

The application project file (.spp file) typically will be located directly inside the local workspace folder (see diagram above).

When one or more files in this (workspace) folder are placed under source control, the local workspace folder's structure is partly or wholly reproduced in the repository. For example, if the file `Persons.xml` from the local folder shown above is placed under source control, then the path to it in the repository will be:

```
[RepositoryFolder]/MyProject/Grouping/Persons/Persons.xml
```

The `MyProject` folder in the repository folder is bound to the local folder. Typically it would be the name of the project, but you could give it any name.

If the entire application project is placed under source control (by selecting the project name in the Projects window and placing it under source control), then the entire local folder structure is recreated in the repository.

**Note:** Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.
8.4 Application Project

Create or load the Altova application project you wish to place under source control. If you wish to place a single file under source control, this file must be included in a project—since source control can only be accessed via a project.

For example, consider a project in Altova's XMLSpy application. The project's properties are saved in a .spp file. In the application, the project is displayed in the application's Project window (see screenshot below). The project in the screenshot below is named MyProject and the project's properties are saved in the file MyProject.spp.

You can place the entire project (all files in the project) or only some project files under source control. Only files that are in the project can be placed under source control. So you will need to add files to the project before you can place them under source control. The project file (.spp file) will automatically be placed under source control as soon as a file from within the project is placed under source control.

The entire project, or one or more project files, is placed under source control via the command Project | Source Control | Add to Source Control (see next section below).

Note, however, that the folder structure of the repository corresponds not to the project's folder structure (screenshot above) but to the structure of the local workspace folder (see folder diagram below). In the diagram below, notice that the MyProject folder in the repository has a folder structure corresponding to that of the local workspace folder. Note that the bound folder occurs within the repository folder.
Note: An application project can contain project folders (green) and external folders (yellow). Only files in (green) project folders can be placed under source control. Files in (yellow) external folders cannot be placed under source control.

Note: Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.
8.5 Add to Source Control

Adding the project to source control will automatically create the correct bindings and repository structure before adding the project file (.spp file) or individual files to source control. Add the project to source control as follows.

Select the project in the Project window (MyProject in the screenshot below) so that it is highlighted (as in the screenshot below). Alternatively select a single file, or select multiple files by clicking them with the Ctrl key pressed. Adding a single file to source control will automatically add the project file (.spp file) to source control as well.

Next, select the menu command Project | Source Control | Add to Source Control. This pops up the connection and configuration dialogs of the currently selected source control system. (You can change the source control system via the Change Source Control dialog (Project | Source Control | Change Source Control).)

Follow the source control system's instructions to make the connection and configuration. After this has been completed, all the files selected for addition plus the project file (.spp file) are displayed in an Add to Source Control dialog (screenshot below). Select the files you wish to add and click OK.
The files will be added to the repository and be either checked in or checked out depending on whether the Keep Checked Out check box has been checked or not.

**Configuration notes**

You might be prompted to create a folder in the repository for the project if it has not already been created. If you are, go ahead and create it. The **local workspace folder** will be bound to this folder created in the repository (see diagrams below).

```
Local Workspace Folder | Repository
----------------------|-------------------
| [-- MyProject.spp   | [-- MyProject (bound to Local Workspace)
| |-- QuickStart      | | |-- MyProject.spp
| | |-- QuickStart.css| | | |-- QuickStart
| | |-- QuickStart.xml| | | |-- QuickStart.css
| | |-- QuickStart.xsd| | | |-- QuickStart.xml
| |-- Grouping        | | |-- QuickStart.xsd
| | |-- Persons        | | |-- Grouping
| | | |-- Persons.xml  | | | |-- Persons
| | | | |-- Persons.xml
```

The configuration dialog of Jalindi Igloo is show below. The CVSROOT field is the path to the repository folder.

In the screenshot above, the local path locates the local workspace folder, which corresponds to the CVS module, MyProject, and is bound to it.
8.6 Working with Source Control

To work with source control, select the project, a project folder, or a project file in the Project window (screenshot below) and then select the command you want in the Project | Source Control menu. The Check In and Check Out commands are available as context menu commands of Project window items.

In this section, we describe the main source control features in detail:

- Add to, Remove from Source Control
- Check Out, Check In
- Getting Files as Read-Only
- Copying and Sharing from Source Control
- Changing Source Control

Additional commands in the Project | Source Control menu are described in the User Reference section of the manual. For information specific to a particular source control system, please see the user documentation of that system.
8.6.1 Add to, Remove from Source Control

Adding

After a project has been added to source control, you can place files either singly or in groups under source control. This is also known as adding the files to source control. Select the file in the Project window and then click the command Project | Source Control | Add to Source Control. To select multiple files, keep the Ctrl key pressed while clicking on the files you wish to add. Running the command on a (green) project folder (see screenshot below) adds all files in the folder and its sub-folders to source control.

When files are added to source control, the local folder hierarchy is replicated in the repository (it is not the project folder hierarchy that is replicated). So, if a file is in a sub-folder X levels deep in the local folder, then the file's parent folder and all other ancestor folders are automatically created in the repository.

When the first file from a project is added to source control, the correct bindings are created in the repository and the project file (.spp file) is added automatically. For more details, see the section Add to Source Control.

Source control symbols

Files and the project folder display certain symbols, the meanings of which are given below.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon]</td>
<td>Checked in. Available for check-out.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Checked out by another user. Not available for check-out.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Checked out locally. Can be edited and checked-in.</td>
</tr>
</tbody>
</table>

Removing

To remove a file from source control, select the file and click the command Project | Source Control | Remove from Source Control. You can also remove: (i) files in a project folder by executing the command on the folder, and (ii) the entire project by executing the command on the project.
8.6.2 Check Out, Check In

After a project file has been placed under source control, it can be checked out or checked in by selecting the file (in the Project window) and clicking the respective command in the Project | Source Control menu: Check Out and Check In.

When a file is checked out, a copy from the repository is placed in the local folder. A file that is checked out can be edited. If a file that is under source control is not checked out, it cannot be edited. After a file has been edited, the changes can be saved to the repository by checking in the file. Even if the file is not saved in the application, checking it in will save the changes to the repository. Whether a file is checked out or not is indicated with a tick or lock symbol in its Project window icon.

Files and the project folder display certain symbols, the meanings of which are given below.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Symbol]</td>
<td>Checked out by another user. Not available for check-out.</td>
</tr>
<tr>
<td>![Symbol]</td>
<td>Checked out locally. Can be edited and checked-in.</td>
</tr>
</tbody>
</table>

Selecting the project or a folder within the project selects all files in the selected object. To select multiple objects (files and folders), press the Ctrl key while clicking the objects. The screenshot below shows a project that has been checked out. The file QuickStart.css has subsequently been checked in.

Saving and rejecting editing changes

Note that, when checking in a file, you can choose to leave the file checked out. What this does is save editing changes to the repository while continuing to keep the file checked out, which is useful if you wish to periodically save editing changes to the repository and then continue editing.

If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command Project | Source Control | Undo Check Out.
Checking out
The Check Out dialog (screenshot below) allows you: (i) to select the files to check out, and (ii) to select whether the repository version or the local version should be checked out.

Checking in
The Check In dialog (screenshot below) allows you: (i) to select the files to check in, and (ii) if you wish, to keep the file checked out.

Note: In both dialogs (Check Out and Check In), multiple files appear if the selected object (project or project folder/s) contain multiple files.
8.6.3 Getting Files as Read-Only

The **Get** command (in the Project | Source Control menu) retrieves files from the repository as read-only files. (To be able to edit a file, you must check it out.) The Get dialog lists the files in the object (project or folder) on which the Get command was executed (see screenshot below). You can select the files to retrieve by checking them in the Get dialog list.

**Note:** The **Get Folders** command allows you to select individual sub-folders in the repository if this is allowed by your source control system.

You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking **OK**, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (screenshot below) asking whether you wish to reload the file/s (**Reload** button), close the file/s (**Close**), or retain the current view of the file (**Cancel**).

**Advanced Get Options**

The Advanced Get Options dialog (screenshot below) is accessed via the **Advanced** button in the Get dialog (see first screenshot in this section).
Here you can set options for (i) replacing writable files that are checked out, (ii) the timestamp, and (iii) whether the read-only property of the retrieved file should be changed so that it will be writable.

**Get latest version**

The Get Latest Version command (in the Project | Source Control menu) retrieves and places the latest source control version of the selected file(s) in the working directory. The files are retrieved as read-only and are not checked out. This command works like the Get command (see above), but does not display the Get dialog.

If the selected files are currently checked out, then the action taken will depend on how your source control system handles such a situation. Typically, the source control system will ask whether you wish to replace, merge with, or leave the checked-out file as it is.

**Note:** This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.
8.6.4 Copying and Sharing from Source Control

The Open from Source Control command creates a new application project from a project under source control.

Create the new project as follows:

1. Depending on the source control system used, it might be necessary, before you create a new project from source control, to make sure that no file from the source-controlled project is checked out.
2. No project need be open in the application, but can be.
3. Select the command Project | Source Control | Open from Source Control.
4. The source control system that is currently set will pop up its verification and connection dialogs. Make the connection to the bound folder in the repository that you want to copy.
5. In the dialog that pops up (screenshot below), browse for the local folder to which the contents of the bound folder in the repository (that you have just connected to) must be copied. In the screenshot below the bound folder is called MyProject and is represented by the $ sign; the local folder is C:\M20130326.

![Screenshot of Create local project from SourceSafe dialog]

6. Click OK. The contents of the bound folder (MyProject) will be copied to the local folder C:\M20130326, and a dialog pops up asking you to select the project file (.spp file) that is to be created as the new project.
7. Select the .spp file that will have been copied to the local folder. In our example, this will be MyProject.spp located in the C:\M20130326 folder. A new project named MyProject will be created in the application and will be displayed in the Project window. The project's files will be in the folder C:\M20130326.

Sharing from source control

The Share from Source Control command is supported when the source control system being used supports shares. You can share a file, so that it is available at multiple local locations. A
change made to one of these local files will be reflected in all the other "shared" versions.

In the application's Project window first select the project (highlighted in the screenshot below). Then click the **Share from Source Control**.

The Share To [Folder] dialog (screenshot below) pops up.

To select the files to share, first choose, in the project tree in the right-hand pane of the dialog (see screenshot above), the folder in which the files are. The files in the chosen folder are displayed in the left-hand pane. Select the file you wish to share (multiple files by pressing the **Ctrl** key and clicking the files you want to share). The selected file/s will be displayed in the **Files to Share** text box (at top left). The files disappear from the left hand pane. Click **Share** and then **Close** to copy the selected file/s to the local share folder. When you click **Close**, the files to share will be copied to the selected local location.
The share folder is noted in the name of the Share to [Folder] dialog. In the screenshot above it is the local folder (since the $ sign is the folder in the repository to which the local folder is bound). You can see and set the share folder in the Change Source Control dialog (screenshot below, Change Source Control) by changing the local path and server binding.

For more details about sharing using your source control system, see the source control system’s user documentation.
8.6.5 Changing Source Control

Source control settings can be changed via two commands in the Project | Source Control menu:

- **Source Control Manager**, which opens the source control system application and allows you to set up databases and configure bindings.
- **Change Source Control**, which pops up the Change Source Control dialog, in which you can change the source control system being used by the Altova application and the current binding. This dialog is described below.

The current binding is what the active application project will use to connect to the source control database. The current binding is correct when the application project file (.spp file) is in the local folder and the bound folder in the repository is where this project's files are stored. Typically the bound folder and its sub-structure will correspond with the local workspace folder and its sub-structure.

In the Change Source Control dialog (screenshot below), you can change the source control system (SCC Provider), the local folder (Local Path), and the repository binding (Server Name and Server Binding).

Only after undoing the current binding can the settings be changed. Undo the current binding with the **Undo** button. All the settings are now editable.

![Change Source Control dialog](image)

Change source control settings as follows:

1. Use the **Browse** button to browse for the local folder and the **Select** button to select from among the installed source control systems.
2. After doing this you can bind the local folder to a repository database. Click the **Bind** button to do this. This pops up the connection dialog of your source control system.
3. If you have entered a **Logon ID**, this will be passed to the source control system; otherwise you might have to enter your logon details in the connection dialog.
4. Select the database in the repository that you wish to bind to this local folder. This setting might be spread over more than one dialog.
5. After the setting has been created, click **OK** in the Change Source Control dialog.
### 8.7 Source Control with Git

Support for Git as a source control system in Authentic Desktop is available through a third-party plug-in called **GIT SCC plug-in** ([http://www.pushok.com/software/git.html](http://www.pushok.com/software/git.html)).

At the time when this documentation is written, the **GIT SCC plug-in** plug-in is available for experimental use. Registration with the plug-in publisher is required in order to use the plug-in.

The **GIT SCC plug-in** enables you to work with a Git repository using the commands available in the **Project | Source Control** menu of Authentic Desktop. Note that the commands in the **Project | Source Control** menu of Authentic Desktop are provided by the Microsoft Source Control Plug-in API (MSSCCI API), which uses a design philosophy different from Git. As a result, the plug-in essentially intermediates between "Visual Source Safe"-like functionality and Git functionality. On one hand, this means that a command such as **Get latest version** may not be applicable with Git. On the other hand, there are new Git-specific actions, which are available in the "Source Control Manager" dialog box provided by the plug-in (under the **Project | Source Control | Source Control Manager** menu of Authentic Desktop).

*The Source Control Manager dialog box*

Other commands that you will likely need to use frequently are available directly under the **Project | Source Control** menu.

The following sections describe the initial configuration of the plug-in, as well as the basic workflow:

- **Enabling Git Source Control with GIT SCC Plug-in**
- **Adding a Project to Git Source Control**
- **Cloning a Project from Git Source Control**
8.7.1 Enabling Git Source Control with Git SCC Plug-in

To enable Git source control with Authentic Desktop, the third-party **PushOK GIT SCC plug-in** must be installed, registered, and selected as source control provider, as follows:

1. Download the plug-in installation file from the publisher's website ([http://www.pushok.com](http://www.pushok.com)), run it, and follow the installation steps.

2. On the **Project** menu of Authentic Desktop, click **Change Source Control**, and make sure **PushOk GITSCC** is selected as source control provider. If you do not see **Push Ok GITSCC** in the list of providers, it is likely that the installation of the plug-in was not successful. In this case, check the publisher’s documentation for a solution.

3. When a dialog box prompts you to register the plug-in, click **Registration** and follow the wizard steps to complete the registration process.

![Change Source Control dialog box](image)
8.7.2 Adding a Project to Git Source Control

You can save Authentic Desktop projects as Git repositories. The structure of files or folders that you add to the project would then correspond to the structure of the Git repository.

To add a project to Git source control:

1. Make sure that PushOK GIT SCC Plug-in is set as source control provider (see Enabling Git Source Control with GIT SCC Plug-in).
2. Create a new project using the menu command Project | Create Project.
3. Save the project to a local folder, for example C:\MyRepo\Project.spp.
4. On the Project menu, under Source Control, click Add to Source Control.

5. Click OK.

6. Enter the text of your commit message, and click OK.

You can now start adding files and folders to your project. Note that all project files and folders must be under the root folder of the project. For example, if the project was created in the C: \MyRepo folder, then only files under C:\MyRepo should be added to the project. Otherwise, if you attempt to add to your project files that are outside the project root folder, a warning message is displayed:
Files should only be added to a location below the binding root of your project (C:\MyRepo).

Don't show this dialog again (Always add files even if they are outside the binding root)!
8.7.3 Cloning a Project from Git Source Control

Projects that have been previously added to Git source control (see Adding a Project to Git Source Control) can be opened from the Git repository as follows:

1. Make sure that PushOK GIT SCC Plug-in is set as source control provider (see Enabling Git Source Control with GIT SCC Plug-in).
2. On the Project menu, click Source Control | Open from Source Control.
3. Enter the path or the URL of the source repository. Click Check to verify the validity of the path or URL.
4. Under Local Path, enter the path to local folder where you want the project to be created, and click Next. If the local folder exists (even if it is empty), the following dialog box opens:

   ![Open from Source Control Wizard]

5. Click Yes to confirm, and then click Next.
6. Follow the remaining wizard steps, as required by your specific case.
7. When the wizard completes, a Browse dialog box appears, asking you to open the Authentic Desktop Project (*.spp) file. Select the project file to load the project contents into Authentic Desktop.
9  Authentic Desktop in Visual Studio

Authentic Desktop can be integrated into the Microsoft Visual Studio IDE versions 2010/2012/2013/2015/2017. This unifies the best of both worlds, integrating XML editing capabilities with the advanced development environment of Visual Studio.

In this section, we describe:

- The broad installation process and the integration of the Authentic Desktop plugin in Visual Studio.
- Differences between the Visual Studio version and the standalone version.
9.1 Installing the Authentic Desktop Plugin for Visual Studio

To install the Authentic Desktop Plugin for Visual Studio, you need to do the following:

2. Install Authentic Desktop (Enterprise or Professional Edition).
3. Download and run the Authentic Desktop integration package for Microsoft Visual Studio. This package is available on the Authentic Desktop (Enterprise and Professional Editions) download page at www.altova.com. (Please note: You must use the integration package corresponding to your Authentic Desktop version (current version is 2019).)

Once the integration package has been installed, you will be able to use Authentic Desktop in the Visual Studio environment.

How to enable the plug-in

If the plug-in was not automatically enabled during the installation process, do the following:

1. Navigate to the directory where the Visual Studio IDE executable was installed, for example in C:\Program Files\MS Visual Studio\Common7\IDE
2. Enter the following command on the command-line devenv.exe /setup.
3. Wait for the process to terminate normally before starting to use the application within Visual Studio.
9.2 Differences with Standalone Version

This section lists the ways in which the Visual Studio versions differ from the standalone versions of Authentic Desktop.

Entry helpers (Tool windows in Visual Studio)

The entry helpers of Authentic Desktop are available as Tool windows in Visual Studio. The following points about them should be noted. (For a description of entry helpers and the Authentic Desktop GUI, see the section, Introduction.)

- You can drag entry helper windows to any position in the development environment.
- Right-clicking an entry helper tab allows you to further customize your interface. Entry helper configuration options are: dockable, hide, floating, and auto-hide.

Authentic Desktop commands as Visual Studio commands

Some Authentic Desktop commands are present as Visual Studio commands in the Visual Studio GUI. These are:

- **Undo, Redo**: These Visual Studio commands affect all actions in the Visual Studio development environment.
- **Projects**: Authentic Desktop projects are handled as Visual Studio projects.
- **Customize Toolbars, Customize Commands**: The Toolbars and Commands tabs in the Customize dialog (Tools | Customize) contain both Visual Studio commands as well as Authentic Desktop commands.
- **Views**: In the View menu, the command Authentic Desktop contains options to toggle on entry helper windows and other sidebars, and to switch between editing views, and toggle certain editing guides on and off.
- **Authentic Desktop Help**: This Authentic Desktop menu appears as a submenu in Visual Studio's Help menu.

Additional Notes

Some additional notes and tips are given below:

- To edit an XML file with the Authentic plugin, select the File | Open command. Then, in the File Open dialog, use the Open With option to select the Authentic plugin.
10 Authentic Desktop in Eclipse

Eclipse is an open source framework that integrates different types of applications delivered in the form of plugins.

The Authentic Plugin for Eclipse enables you to access the functionality of Authentic Desktop from within the Eclipse 4.7 / 4.8 / 4.9 / 4.10 Platform. It is available on Windows platforms. In this section, we describe how to install the Authentic Plugin for Eclipse and how to set up the Authentic perspective. After you have done this, components of the Authentic Desktop GUI and Authentic Desktop menu commands will be available within the Eclipse GUI.
10.1 Installing the Authentic Desktop Plugin for Eclipse

Before installing the Authentic Desktop Plugin for Eclipse, ensure that the following are already installed:

- Authentic Desktop Enterprise or Professional Edition.
- Java SE Runtime Environment 6.0 (JRE 6.0) or higher, which is required for Eclipse. See the Eclipse website for more information. Install a 32-bit or 64-bit JRE to match your version of Authentic Desktop (32-bit or 64-bit).
- Eclipse Platform 4.7 / 4.8 / 4.9 / 4.10. Install a 32-bit or 64-bit Eclipse to match your version of Authentic Desktop (32-bit or 64-bit).

After these have been installed, you can install the Authentic Desktop Plugin for Eclipse, which is contained in the Authentic Desktop Integration Package (see below).

Authentic Desktop Integration Package

The Authentic Desktop Plugin for Eclipse is contained in the Authentic Desktop Integration Package and is installed during the installation of the Authentic Desktop Integration Package. Install as follows:

1. Ensure that Authentic Desktop (Enterprise or Professional Edition), JRE, and Eclipse are already installed (see above).
2. From the Components Download page of the Altova website, download and install the Authentic Desktop Integration Package. There are two important steps during the installation; these are described in Steps 3 and 4 below.
3. During installation of the Authentic Desktop Integration Package, a dialog will appear asking whether you wish to install the Authentic Desktop Plugin for Eclipse (see screenshot below). Check the option and then click Next.
4. In the next dialog ((Eclipse) Installation Location, screenshot below), you can choose whether the Install Wizard should integrate the Authentic Desktop Plugin into Eclipse during the installation (the *Automatically* option) or whether you will integrate the Authentic Desktop Plugin into Eclipse (via the Eclipse GUI) at a later time.
We recommend that you let the Installation Wizard do the integration. Do this by checking the *Automatically* option and then browsing for the folder in which the Eclipse executable (`eclipse.exe`) is located. Click **Next** when done. If you choose to manually integrate Authentic Desktop Plugin for Eclipse in Eclipse, select the *Manually* option (screenshot below). See the section below for instructions about how to manually integrate from within Eclipse.
5. Complete the installation. If you set up automatic integration, the Authentic Desktop Plugin for Eclipse will be integrated in Eclipse and will be available when you start Eclipse the next time.

Manually integrating the Authentic Desktop plugin in Eclipse
To manually integrate the Authentic Desktop Plugin for Eclipse, do the following:

1. In Eclipse, click the menu command Help | Install New Software.
2. In the Install dialog that pops up (screenshot below), click the Add button.
3. In the Add Repository dialog that pops up (screenshot below), click the Local button.

4. Browse for the folder `c:\Program Files\Altova\Common2019\eclipse\UpdateSite`, and select it. Provide a name for the site (such as 'Altova'), and click OK.

5. Repeat Steps 2 to 4, this time selecting the folder `c:\Program Files\Altova\Authentic Desktop2019\eclipse\UpdateSite`, and providing a name such as 'Altova Authentic Desktop'.
6. In the Work With combo box of the Install dialog, select the option — All Available Sites — (see screenshot below). This causes all available plugins to be displayed in the pane below. Check the top-level check box of the Altova category folder (see screenshot below). Then click the Next button.

7. An Install Details screen allows you to review the items to be installed. Click Next to proceed.

8. In the Review Licenses screen that appears, select I accept the terms of the license agreement. (No license agreement (additional to your Authentic Desktop Enterprise or Professional Edition license) is required for the Authentic Desktop plugin.) Then click Finish to complete the installation.
If there are problems with the plug-in (missing icons, for example), start Eclipse via the command line with the `-clean` flag.

**Currently installed version**

To check the currently installed version of the Authentic Desktop Plugin for Eclipse, select the Eclipse menu option **Help | About Eclipse**. Then select the Authentic Desktop icon.
10.2 Authentic Desktop Entry Points in Eclipse

The following entry points in Eclipse can be used to access Authentic Desktop functionality:

- **Authentic Desktop Perspective**, which provides Authentic Desktop's GUI features within the Eclipse GUI.
- **Authentic Desktop menu and toolbar**

**Authentic Desktop Perspective**

In Eclipse, a perspective is a configured GUI view with functionality from various applications. When the Authentic Desktop Plugin for Eclipse is integrated in Eclipse, a default Authentic Desktop perspective is automatically created. This perspective is a GUI that includes Authentic Desktop's GUI elements: its editing views, menus, entry helpers, and other sidebars.

When a file having a filetype associated with Authentic Desktop is opened (`.xml`), this file can be edited in the Authentic Desktop perspective. Similarly, a file of another filetype can be opened in another perspective in Eclipse. Additionally, for any active file, you can switch the perspective, thus allowing you to edit or process that file in another environment. There are therefore two main advantage of perspectives:

1. Being able to quickly change the working environment of the active file, and
2. Being able to switch between files without having to open a new development environment (the associated environment is available in a perspective)

**Working with the Authentic Desktop perspective involves the following:**

- Switching to the Authentic Desktop perspective.
- Setting preferences for the Authentic Desktop perspective.
- Customizing the Authentic Desktop perspective.

**Switching to the Authentic Desktop perspective**

In Eclipse, select the command **Window | Open Perspective | Other**. In the dialog that pops up (screenshot below), select **Authentic Desktop**, and click **OK**.
The empty window or the active document will now have the Authentic Desktop perspective. This is how the user switches the perspective via the menu. To access a perspective faster from another perspective, the required perspective can be listed in the **Open Perspective** submenu, above the **Other** item; this setting is in the customization dialog (*see further below*).

Perspectives can also be switched when a file is opened or made active. The perspective of the application associated with a file's filetype will be automatically opened when that file is opened for the first time. Before the perspective is switched, a dialog appears asking whether you wish to have the default perspective automatically associated with this filetype. Check the **Do Not Ask Again** option if you wish to associate the perspective with the filetype without having to be prompted each time a file of this filetype is opened and then click **OK**.

### Setting preferences for the Authentic Desktop perspective

The preferences of a perspective include: (i) a setting to automatically change the perspective when a file of an associated filetype is opened (*see above*), and (ii) options for including or excluding individual Authentic Desktop toolbars. In the list of perspectives in the left pane, select Authentic Desktop, then select the required preferences. Finish by clicking **OK**.

### Customizing the Authentic Desktop perspective

The customization options enable you to determine what shortcuts and commands are included in the perspective. To access the Customize Perspective dialog of a perspective (*screenshot below shows dialog for the Authentic Desktop perspective*), make the perspective active (in this case the Authentic Desktop perspective), and select the command **Window | Customize Perspective**.

In the Tool Bar Visibility and Menu Visibility tabs, you can specify which toolbars and menus are to be displayed. In the Command Groups Availability tab, you can add command groups to their...
parent menus and to the toolbar. If you wish to enable a command group, check its check box. In the Shortcuts tab of the Customize Perspective dialog, you can set shortcuts for submenus. Select the required submenu in the Submenus combo box. Then select a shortcut category, and check the shortcuts you wish to include for the perspective. Click OK to complete the customization and for the changes to take effect.

**Authentic Desktop menu and toolbar**

The Authentic Desktop menu contains commands that are relevant even if a document type recognized by Authentic Desktop is not currently open in Eclipse. In the standalone version of Authentic Desktop, some of these commands are in the File menu.

The Authentic Desktop toolbar contains the following buttons (*screenshot below*).

![Screenshot of Authentic Desktop toolbar](image)

These are for, from left: (i) opening the Authentic Desktop Help, and (ii) accessing Authentic Desktop commands (as an alternative to accessing them from the Authentic Desktop menu).
Menu Commands

The **User Reference** section contains a complete description of all Authentic Desktop menu commands and explains their use. We have tried to be comprehensive. If, however, you have questions which are not covered in the User Reference or other parts of this documentation, please look up the FAQs and Discussion Forums on the Altova website. If you cannot find a suitable answer at these locations, please do not hesitate to contact the [Altova Support Center](https://www.altova.com/support).

Standard Windows commands, such as *(Open, Save, Cut, Copy and Paste)* are in the **File** and **Edit** menus. These menus additionally contain XML- and Internet-related commands.
11.1 File Menu

The File menu contains commands for file operations, ordered as in most Windows applications. In addition to the standard New, Open, Save, Print, Print Setup, and Exit commands, Authentic Desktop also offers XML-specific and application-specific commands.
11.1.1 New

This section:

- Icon and shortcut
- Description

Icon and shortcut

<table>
<thead>
<tr>
<th>Icon:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortcut:</td>
<td>Ctrl+N</td>
</tr>
</tbody>
</table>

Description

This command enables you to open a new XML document template in Authentic View. The XML document template is based on a StyleVision Power Stylesheet (.sps file), and is opened by selecting the StyleVision Power Stylesheet (SPS file) in the Create New Document dialog (screenshot below). On selecting an SPS and clicking OK, the XML document template defined for that SPS file is opened in Authentic View.

The Create New Document dialog offers a choice of XML document templates that are based on popular DTDs or schemas. Alternatively, you can browse for a custom-made SPS file that has a Template XML File assigned to it. SPS files are created using Altova StyleVision, an application that enables you to design XML document templates based on a DTD or XML Schema. After designing the required SPS in StyleVision, an XML file is assigned (in StyleVision) as a Template XML File to the SPS. The data in this XML file provides the starting data of the new document template that is opened in the Authentic View of Authentic Desktop.

The new XML document template will therefore have the documentation presentation properties defined in the SPS and the data of the XML file that was selected as the Template XML File. The Authentic View user can now edit the XML document template in a graphical WYSIWYG interface, and save it as an XML document.
11.1.2 Open

Icon and shortcut

<table>
<thead>
<tr>
<th>Icon:</th>
<th><img src="image1" alt="Image" /></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortcut:</td>
<td>Ctrl+O</td>
</tr>
</tbody>
</table>

Description

The Open command pops up the familiar Windows Open dialog, and allows you to open any XML-related document or text document. In the Open dialog, you can select more than one file to open. Use the Files of Type combo box to restrict the kind of files displayed in the dialog box. (The list of available file types can be configured in the File Types section of the Options dialog ([Tools | Options](Tools | Options)).) When an XML file is opened, it is checked for well-formedness. If the file is not well-formed, you will get a file-not-well-formed error. Fix the error and select the menu command [XML | Check Well-Formedness (F7)](XML | Check Well-Formedness (F7)) to recheck. If you have opted for automatic validation upon opening and the file is invalid, you will get an error message. Fix the error and select the menu command [XML | Validate XML (F8)](XML | Validate XML (F8)) to revalidate.

Selecting and saving files via URLs and Global Resources

In several File Open and File Save dialogs, you can choose to select the required file or save a file via a URL or a global resource (see screenshot below). Click [Switch to URL](Switch to URL) or [Switch to Global Resource](Switch to Global Resource) to go to one of these selection processes.
Selecting files via URLs

To select a file via a URL (either for opening or saving), do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open or Save dialog (the screenshot below shows the Open dialog).
2. Enter the URL you want to access in the **Server URL** field *(screenshot above)*. If the server is a Microsoft® SharePoint® Server, check the **Microsoft® SharePoint® Server** check box. See the Microsoft® SharePoint® Server Notes below for further information about working with files on this type of server.

3. If the server is password protected, enter your User-ID and password in the **User** and **Password** fields.

4. Click **Browse** to view and navigate the directory structure of the server.

5. In the folder tree, browse for the file you want to load and click it.
The file URL appears in the File URL field (see screenshot above). The **Open** or **Save** button only becomes active at this point.

6. Click **Open** to load the file or **Save** to save it.

**Note the following:**

- The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.
- To give you more control over the loading process when opening a file, you can choose to load the file through the local cache or a proxy server (which considerably speeds up the process if the file has been loaded before). Alternatively, you may want to reload the file if you are working, say, with an electronic publishing or database system; select the **Reload** option in this case.

**Microsoft® SharePoint® Server Notes**

Note the following points about files on Microsoft® SharePoint® Servers:

- In the directory structure that appears in the Available Files pane (screenshot below), file icons have symbols that indicate the check-in/check-out status of files.
Right-clicking a file pops up a context menu containing commands available for that file (screenshot above).

- The various file icons are shown below:

  - Checked in. Available for check-out.
  - Checked out by another user. Not available for check-out.
  - Checked out locally. Can be edited and checked-in.

- After you check out a file, you can edit it in your Altova application and save it using File | Save (Ctrl+S).
- You can check-in the edited file via the context menu in the Open URL dialog (see screenshot above), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (screenshot below).

- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft®
SharePoint® Server will be: **Check In** and **Undo Check Out**.

- Opening and saving files via Global Resources
  To open or save a file via a global resource, click **Switch to Global Resource**. This pops up a dialog in which you can select the global resource. These dialogs are described in the section, **Using Global Resources**. For a general description of Global Resources, see the **Global Resources** section in this documentation.
11.1.3 Reload

Icon

Description
Reloads any open documents that have modified outside Authentic Desktop. If one or more documents is modified outside Authentic Desktop, a prompt appears asking whether you wish to reload the modified document/s. If you choose to reload, then any changes you may have made to the file since the last time it was saved will be lost.
11.1.4 Encoding

The Encoding command lets you: (i) view the current encoding of the active document (XML or non-XML), and (ii) select a different encoding with which the active document will be saved the next time.

In XML documents, if you select a different encoding than the one currently in use, the encoding attribute in the XML declaration will be modified accordingly. For two-byte and four-byte character encodings (UTF-16, UCS-2, and UCS-4) you can also specify the byte-order to be used for the file. Another way to change the encoding of an XML document is to directly edit the encoding attribute of the document's XML declaration. Default encodings for existing and new XML and non-XML documents can be set in the Encoding section of the Options dialog.

**Note:** When saving a document, Authentic Desktop automatically checks the encoding specification and enables you to select the required encoding via the Encoding dialog. If your document contains characters that cannot be represented in the selected encoding and you attempt to save the file, you will get a warning message to this effect.
11.1.5 Close, Close All, Close All But Active

Close
The Close command closes the active document window. If the file was modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

Close All
The Close All command closes all open document windows. If any document has been modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

Close All But Active
The Close All But Active command closes all open document windows except the active document window. If any document has been modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.
11.1.6 Save, Save As, Save All

Icons and shortcuts

<table>
<thead>
<tr>
<th>Command</th>
<th>Icon</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>![Icon]</td>
<td>Ctrl+S</td>
</tr>
<tr>
<td>Save All</td>
<td>![Icon]</td>
<td></td>
</tr>
</tbody>
</table>

Save

The **Save** command (Ctrl+S) saves the contents of the active document to the file from which it has been opened. When saving a document, the file is automatically checked for well-formedness. The file will also be validated automatically if this option has been set in the File section of the Options dialog (Tools | Options). The XML declaration is also checked for the encoding specification, and this encoding is applied to the document when the file is saved.

Save As

The **Save As** command pops up the familiar Windows Save As dialog box, in which you enter the name and location of the file you wish to save the active file as. The same checks and validations occur as for the **Save** command.

Save All

The **Save All** command saves all modifications that have been made to any open documents. The command is useful if you edit multiple documents simultaneously. If a document has not been saved before (for example, after being newly created), the Save As dialog box is presented for that document.

Selecting and saving files via URLs and Global Resources

In several File Open and File Save dialogs, you can choose to select the required file or save a file via a URL or a global resource (see screenshot below). Click **Switch to URL** or **Switch to Global Resource** to go to one of these selection processes.
Selecting files via URLs

To select a file via a URL (either for opening or saving), do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open or Save dialog (the screenshot below shows the Open dialog).
2. Enter the URL you want to access in the Server URL field (screenshot above). If the server is a Microsoft® SharePoint® Server, check the Microsoft® SharePoint® Server check box. See the Microsoft® SharePoint® Server Notes below for further information about working with files on this type of server.

3. If the server is password protected, enter your User-ID and password in the User and Password fields.

4. Click Browse to view and navigate the directory structure of the server.

5. In the folder tree, browse for the file you want to load and click it.
6. Click **Open** to load the file or **Save** to save it.

**Note the following:**

- The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.
- To give you more control over the loading process when opening a file, you can choose to load the file through the local cache or a proxy server (which considerably speeds up the process if the file has been loaded before). Alternatively, you may want to reload the file if you are working, say, with an electronic publishing or database system; select the **Reload** option in this case.

**Microsoft® SharePoint® Server Notes**

Note the following points about files on Microsoft® SharePoint® Servers:

- In the directory structure that appears in the Available Files pane (see screenshot below), file icons have symbols that indicate the check-in/check-out status of files.
Right-clicking a file pops up a context menu containing commands available for that file (screenshot above).

- The various file icons are shown below:

  - Checked in. Available for check-out.
  - Checked out by another user. Not available for check-out.
  - Checked out locally. Can be edited and checked-in.

- After you check out a file, you can edit it in your Altova application and save it using File | Save (Ctrl+S).
- You can check-in the edited file via the context menu in the Open URL dialog (see screenshot above), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (screenshot below).

- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft®
SharePoint® Server will be: **Check In** and **Undo Check Out**.

**Opening and saving files via Global Resources**

To open or save a file via a global resources, click **Switch to Global Resource**. This pops up a dialog in which you can select the global resource. These dialogs are described in the section, **Using Global Resources**. For a general description of Global Resources, see the **Global Resources** section in this documentation.
11.1.7 Send by Mail

Icon

Icon: 📬

Description

The **Send by Mail** command lets you send XML document/s or selections from an XML document by e-mail. Depending on what kind it is, a document or selection can be sent as an attachment, content, or as a link. See the table below for details.

<table>
<thead>
<tr>
<th>What can be sent</th>
<th>How it can be sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active XML document</td>
<td>As e-mail attachment</td>
</tr>
<tr>
<td>Selection in active XML document</td>
<td>As e-mail attachment or e-mail content</td>
</tr>
<tr>
<td>One or more files in Project window</td>
<td>As e-mail attachment</td>
</tr>
<tr>
<td>One or more URLs in Project window</td>
<td>As e-mail attachment or link</td>
</tr>
</tbody>
</table>

When the **Send by Mail** command is invoked on a selection in the active XML document, the Send by Mail dialog (screenshot below) pops up and offers the sending options shown in the screenshot. If the **Send by Mail** command is invoked with no text selected in the active file, then the **Whole File** radio button (refer to screenshot above) is the only option that is enabled; the other options are disabled.

Since files sent from the Project window are always sent as e-mail attachments only, the Send by Email dialog is skipped and an e-mail is opened that has the selected file/s as attachments. URLs in the project window can be sent as an attachment or as a link (see screenshot below). Select how the URL is to be sent and click **OK**.
11.1.8 Print

Icon and shortcut

<table>
<thead>
<tr>
<th>Icon:</th>
<th>📑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortcut:</td>
<td>Ctrl+P</td>
</tr>
</tbody>
</table>

Description

The Print command opens the Print dialog box, in which you can select printing options for printing the currently active document.
11.1.9 Print Preview, Print Setup

Print Preview

The Print Preview command clicked in Text View and Browser View opens a print preview of the currently active document.

In Print Preview mode, the Print Preview toolbar at top left of the preview window provides print- and preview-related options. The preview can be magnified or miniaturized using the the Zoom In and Zoom Out buttons. When the page magnification is such that an entire page length fits in a preview window, then the One Page / Two Page button toggles the preview to one or two pages at a time. The Next Page and Previous Page buttons can be used to navigate among the pages. The toolbar also contains buttons to print all pages and to close the preview window.

Note: To enable background colors and images in Print Preview, do the following: (i) In the Tools menu of Internet Explorer, click Internet Options, and then click the Advanced tab; (ii) In the Settings box, under Printing, select the Print background colors and images check box, and (iii) Then click OK.

Print Setup

The Print Setup command, displays the printer-specific Print Setup dialog box, in which you specify such printer settings as paper format and page orientation. These settings are applied to all subsequent print jobs.
11.1.10 Recent Files, Exit

Recent Files
At the bottom of the File menu is a list of the nine most recently used files, with the most recently opened file shown at the top of the list. You can open any of these files by clicking its name. To open a file in the list using the keyboard, press Alt+F to open the File menu, and then press the number of the file you want to open.

Exit
Quits Authentic Desktop. If you have any open files with unsaved changes, you are prompted to save these changes. Authentic Desktop also saves modifications to program settings and information about the most recently used files.
11.2 Edit Menu

The **Edit** menu contains commands for editing documents in Authentic Desktop. These include the familiar **Undo**, **Redo**, **Cut**, **Copy**, **Paste**, **Delete**, **Select All**, **Find**, **Find Next** and **Replace** commands.
11.2.1 Undo, Redo

Icons and shortcuts

<table>
<thead>
<tr>
<th>Command</th>
<th>Icon</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>![Undo Icon]</td>
<td>Ctrl+Z</td>
</tr>
<tr>
<td>Redo</td>
<td>![Redo Icon]</td>
<td>Ctrl+Y</td>
</tr>
</tbody>
</table>

Undo

The **Undo** command contains support for unlimited levels of Undo. Every action can be undone and it is possible to undo one command after another. The Undo history is retained after using the **Save** command, enabling you go back to the state the document was in before you saved your changes. You can step backwards and forwards through this history using the **Undo** and **Redo** commands (see **Redo** command below).

Redo

The **Redo** command allows you to redo previously undone commands, thereby giving you a complete history of work completed. You can step backwards and forwards through this history using the **Undo** and **Redo** commands.
11.2.2 Cut, Copy, Paste, Delete

Icons and shortcuts

<table>
<thead>
<tr>
<th>Command</th>
<th>Icon</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td>![Cut Icon]</td>
<td>Ctrl+X or Shift+Del</td>
</tr>
<tr>
<td>Copy</td>
<td>![Copy Icon]</td>
<td>Ctrl+C</td>
</tr>
<tr>
<td>Paste</td>
<td>![Paste Icon]</td>
<td>Ctrl+V</td>
</tr>
<tr>
<td>Delete</td>
<td>![Delete Icon]</td>
<td>Del</td>
</tr>
</tbody>
</table>

Cut
The Cut command copies the selected text or items to the clipboard and deletes them from their present location.

Copy
The Copy command copies the selected text or items to the clipboard. This can be used to duplicate data within Authentic Desktop or to move data to another application.

Paste
The Paste command inserts the contents of the clipboard at the current cursor position.

Delete
The Delete command deletes the currently selected text or items without placing them in the clipboard.
11.2.3 Select All

The Select All command (**Ctrl+A**) selects the contents of the entire document.
11.2.4 Find, Find Next

Icons and shortcuts

<table>
<thead>
<tr>
<th>Command</th>
<th>Icon</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>![Find Icon]</td>
<td>Ctrl+F</td>
</tr>
<tr>
<td>Find Next</td>
<td>![Find Next Icon]</td>
<td>F3</td>
</tr>
</tbody>
</table>

Find

The Find command displays the Find dialog, in which you can specify the string you want to find and other options for the search. To find text, enter the text in the Find field or use the combo box to select from one of the last 10 search criteria, and then specify the options for the search.

The Find and Find Next commands can also be used to find file and folder names when a project is selected in the Project window.

Find Next

The Find Next command repeats the last Find command. It searches for the next occurrence of the input text.
11.2.5 Replace

Icons and shortcuts

<table>
<thead>
<tr>
<th>Command</th>
<th>Icon</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace</td>
<td><img src="image" alt="icon" /></td>
<td>Ctrl+H</td>
</tr>
</tbody>
</table>

Description

The **Replace** command enables you to find and replace one text string with another. It features the same options as the **Find** command. You can replace each item individually, or you can use the **Replace All** button to perform a global find-and-replace operation.
11.3 Project Menu

Authentic Desktop uses the familiar tree view to manage multiple files or URLs in XML projects. Files and URLs can be grouped into folders by common extension or any arbitrary criteria, allowing for easy structuring and batch manipulation.

Please note: Most project-related commands are also available in the context menu, which appears when you right-click any item in the project window.

Absolute and relative paths

Each project is saved as a project file, and has the .spp extension. These files are actually XML documents that you can edit like any regular XML File. In the project file, absolute paths are used for files/folders on the same level or higher, and relative paths for files/folders in the current folder or in sub-folders. For example, if your directory structure looks like this:

```
|---Folder1
|   |---Folder2
|   |   |---Folder3
```
If your .spp file is located in Folder3, then references to files in Folder1 and Folder2 will look something like this:

- c:\Folder1\NameOfFile.ext
- c:\Folder1\Folder2\NameOfFile.ext

References to files in Folder3 and Folder4 will look something like this:

- .\NameOfFile.ext
- .\Folder4\NameOfFile.ext

If you wish to ensure that all paths will be relative, save the .spp files in the root directory of your working disk.

**Drag-and-drop**

In the Project window, a folder can be dragged to another folder or to another location within the same folder. A file can be dragged to another folder, but cannot be moved within the same folder (within which files are arranged alphabetically). Additionally, files and folders can be dragged from Windows File Explorer to the Project window.

**Find in project**

You can search for project files and folders using their names or a part of their name. If the search is successful, files or folders that are located are highlighted one by one.

To start a search, select the project folder in the Project sidebar that you wish to search, then select the command **Edit | Find** (or the shortcut **Ctrl+F**). In the Find dialog that pops up (screenshot below) enter the text string you wish to search for and select or deselect the search options (explained below) according to your requirements.

The following search options are available:

- Whole-word matching is more restricted since the entire string must match an entire word in the file or folder name. In file names, the parts before and after the dot (without the dot) are each treated as a word.
- It can be specified that casing in the search string must exactly match the text string in the file or folder name.
- Folder names can be included in the search. Otherwise, only file names are searched.
- **External folders** can be included or excluded from the search. External folders are actual folders on the system or network, as opposed to project folders, which are created within the project and not on the system.

If the search is successful, the first matching item is highlighted in the Project sidebar. You can then browse through all the returned matching items by clicking the **Find Next** and **Find Prev** buttons in the Find dialog.

**Refreshing projects**
If a change is made to an external folder, this change will not be reflected in the Project Window till the project is refreshed.

**Global resources in the context menu**
When you right-click a folder in the Project window, in the context menu that appears, you can select the **Add Global Resource** menu item to add a **global resource**. The menu command itself pops up the Choose Global Resource dialog, which lists all the file-type and folder-type global resources in the currently active Global Resources XML File. Select the required global resource, and it will be added to the selected project folder.

**Projects and source control providers**
If you intend to add an Authentic Desktop project to a source control repository, please ensure that the project files position in the hierarchical file system structure is one which enables you to add files only from below it (taking the root directory to be the top of the directory tree).

In other words, the directory where the **project file** is located, essentially represents the **root directory** of the project within the source control repository. Files added from above it (the project root directory) will be added to the Authentic Desktop project, but their location in the repository may be an unexpected one—if they are allowed to be placed there at all.

For example, given the directory structure show above, if a project file is saved in **Folder3** and placed under source control:

- Files added to Folder1 may not be placed under source control,
- Files added to Folder2 are added to the root directory of the repository, instead of to the project folder, but are still under source control,
- Files located in Folder3 and Folder4 work as expected, and are placed under source control.
11.3.1 New Project

The New Project command creates a new project in Authentic Desktop. If you are currently working with another project, a prompt appears asking if you want to close all documents belonging to the current project.
11.3.2 Open Project

The Open Project... command opens an existing project in Authentic Desktop. If you are currently working with another project, the previous project is closed first.
11.3.3 Reload Project

The **Reload Project** command reloads the current project from disk. If you are working in a multi-user environment, it can sometimes become necessary to reload the project from disk, because other users might have made changes to the project.

**Please note:** Project files (.spp files) are actually XML documents that you can edit like any regular XML File.
11.3.4 Close Project

The **Close Project** command **closes** the active project. If the project has been modified, you will be asked whether you want to save the project first. When a project is modified in any way, an asterisk is added to the project name in the Project Window.
11.3.5 Save Project, Save Project As

The **Save Project** command **saves** the current project. You can also save a project by making the project window active and clicking the icon.

The **Save Project As** command **saves** the current project with a new name that you can enter when prompted for one.
11.3.6 **Source Control**

Your Altova application supports Microsoft SourceSafe and other compatible repositories. A list of supported systems is given in the section, Supported Source Control Systems. This section describes the commands in the Project | Source Control submenu, which are used to work with the source control system from within your Altova application.

**Overview of the Source Control feature**

The mechanism for placing files in an application project under source control is as follows:

1. In Authentic Desktop, an application project folder containing the files to be placed under source control is created. Typically, the application project folder will correspond to a local folder in which the project files are located. The path to the local folder is referred to as the local path.
2. In the source control system's database (also referred to as source control or repository), a folder is created that will contain the files to be placed under source control.
3. Application project files are added to source control using the command **Project | Source Control | Add to Source Control**.
4. Source control actions, such as checking in to, checking out from, and removing files from source control, can be carried out by using the commands in the **Project | Source Control** submenu. The commands in this submenu are listed in the sub-sections of this section.

**Note:** If you wish to change the current source control provider, this can be done in any of two ways: (i) via the Source Control options (Tools | Options | Source Control), or (ii) in the Change Source Control dialog (Project | Source Control | Change Source Control).

**Note:** Note that a source control project is not the same as an application project. Source control projects are directory-dependent, while Authentic Desktop projects are logical constructions without direct directory dependence.

For additional information, see the section, Source Control.

**Open from Source Control**

The **Open from Source Control** command creates a new application project from a project under source control.

Create the new project as follows:

1. Depending on the source control system used, it might be necessary, before you create a new project from source control, to make sure that no file from the project is checked out.
2. No project need be open in the application, but can be.
3. Select the command Project | Source Control | Open from Source Control.
4. The source control system that is currently set will pop up its verification and connection dialogs. Make the connection to the repository you want, that is, to the bound folder in the repository that corresponds to the local folder.
5. In the dialog that pops up (screenshot below), browse for the local folder to which the contents of the bound folder in the repository (that you have just connected to) must be copied. In the screenshot below the bound folder is called MyProject and is represented
6. Click OK. The contents of the bound folder (MyProject) will be copied to the local folder C:\M20130326, and a dialog pops up asking you to select the project file (.spp file) that is to be created as the new project.

7. Select the .spp file that will have been copied to the local folder. In our example, this will be MyProject.spp located in the C:\M20130326 folder. A new project named MyProject will be created in the application and will be displayed in the Project window. The project's files will be in the folder C:\M20130326.

Source control symbols
Files and the project folder display certain symbols, the meanings of which are given below.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Checked out by another user. Not available for check-out.</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Checked out locally. Can be edited and checked-in.</td>
</tr>
</tbody>
</table>

Enable Source Control
The Enable Source Control command allows you to enable or disable source control for an application project. Selecting this option on any file or folder, enables/disables source control for the whole project. After source control is enabled, the check in/out status of the various files are retrieved and displayed in the Project window.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td>Checked out by another user. Not available for check-out.</td>
</tr>
</tbody>
</table>
Checked out locally. Can be edited and checked-in.

Get Latest Version

The Get Latest Version command (in the Project | Source Control menu) retrieves and places the latest source control version of the selected file(s) in the working directory. The files are retrieved as read-only and are not checked out. This command works like the Get command, but does not display the Get dialog.

If the selected files are currently checked out, then the action taken will depend on how your source control system handles such a situation. Typically, the source control system will ask whether you wish to replace, merge with, or leave the checked-out file as it is.

Note: This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.

Get, Get Folders

The Get command (in the Project | Source Control menu) retrieves files from the repository as read-only files. (To be able to edit a file, you must check it out.) The Get dialog lists the files in the object (project or folder) on which the Get command was executed (see screenshot below). You can select the files to retrieve by checking them.

Note: The Get Folders command allows you to select individual sub-folders in the repository if this is allowed by your source control system.

You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking OK, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (screenshot below) asking whether you wish to reload the file/s (Reload button), close the file/s (Close), or retain the current view of the file (Cancel).
Advanced Get Options

The Advanced Get Options dialog (screenshot below) is accessed via the Advanced button in the Get dialog (see first screenshot in this section).

Here you can set options for (i) replacing writable files that are checked out, (ii) the timestamp, and (iii) whether the read-only property of the retrieved file should be changed so that it will be writable.

Check Out, Check In

After a project file has been placed under source control, it can be checked out or checked in by selecting the file (in the Project window) and clicking the respective command in the Project | Source Control menu: Check Out and Check In.

When a file is checked out, a copy from the repository is placed in the local folder. A file that is checked out can be edited. If a file that is under source control is not checked out, it cannot be edited. After a file has been edited, the changes can be saved to the repository by checking in the file. Even if the file is not saved, checking it in will save the changes to the repository. Whether a file is checked out or not is indicated with a tick or lock symbol in its icon.

Files and the project folder display certain symbols, the meanings of which are given below.

- Checked in. Available for check-out.
Selecting the project or a folder within the project, selects all files in the selected object. To select multiple objects (files and folders), press the Ctrl key while clicking the objects. The screenshot below shows a project that has been checked out. The file `QuickStart.css` has subsequently been checked in.

Saving and rejecting editing changes

Note that, when checking in a file, you can choose to leave the file checked out. What this does is save editing changes to the repository while continuing to keep the file checked out, which is useful if you wish to periodically save editing changes to the repository and then continue editing.

If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command `Project | Source Control | Undo Check Out`.

Checking out

The Check Out dialog (screenshot below) allows you: (i) to select the files to check out, and (ii) to select whether the repository version or the local version should be checked out.
Checking in

The Check In dialog (*screenshot below*) allows you: (i) to select the files to check in, and (ii) if you wish, to keep the file checked out.

**Note:** In both dialogs (Check Out and Check In), multiple files appear if the selected object (project or project folder/s) contain multiple files.

**Undo Check Out**

If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command **Project | Source Control | Undo Check Out**.

Files and the project folder display certain symbols, the meanings of which are given below.
Add to Source Control

After a project has been added to source control, you can add files either singly or in groups to source control. Select the file in the Project window and then click the command **Project | Source Control | Add to Source Control**. To select multiple files, keep the **Ctrl** key pressed while clicking on the files you wish to add. Running the command on a (green) project folder *(see screenshot below)* adds all files in the folder and its sub-folders to source control.

![Project window with MyProject directory and its contents](image)

When files are added to source control, the local folder hierarchy is replicated in the repository (not the project folder hierarchy). So, if a file is in a sub-folder X levels deep in the local folder, then the file's parent folder and all other ancestor folders are automatically created in the repository.

When the first file from a project is added to source control, the correct bindings are created in the repository and the project file (`.spp` file) is added automatically. For more details, see the section **Add to Source Control**.

Source control symbols

Files and the project folder display certain symbols, the meanings of which are given below.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Check-in" /></td>
<td>Checked in. Available for check-out.</td>
</tr>
<tr>
<td><img src="image" alt="Check-out by another user" /></td>
<td>Checked out by another user. Not available for check-out.</td>
</tr>
<tr>
<td><img src="image" alt="Check-out locally" /></td>
<td>Checked out locally. Can be edited and checked-in.</td>
</tr>
</tbody>
</table>
Remove from Source Control

To remove a file from source control, select the file and click the command **Project | Source Control | Remove from Source Control**. You can also remove: (i) files in a project folder by executing the command on the folder, (ii) multiple files that you select while keeping the **Ctrl** key pressed, and (iii) the entire project by executing the command on the project.

Share from Source Control

The **Share from Source Control** command is supported when the source control system being used supports shares. You can share a file, so that it is available at multiple local locations. A change made to one of these local files will be reflected in all the other "shared" versions.

In the application's Project window first select the project (*highlighted in the screenshot below*). Then click the **Share from Source Control**.

The Share To [Folder] dialog (*screenshot below*) pops up.
To select the files to share, first choose, in the project tree in the right-hand pane, the folder in which the files are. The files in the chosen folder are displayed in the left hand pane. Select the file you wish to share (multiple files by pressing the Ctrl key and clicking the files you want to share). The selected file/s will be displayed in the Files to Share text box (at top left). Click Share and then Close to copy the selected file/s to the local share folder.

The share folder is noted in the name of the Share to [Folder] dialog. In the screenshot above it is the local folder (since the $ sign is the folder in the repository to which the local folder is bound). You can see and set the share folder in the Change Source Control dialog (screenshot below, Change Source Control) by changing the local path and server binding.
For more details about sharing using your source control system, see the source control system’s user documentation.

**Show History**

The **Show History** command activates the Show History feature of the active source control system. It displays the history of the file selected in the Project window. Select the project title to display the history of the project file (.spp file). You can view information about previous versions of a file and differences, as well as retrieve previous versions of the file.

The screenshot below shows the History dialog of the Visual SourceSafe source control system. It lists the various versions of the MyProject.spp file.

![History Dialog](image)

This History dialog provides various ways of comparing and getting specific versions of the file in question. Double-clicking an entry in the list opens the History Details dialog box for that file. The buttons in the History dialog provide the following functionality:

- **Close**: Closes this dialog box.
- **View**: Opens a dialog box in which you can select the type of file viewer.
- **Details**: Opens a dialog box in which you can see the properties of the currently active file.
- **Get**: Retrieves a previous file version and places it in the working directory.
- **Check Out**: Allows you to check out a previous version of the file.
- **Diff**: Opens the Difference options dialog box for differencing options between two file versions. Use Ctrl+Click to mark two file versions in this window, then click Diff to view the differences between them.
- **Pin**: Pins or unpins a version of the file, allowing you to define the specific file version to use when differencing two files.
- **Rollback**: Rolls back to the selected version of the file.
- **Report**: Generates a history report that you can send to a printer, file, or clipboard.
- **Help**: Opens the online help of the source control provider plugin.

### Show Differences

The **Show Differences** command is enabled when a file in the Project window is selected. To select the project file (.spp file), select the project title in the Project window. The **Show Differences** command starts the source control system's differencing tool so that differences between files can be directly checked from your Altova application.

The screenshot below shows the differencing tool of the Visual SourceSafe source control system.

![Difference Options](image)

The repository and local versions are shown by default in the *Compare* and *To* text fields respectively. You can browse for other files as follows:

1. From the **Browse** button dropdown list, select SourceSafe projects (for browsing repository files) or Windows folders (for browsing local folders).
2. Browse for the files you want and select them.

Select the options you want and click **OK** to run the check. The differencing results are displayed in a separate window. The screenshots below show the results of a check in two formats.
The screenshot above shows the Visual SourceSafe differencing result in Visual format (see Options dialog above), while the screenshot below shows the result in Unix format. In both, there are two differences, each of which is a change of the grade from C to B.

For a detailed description of how your source control system handles differencing, see the product's user documentation.

**Show Properties**

The Show Properties command displays the properties of the currently selected file (screenshot below). What properties are displayed depends on the source control system you are using. The screenshot below shows properties when Visual SourceSafe is the active source control system.

Note that this command is enabled only for single files.
For details, see the source control system’s user documentation.

**Refresh Status**

The **Refresh Status** command refreshes the status of all project files independent of their current status.

**Source Control Manager**

The **Source Control Manager** command starts your source control software with its native user interface.

**Change Source Control**

The current binding is what the active application project will use to connect to the source control database, so the current binding must be correct. By this is meant that the application project file (`.spp` file) must be in the local path folder and the bound folder on the repository must be the database where this project's files are stored. Typically the bound folder and its sub-structure will correspond with the local workspace folder and its sub-structure.

In the Change Source Control dialog (*screenshot below*), you can change the source control system (**SCC Provider**), the local folder (**Local Path**), and the repository binding (**Server Name** and **Server Binding**).
Only after unbinding the current binding can the settings be changed. Unbind the current binding with the **Unbind** button. All the settings are now editable.

<table>
<thead>
<tr>
<th>Change Source Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Path: C:\LocalWorkspace</td>
</tr>
<tr>
<td>Scc Provider: Microsoft Visual SourceSafe</td>
</tr>
<tr>
<td>Server Name: C:\VSSRepository</td>
</tr>
<tr>
<td>Server Binding: &quot;$/&quot;, AAAAAAAA</td>
</tr>
<tr>
<td>Logon ID: AAAA</td>
</tr>
<tr>
<td>Connected: ✓</td>
</tr>
</tbody>
</table>

Change source control settings as follows:

1. Use the **Browse** button to browse for the local folder and the **Select** button to select from among the installed source control systems.
2. After doing this you can bind the local folder to a repository database. Click the **Bind** button to do this. This pops up the connection dialog of your source control system.
3. If you have entered a **Logon ID**, this will be passed to the source control system; otherwise you might have to enter your logon details in the connection dialog.
4. Select the database in the repository that you wish to bind to this local folder. This setting might be spread over more than one dialog.
5. After the setting has been created, click **OK** in the Change Source Control dialog.
11.3.7 Add Files to Project

The Project | Add Files to Project command adds files to the current project. Use this command to add files to any folder in your project. You can either select a single file or any group of files (using Ctrl+ click) in the Open dialog box. If you are adding files to the project, they will be distributed among the respective folders based on the File Type Extensions defined in the Project Properties dialog box.
11.3.8 Add Global Resource to Project

The Project | Add Global Resource to Project command pops up the Choose Global Resource dialog, in which you can select a global resource of file or folder type to add to the project. If a file-type global resource is selected, then the file is added to the appropriate folder based on the File Type Extensions defined in the Project Properties dialog box. If a folder-type global resource is selected, that folder will be opened in a file-open dialog and you will be prompted to select a file; the selected file is added to the appropriate folder based on the File Type Extensions defined in the Project Properties dialog box. For a description of global resources, see the Global Resources section in this documentation.
11.3.9 Add URL to Project

The **Project | Add URL to Project** command adds a URL to the current project. URLs in a project cause the target object of the URL to be included in the project. Whenever a batch operation is performed on a URL or on a folder that contains a URL object, Authentic Desktop retrieves the document from the URL, and performs the requested operation.
11.3.10 Add Active File to Project

The Project | Add Active File to Project command adds the active file to the current project. If you have just opened a file from your hard disk or through an URL, you can add the file to the current project using this command.
11.3.11 Add Active And Related Files to Project

The Project | Add Active and Related Files to Project command adds the currently active XML document and all related files to the project. When working on an XML document that is based on a DTD or Schema, this command adds not only the XML document but also all related files (for example, the DTD and all external parsed entities to which the DTD refers) to the current project.

Please note: Files referenced by processing instructions (such as XSLT files) are not considered to be related files.
11.3.12 Add Project Folder to Project

The **Project | Add Project Folder to Project** command adds a new folder to the current project. Use this command to add a new folder to the current project or a sub-folder to a project folder. You can also access this command from the context-menu when you right-click on a folder in the project window.

**Note:** A project folder can be dragged and dropped into another project folder or to any other location in the project. Also, a folder can be dragged from Windows (File) Explorer and dropped into any project folder.

**Note:** Project folders are green, while [external folders](#) are yellow.
11.3.13 Add External Folder to Project

The **Project | Add External Folder to Project** command adds a new external folder to the current project. Use this command to add a local or network folder to the current project. You can also access this command from the context-menu when you right-click a folder in the project window.

**Note:** External folders are yellow, while **project folders** are green.

**Note:** Files contained in external folders cannot be placed under source control.

**Adding external folders to projects**

To add an external folder to the project:

1. Select the menu option **Project | Add External Folder to Project.**
2. Select the folder you want to include from the Browse for Folder dialog box, and click **OK** to confirm.

The selected folder now appears in the project window.

3. Click the plus icon to view the folder contents.
Filtering contents of folders

To filter the contents of the folder:

1. Right-click the local folder, and select the popup menu option **Properties**. This opens the Properties dialog box.

2. Click in the **File extensions** field and enter the file extensions of the file types you want to see. You can separate each file type with a **semicolon** to define multiple types (XML and Schema XSDs in this example).

3. Click **OK** to confirm.
The Project window now only shows the XML and XSD files of the tutorial folder.

Validating external folders
To validate and check an external folder for well-formedness:

1. Select the file types you want to see or check from the external folder,

2. Click the folder and click the Check well-formedness 🚚 or Validate 🚚 icon (hotkeys F7 or F8). All the files visible under the folder are checked. If a file is malformed or invalid, then this file is opened in the Main Window, allowing you to edit it.

3. Correct the error and run the validation process once more to recheck.

Updating a project folder
You might add or delete files in the local or network directory at any time. To update the folder view, right-click the external folder, and select the popup menu option Refresh external folder.

Deleting external folders and files in them
Select an external folder and press the Delete key to delete the folder from the Project window. Alternatively, right-click the external folder and select the Delete command. Each of these actions only deletes the external folder from the Project window. The external folder is not deleted from the hard disk or network.

To delete a file in an external folder, you have to delete it physically from the hard disk or network. To see the change in the project, refresh the external folder contents (right-click the external folder and select Refresh).

Note: An external folder can be dragged and dropped into a project folder or to any other location in the project (but not into another external folder). Also, an external folder can be dragged from Windows (File) Explorer and dropped into any location in the project window except into another external folder.
11.3.14 Add External Web Folder to Project

This command adds a new external web folder to the current project. You can also access this command from the context-menu when you right-click a folder in the project window. Note that files contained in external folders cannot be placed under source control.

Adding an external web folder to the project

To add an external web folder to the project, do the following:

1. Select the menu option **Project | Add External Web Folder to Project**. This opens the Add Web Folder to Project dialog box (screenshot below).

2. Click in the Server URL field and enter the URL of the server URL. If the server is a Microsoft® SharePoint® Server, check this option. See the *Folders on a Microsoft® SharePoint® Server* section below for further information about working with files on this type of server.

3. If the server is password-protected, enter your User ID and password in the User and Password fields.

4. Click **Browse** to connect to the server and view the available folders.
5. Click the folder you want to add to the project view. The Open button only becomes active once you do this. The URL of the folder now appears in the File URL field.

6. Click Open to add the folder to the project.

7. Click the plus icon to view the folder contents.
Filtering folder contents
To filter the contents of a folder, right-click the folder and select Properties from the context menu. In the Properties dialog that pops up, click in the File Extensions field and enter the file extensions of the file types you want to see (for example, XML and XSD files). Separate each file type with a semicolon (for example: xml; xsd; sps). The Project window will now show that folder only with files having the specified extension.

Validating and checking a folder for well-formedness
To check the files in a folder for well-formedness or to validate them, select the folder and then click the Check well-formedness or Validate icon (hotkeys F7 or F8, respectively). All the files that are visible in the folder are checked. If a file is malformed or invalid, then this file is opened in the main window, allowing you to edit it. Correct the error and restart the process to recheck the rest of the folder. Note that you can select discontinuous files in the folder by holding Ctrl and clicking the files singly. Only these files are then checked when you press F7 or F8.

Updating the contents of the project folder
Files may be added or deleted from the web folder at any time. To update the folder view, right-click the external folder and select the context menu option Refresh.

Deleting folders and files
Since it is the Web folder that has been added to the project, it is only the Web folder (and not files within it) that can be deleted from the project. You can delete a Web folder from a project, by either (i) right-clicking the folder and selecting Delete, or (ii) selecting the folder and pressing the Delete key. This only deletes the folder from the Project view; it does not delete anything on the web server.

Note: Right-clicking a single file and pressing the Delete key does not delete a file from the Project window. You have to delete it physically on the server and then refresh the contents of the external folder.

Folders on a Microsoft® SharePoint® Server
When a folder on a Microsoft® SharePoint® Server has been added to a project, files in the folder can be checked out and checked in via commands in the context menu of the file listing in the Project window (see screenshot below). To access these commands, right-click the file you wish to work with and select the command you want (Check Out, Check In, Undo Check Out).

The User ID and password can be saved in the properties of individual folders in the project, thereby enabling you to skip the verification process each time the server is accessed.
In the Project window (screenshot below), file icons have symbols that indicate the check-in/check-out status of files. The various file icons are shown below:

- Checked in. Available for check-out.
- Checked out by another user. Not available for check-out.
- Checked out locally. Can be edited and checked-in.

The following points should be noted:

- After you check out a file, you can edit it in your Altova application and save it using File | Save (Ctrl+S).
- You can check-in the edited file via the context menu in the Project window (see screenshot above), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (screenshot below).

- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft® SharePoint® Server will be: Check In and Undo Check Out.
11.3.15 Script Settings

A scripting project is assigned to an Authentic Desktop project as follows:

1. In the Authentic Desktop GUI, open the required application project.
2. Select the menu command **Project | Script Settings**. The Scripting dialog (screenshot below) opens.

3. Check the **Activate Project Scripts** check box and select the required scripting project (.asprj file). If you wish to run Auto-Macros when the Authentic Desktop project is loaded, check the **Run Auto-Macros** check box.
4. Click **OK** to finish.

**Note:** To deactivate (that is, unassign) the scripting project of an Authentic Desktop project, uncheck the **Activate Project Scripts** check box.
11.3.16 Properties

The **Project | Project Properties** command opens the Properties dialog (screenshot below) of the active project. If you right-click a folder in the Project window (as opposed to the project folder itself) and select **Properties**, the Properties dialog of that folder is opened. The dialog settings are described below.

**Note:** If your project file is under source control, a prompt appears asking if you want to check out the project (.spp) file. Click **OK** if you want to edit settings and be able to save them.

![Properties dialog](image.png)

**Settings**

**File extensions**
The **File Extensions** setting is enabled for individual folders, and not for the project folder. When a file is added to a project, it will be added to the folder on which its file extension has been defined. For example, say a file named `MyReport.xml` is added to the project. If `.xml` file extensions have been set on the `Invoices-EU` folder (as shown in the screenshot above), then `MyReport.xml` will be added to the `Invoices-EU` folder. If there is more than one folder to which you wish to add `XML` files, you can specify the `Use config:` field to specify the configuration file to use for file extension settings.
files, then you should add individual XML files directly to the folder (instead of to the project).

**User ID and password for external folders**
On external folders (including external Web folders), you can save the user ID and password that might be required for accessing the server.

**Validation**
The DTD, XML Schema, or JSON schema that should be used to validate the files in the current folder (or entire project if the properties are those of the project).

**XSL transformation of XML files**
The XSLT stylesheet to be used for XSLT transformation of XML files in the folder.

**XSL-FO transformation of XML files**
The XSLT stylesheet to transform XML files in the folder to XSL-FO.

**XSL transformation of XSL files**
The XML file to be used for XSLT transformation with XSLT files in the folder.

**XQuery transformation of XML files**
The XQuery file to be used for XQuery executions of XML files in the folder.

**Destination files of XSL transformation**
The destination directory of XSLT transformations, and, optionally, the file extension of the result document.

**Authentic View**
The `Use config` specifies the StyleVision Power Stylesheet (SPS file) to use for the Authentic View display of XML files in the folder. Note that the XML file must be valid against the same schema used for the SPS.

**Notes about project properties**
Note the following points about precedence:

- When validations or XSLT/XQuery transformations are carried out via project folder context menus, then the validation or transformation files specified in this dialog take precedence over any assignment in the XML file. Also, settings specified for individual project folders take precedence over settings specified for ancestor folders.
- If one file is present in multiple folders of the project and has been assigned different validation or transformation files in the different folders, then you can set which assignment to use when the file is processed outside the project. Specify this as follows: Locate the file in the project folder whose assignment/s you wish to use. Right-click the file in that project folder, and select **Properties**. In the dialog that appears (screenshot below), select **Use settings in current folder as default**. (The current folder is the project
folder in which the file is located.) If the option is disabled, it means that the settings of the current folder are already selected as the default settings to use. If you select a file instance that is in a project folder that is not the default, then the option is enabled, and you can switch the default settings to be this folder's settings. Note that, if the file has a local assignment (that is, an assignment within the file itself), then the local assignment will be used, and the default folder settings will be ignored.

Properties

File name: C:\Example\ipo.xml

[ ] Use settings in current folder as default

OK Cancel
11.3.17 Most Recently Used Projects

This command displays the file name and path for the nine most recently used projects, allowing quick access to these files.

Also note, that Authentic Desktop can automatically open the last project that you used, whenever you start Authentic Desktop. (Tools | Options | File section, Project | Open last project on program start).
11.4 XML Menu

The XML menu contains commands commonly used when working with XML documents.

![Check well-formedness](F7)
![Validate](F8)

Among the most frequently used XML tasks are checks for the well-formedness of documents and validity of XML documents. Commands for these tasks are in this menu.
11.4.1 Check Well-Formedness

The XML | Check well-formedness (F7) command checks the active document for well-formedness by the definitions of the XML 1.0 specification. Every XML document must be well-formed. Authentic Desktop checks for well-formedness whenever a document is opened or saved.

If the well-formedness check succeeds, a message is displayed in the Messages window (screenshot below).

![Messages Window](image)

If an error is encountered during the well-formedness check, a corresponding error message is displayed (screenshot below).

![Messages Window](image)

**Note:** The Messages window has nine tabs. The validation result is always displayed in the active tab. So you can validate one XML document in Tab-1 and retain the result in that tab. To validate a second document, switch to Tab-2 (or Tab-3 if you like) before running the check. If you do not switch tabs, Tab-1 (or the active tab) will be overwritten with the results of the latest validation.

**Validating from the Project window**

The **Validate** command can also be applied to a file, folder, or group of files in the active project. Select the required file or folder in the Project Window (by clicking on it). Then click XML | **Validate** or F8. Invalid files in a project will be opened and made active in the Main Window, and the **File is not valid** error message will be displayed.
**Note:** The Messages window has nine tabs. The result of the well-formed check is always displayed in the active tab. So you can check the well-formedness of one XML document in Tab-1 and retain the result in that tab. To check the well-formedness of a second document, switch to Tab-2 (or Tab-3 if you like) before running the check. If you do not switch tabs, Tab-1 (or the active tab) will be overwritten with the results of the latest check.

It is generally not permitted to save a malformed XML document, but Authentic Desktop gives you a Save Anyway option. This is useful when you want to suspend your work temporarily (in a not well-formed condition) and resume it later.

**Note:** You can also use the **Check well-formedness** command on any file, folder, or group of files in the active project window. Click on the respective item, and then on the Check Well-Formedness icon.
11.4.2 Validate XML

The XML | Validate (F8) command enables you to validate XML documents against DTDs, XML Schemas, and other schemas. You can specify that a document be automatically validated when a file is opened or saved (Tools | Options | File). The Validate command also carries out a well-formedness check before checking validity, so there is no need to use the Check Well-Formedness command before using the Validate command.

If a document is valid, a successful validation message is displayed in the Messages window.

Otherwise, a message that describes the error is displayed. You can click on the links in the error message to jump to the node in the XML document where the error was found.

**Note:** The validation error indicators and smart fixes described above are refreshed only when the XML | Validate (F8) command is executed; they are not updated in the background. So, after correcting an error, you must run the Validate (F8) command again to make sure that the error has indeed been fixed.

**Note:** The Messages window has nine tabs. The validation result is always displayed in the active tab. So you can validate one XML document in Tab-1 and retain the result in that tab. To validate a second document, switch to Tab-2 (or Tab-3 if you like) before running the check. If you do not switch tabs, Tab-1 (or the active tab) will be overwritten with the results of the latest validation.

Validating from the Project window
The Validate command can also be applied to a file, folder, or group of files in the active project. Select the required file or folder in the Project Window (by clicking on it). Then click XML | Validate or F8. Invalid files in a project will be opened and made active in the Main Window, and the File is not valid error message will be displayed.

Automating validation with RaptorXML 2019
RaptorXML is Altova's standalone application for XML validation, XSLT transformation, and XQuery transformation. It can be used from the command line, via a COM interface, in Java programs, and in .NET applications. Validation tasks can therefore be automated with the use of RaptorXML. For example, you can create a batch file that calls RaptorXML to perform validation on
a set of documents and sends the output to a text file. See the RaptorXML documentation for details.
11.5 XSL/XQuery Menu

The XSL Transformation language lets you specify how an XML document should be converted into other XML documents or text files. One kind of XML document that is generated with an XSLT document is an FO document, which can then be further processed to generate PDF output. Authentic Desktop contains built-in XSLT processors (for XSLT 1.0, XSLT 2.0, and XSLT 3.0) and can link to an FO processor on your system to transform XML files and generate various kinds of outputs. The location of the FO processor must be specified in the XSL section of the Options dialog (Tools | Options) in order to be able to use it directly from within the Authentic Desktop interface.

Commands to deal with all the above transformations are accessible in the XSL/XQuery menu. In addition, this menu also contains commands to work with the Altova XSLT/XQuery Debugger.

<table>
<thead>
<tr>
<th>Command</th>
<th>Keyboard Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSL Transformation</td>
<td>F10</td>
</tr>
<tr>
<td>XSL:FO Transformation</td>
<td>Ctrl+F10</td>
</tr>
<tr>
<td>XSL Parameters/XQuery Variables</td>
<td></td>
</tr>
</tbody>
</table>
11.5.1 XSL Transformation

The **XSL/XQuery | XSL Transformation** command transforms an XML document using an assigned XSLT stylesheet. The transformation can be carried out using the appropriate built-in Altova XSLT Engine (Altova XSLT 1.0 Engine for XSLT 1.0 stylesheets; Altova XSLT 2.0 Engine for XSLT 2.0 stylesheets; Altova XSLT 3.0 Engine for XSLT 3.0 stylesheets), the Microsoft-supplied MSXML module, or an external XSLT processor. The processor that is used in conjunction with this command is specified in the **XSL section** of the Options dialog (**Tools | Options**).

If your XML document contains a reference to an XSLT stylesheet, then this stylesheet is used for the transformation. (If the XML document is part of a project, an XSLT stylesheet can be specified on a per-folder basis in the **Project Properties** dialog. Right-click the project folder/s or file/s you wish to transform and select XSL Transformation.) If an XSLT stylesheet has not been assigned to an XML file, you are prompted for the XSLT stylesheet to use. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

**Automating validation with RaptorXML 2019**

RaptorXML is Altova's standalone application for XML validation, XSLT transformation, and XQuery transformation. It can be used from the command line, via a COM interface, in Java programs, and in .NET applications. XSLT transformation tasks can therefore be automated with the use of RaptorXML. For example, you can create a batch file that calls RaptorXML to run XSLT transformations on a set of documents and sends the output to a text file. See the RaptorXML documentation for details.

**Transformations to ZIP files**

In order to enforce output to a ZIP file, including Open Office XML (OOXML) files such as .docx, one must specify the ZIP protocol in the file path of the output file. For example:

```
filename.zip|zip/filename.xxx
filename.docx|zip/filename.xxx
```

**Note:** The directory structure might need to be created before running the transformation. If you are generating files for an Open Office XML archive, you would need to zip the archive files in order to create the top-level OOXML file (for example, .docx).
11.5.2 XSL-FO Transformation

CTRL+F10

FO is an XML format that describes paged documents. An FO processor, such as the Apache XML Project’s FOP, takes an FO file as input and generates PDF as output. The production of a PDF document from an XML document is, therefore, a two-step process.

1. The XML document is transformed to an FO document using an XSLT stylesheet.
2. The FO document is processed by an FO processor to generate PDF (or some alternative output).

The XSL/XQuery | XSL:FO Transformation command transforms an XML document or an FO document to PDF.

- If the XSL:FO Transformation command is executed on a source XML document, then both of the steps listed above are executed, in sequence, one after the other. If the XSLT stylesheet required to transform to FO is not referenced in the XML document, you are prompted to assign one for the transformation. Note that you can also select a file via a global resource or a URL (click the Browse button) or a file in one of the open windows in Authentic Desktop (click the Window button). The transformation from XML to XSL-FO is carried out by the XSLT processor specified in the XSL section of the Options dialog (Tools | Options). By default the selected XSLT processor is Authentic Desktop's built-in XSLT processor. The resultant FO document is directly processed with the FO processor specified in the XSL section of the Options dialog (Tools | Options).
- If the XSL:FO Transformation command is executed on an FO document, then the document is processed with the FO processor specified in the XSL section of the Options dialog (Tools | Options).

XSL:FO Transformation output

The XSL:FO Transformation command pops up the Choose XSL:FO Output dialog (screenshot below). (If the active document is an XML document without an XSLT assignment, you are first prompted for an XSLT file.)
You can view the output of the FO processor directly on screen using FOP viewer or you can generate an output file in any one of the following formats: PDF, text, an XML area tree, MIF PCL, or PostScript. You can also switch on messages from the FO processor to show (i) the processor's standard output message in the Messages window; and (ii) the processor's error messages in the Messages window. To switch on either these two options, check the appropriate check box at the bottom of the dialog.

**Note:**

- Unless you deselected the option to install the FOP processor of the Apache XML Project, it will have been installed in the folder `C:\ProgramData\Altova\SharedBetweenVersions`. If installed, the path to it will automatically have been entered in the XSL section of the Options dialog (Tools | Options) as the FO processor to use. You can set the path to any FO processor you wish to use.
- The XSL:FO Transformation command can not only be used on the active file in the Main Window but also on any file or folder you select in the active project. To do this, right-click and select XSL:FO Transformation. The XSLT stylesheet assigned to the selected project folder is used.
11.5.3 XSL Parameters / XQuery Variables

The XSL/XQuery | XSL Parameters/XQuery Variables command opens the XSLT Input Parameters/XQuery External Variables dialog (see screenshot). You can enter the name of one or more parameters you wish to pass to the XSLT stylesheet, or one or more external XQuery variables you wish to pass to the XQuery document, and their respective values. These parameters are used as follows in Authentic Desktop:

- When the **XSL Transformation** command in the XSL/XQuery menu is used to transform an XML document, the parameter values currently saved in the dialog are passed to the selected XSLT document and used for the transformation.
- When the **XQuery Execution** command in the XSL/XQuery menu is used to process an XQuery document, the XQuery external variable values currently saved in the dialog are passed to the XQuery document for the execution.

**Note:** Parameters or variables that you enter in the XSLT Input Parameters/XQuery External Variables dialog are only passed on to the built-in Altova XSLT engine. Therefore, if you are using MSXML or another external engine that you have configured, these parameters are not passed to this engine.

**Note:** It is not an error if an XSLT parameter or external XQuery variable is defined in the XSLT Input Parameters/XQuery External Variables dialog but is not used in the XSLT/XQuery document or the transformation.

Using XSLT Parameters

The value you enter for the parameter can be an XPath expression without quotes or a text string delimited by quotes. If the active document is an XSLT document, the **Get from XSL** button will be enabled. Clicking this button inserts parameters declared in the XSLT into the dialog together with their default values. This enables you to quickly include declared parameters and then change their default values as required.

**Note:** Once a set of parameter-values is entered in the dialog, it is used for all subsequent transformations until it is explicitly deleted or the application is restarted. Parameters entered in the dialog are specified at the application-level for that session, and will be passed to the respective XSLT document for every transformation that is carried out via the IDE from that moment onward. This means that:

- parameters are not associated with any particular document
- any parameter entered in the dialog is erased once Authentic Desktop has been closed.
Usage example for XSLT parameters

We have an XML document that contains the names of countries and their respective capitals:

```xml
<document>
  <countries>
    <country name="USA" capital="Washington DC"/>
    <country name="UK" capital="London"/>
    <country name="France" capital="Paris"/>
    <country name="Russia" capital="Moscow"/>
    <country name="China" capital="Beijing"/>
  </countries>
</document>
```

The following XSLT document will generate an XML document that displays one country from the XML file together with that country’s capital. The country is selected by entering its name as the value of the parameter named `country` (shown highlighted in yellow below).

```xml
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="country" select="'USA'"/>
  <xsl:template match="countries">
    <xsl:for-each select="country[@name="$country"]">
      <country>
        <name><xsl:value-of select="$country"/></name>
        <capital><xsl:value-of select="@capital"/></capital>
      </country>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

When this XSLT document is run on the XML document listed above, the result will be this:

```xml
<country><name>USA</name><capital>Washington DC</capital></country>
```

Now, if in the XSLT Input Parameters/XQuery External Variables dialog you create a parameter named `country` and give it a value (see screenshot above), then this value will be passed to the parameter `country` in the XSLT stylesheet for the transformation. In this way, you can pass different values to different parameters at run time.

Note:

- If you use the [XSL:FO Transformation](/Help/Tools/XSLT/Reference/Transformation/FO) command (XSL/XQuery | XSL:FO Transformation), then parameters entered in the XSLT Input Parameters/XQuery External Variables dialog are not passed to the stylesheets. In order for these parameters to be used in PDF output, first transform from XML to FO using the XSLT Transformation command (XSL/XQuery | XSL Transformation), and then transform the FO to PDF using the [XSL:FO Transformation](/Help/Tools/XSLT/Reference/Transformation/FO) command (XSL/XQuery | XSL:FO Transformation).
- If you use an XSLT processor other than the built-in Altova XSLT Engines, parameters you enter using the Input Parameters dialog will not be passed to the external processor.
Using external XQuery variables

The value you enter for an external XQuery variable could be an XPath expression without quotes or a text string delimited by quotes. The datatype of the external variable is specified in the variable declaration in the XQuery document.

![XSLT Input Parameters/XQuery External Variables](image)

**Note:** Once a set of external XQuery variables are entered in the dialog, they are used for all subsequent executions until they are explicitly deleted or the application is restarted. Variables entered in the dialog are specified at the application-level, and will be passed to the respective XQuery document for every execution that is carried out via the IDE from that moment onward. This means that:

- Variables are not associated with any particular document
- Any variable entered in the dialog is erased once the application (Authentic Desktop) has been closed down.

Usage example for external XQuery variables

In the following example, a variable `$first` is declared in the XQuery document and is then used in the return clause of the FLWOR expression:

```xml
xquery version "1.0";
declare variable $first as xs:string external;
let $last := "Jones"
return concat($first, " ", $last )
```

This XQuery returns Peter Jones, if the value of the external variable (entered in the XSLT Input Parameters/XQuery External Variables dialog) is Peter. Note the following:

- The `external` keyword in the variable declaration in the XQuery document indicates that this variable is an external variable.
- Defining the static type of the variable is optional. If a datatype for the variable is not specified in the variable declaration, then the variable value is assigned the type `xs:untypedAtomic`.
- If an external variable is declared in the XQuery document, but no external variable of that name is passed to the XQuery document, then an error is reported.
- If an external variable is declared and is entered in the XSLT Input Parameters/XQuery External Variables dialog, then it is considered to be in scope for the XQuery document being executed. If a new variable with that name is declared within the XQuery document,
the new variable temporarily overrides the in-scope external variable. For example, the XQuery document below returns Paul Jones even though the in-scope external variable $first has a value of Peter.

```xquery
xquery version "1.0";
declare variable $first as xs:string external;
let $first := "Paul"
let $last := "Jones"
return concat($first, " ", $last )
```
11.6 Authentic Menu

Authentic View enables you to edit XML documents based on StyleVision Power Stylesheets (.sps files) created in Altova’s StyleVision product! These stylesheets contain information that enables an XML file to be displayed graphically in Authentic View. In addition to containing display information, StyleVision Power Stylesheets also allow you to write data to the XML file. This data is dynamically processed using all the capability available to XSLT stylesheets and instantly produces the output in Authentic View.

Additionally, StyleVision Power Stylesheets can be created to display an editable XML view of a database. The StyleVision Power Stylesheet contains information for connecting to the database, displaying the data from the database in Authentic View, and writing back to the database.

The Authentic menu contains commands relevant to editing XML documents in Authentic View. For a tutorial on Authentic View, see the Authentic View Tutorials section.
11.6.1 **New Document**

This command enables you to open a new XML document template in Authentic View. The XML document template is based on a StyleVision Power Stylesheet (.sps file), and is opened by selecting the StyleVision Power Stylesheet (SPS file) in the Create New Document dialog (screenshot below). On selecting an SPS and clicking **OK**, the XML document template defined for that SPS file is opened in Authentic View.

![Create New Document dialog](image)

The new XML document template will therefore have the documentation presentation properties defined in the SPS and the data of the XML file that was selected as the Template XML File. The Authentic View user can now edit the XML document template in a graphical WYSIWYG interface, and save it as an XML document.
11.6.2 Edit Database Data

The Authentic Edit Database Data... command enables you to open an editable view of a database (DB) in Authentic View. All the information about connecting to the DB and how to display the DB and accept changes to it in Authentic View is contained in a StyleVision Power Stylesheet. It is such a DB-based StyleVision Power Stylesheet that you open with the Edit Database Data... command. This sets up a connection to the DB and displays the DB data (through an XML lens) in Authentic View.

Clicking the Edit Database Data... command opens the Edit Database Data dialog.

Browse for the required SPS file, and select it. This connects to the DB and opens an editable view of the DB in Authentic View. The design of the DB view displayed in Authentic View is contained in the StyleVision Power Stylesheet.

Please note: If, with the Edit Database Data... command, you attempt to open a StyleVision Power Stylesheet that is not based on a DB or to open a DB-based StyleVision Power Stylesheet that was created in a version of StyleVision prior to the StyleVision 2005 release, you will receive an error.

Please note: StyleVision Power Stylesheets are created using Altova StyleVision.
11.6.3 Edit StyleVision Stylesheet

The Authentic | Edit StyleVision Stylesheet command starts StyleVision and allows you to edit the StyleVision Power Stylesheet immediately in StyleVision. The command is enabled only if a StyleVision Power Stylesheet has been assigned to the XML document.
11.6.4 Select New Row with XML Data for Editing

The **Select New Row with XML Data for Editing** command enables you to select a new row from the relevant table in an XML DB, such as IBM DB2. This row appears in Authentic View, can be edited there, and then saved back to the DB.

When an XML DB is used as the XML data source, the XML data that is displayed in Authentic View is the XML document contained in one of the cells of the XML data column. The **Select New Row with XML Data for Editing** command enables you to select an XML document from another cell (or row) of that XML column. Selecting the **Select New Row...** command pops up the Choose XML Field dialog (screenshot below), which displays the table containing the XML column.

![Choose XML field dialog](screenshot)

You can enter a filter for this table. The filter should be an SQL **WHERE** clause (just the condition, without the **WHERE** keyword, for example: `CID>1002`). Click **Update** to refresh the dialog. In the screenshot above, you can see the result of a filtered view. Next, select the cell containing the required XML document and click **OK**. The XML document in the selected cell (row) is loaded into Authentic View.
11.6.5 XML Signature

The XML Signature command is available in Authentic View when the associated SPS has XML Signatures enabled. The XML Signature command is also available as the XML Signature toolbar icon in the Authentic toolbar.

Verification and own certificate/password

Clicking the XML Signature command starts the signature verification process. If no signature is present in the document, a message to that effect is displayed in the XML Signature dialog (see screenshot below), and the dialog will have a button that enables the Authentic View user to sign the document.

If the Select Own Certificate or Select Own Password button is present in this dialog, it means that the Authentic View has been given the option of selecting an own certificate/password. (Whether a certificate or password is to be chosen has been decided by the SPS designer at the time the signature was configured. The signature will be either certificate-based or password-based.) Clicking either of these buttons, if present in the dialog, enables the Authentic View user to browse for a certificate or to enter a password. The Authentic View user's selection is stored in memory and is valid for the current session only. If, after selecting a certificate or password, the document or application is closed, the certificate/password setting reverts to the setting originally saved with the SPS.

Verification and authentication information

If the verification process is run on a signed document, two general situations are possible. First: If the authentication information is available (in the signature or the SPS), then the verification process is executed directly and the result is displayed (screenshot below).

Authentication information is either the signing certificate's key information or the signing password. The SPS designer will have specified whether the certificate's key information is saved in the signature when the XML document is signed, or, in the case of a password-based signature, whether the password is saved in the SPS. In either of these cases, the authentication is available. Consequently the verification process will be run directly, without requiring any input from the Authentic View user.
The second possible general situation occurs when authentication information is not available in the signature (certificate’s key information) or SPS file (password). In this situation, the Authentic View user will be asked to supply the authentication information: a password (see screenshot below) or the location of a certificate.
11.6.6 Define XML Entities

You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the Insert Entity submenu of the context menu. When you double-click on an entity in the Entities Entry Helper, that entity is inserted at the cursor insertion point.

An entity is useful if you will be using a text string, XML fragment, or some other external resource in multiple locations in your document. You define the entity, which is basically a short name that stands in for the required data, in the Define Entities dialog. After defining an entity you can use it at multiple locations in your document. This helps you save time and greatly enhances maintenance.

There are two broad types of entities you can use in your document: a parsed entity, which is XML data (either a text string or a fragment of an XML document), or an unparsed entity, which is non-XML data such as a binary file (usually a graphic, sound, or multimedia object). Each entity has a name and a value. In the case of parsed entities the entity is a placeholder for the XML data. The value of the entity is either the XML data itself or a URI that points to a .xml file that contains the XML data. In the case of unparsed entities, the value of the entity is a URI that points to the non-XML data file.

To define an entity:

1. Click Authentic | Define XML Entities... This opens the Define Entities dialog.

![Define Entities](image)

2. Enter the name of your entity in the **Name** field. This is the name that will appear in the Entities Entry Helper.

3. Enter the type of entity from the drop-down list in the **Type** field. Three types are possible. An **Internal** entity is one for which the text to be used is stored in the XML document itself. Selecting **PUBLIC** or **SYSTEM** specifies that the resource is located outside the XML file, and will be located with the use of a public identifier or a system identifier, respectively. A system identifier is a URI that gives the location of the resource. A public identifier is a location-independent identifier, which enables some processors to identify the resource. If you specify both a public and system identifier, the public identifier resolves to the system identifier, and the system identifier is used.

4. If you have selected PUBLIC as the Type, enter the public identifier of your resource in the **PUBLIC** field. If you have selected Internal or SYSTEM as your Type, the **PUBLIC**
field is disabled.

5. In the **Value/Path** field, you can enter any one of the following:

- If the entity type is Internal, enter the text string you want as the value of your entity. Do not enter quotes to delimit the entry. Any quotes that you enter will be treated as part of the text string.
- If the entity type is SYSTEM, enter the URI of the resource or select a resource on your local network by using the **Browse** button. If the resource contains parsed data, it must be an XML file (i.e. it must have a .xml extension). Alternatively, the resource can be a binary file, such as a GIF file.
- If the entity type is PUBLIC, you must additionally enter a system identifier in this field.

6. The NDATA entry tells the processor that this entity is not to be parsed but to be sent to the appropriate processor. The NDATA field should therefore be used with unparsed entities only.

**Dialog features**

You can append, insert, and delete entities by clicking the appropriate buttons. You can also sort entities on the alphabetical value of any column by clicking the column header; clicking once sorts in ascending order, twice in descending order. You can also resize the dialog box and the width of columns.

Once an entity is used in the XML document, it is locked and cannot be edited in the Define Entities dialog. Locked entities are indicated by a lock symbol in the first column. Locking an entity ensures that the XML document valid with respect to entities. (The document would be invalid if an entity is referenced but not defined.)

Duplicate entities are flagged.

**Limitations**

- An entity contained within another entity is not resolved, either in the dialog, Authentic View, or XSLT output, and the ampersand character of such an entity is displayed in its escaped form, i.e. &amp;.
- External entities are not resolved in Authentic View, except in the case where an entity is an image file and it is entered as the value of an attribute which has been defined in the schema as being of type ENTITY or ENTITIES. Such entities are resolved when the document is processed with an XSLT generated from the SPS.
11.6.7 View Markup

The View Markup command has a submenu with options to control markup in the Authentic XML document. These options are described below.

The Hide Markup command hides markup symbols in Authentic View.

The Show Small Markup command shows small markup symbols in Authentic View.

The Show Large Markup command shows large markup symbols in Authentic View.

The Show Mixed Markup command shows mixed markup symbols in Authentic View. The person who designs the StyleVision Power Stylesheet can specify either large markup, small markup, or no markup for individual elements/attributes in the document. The Authentic View user sees this customized markup in mixed markup viewing mode.
11.6.8 RichEdit

Mousing over the RichEdit command pops out a submenu containing the RichEdit markup commands (screenshot below). The menu commands in this submenu are enabled only in Authentic View and when the cursor is placed inside an element that has been created as a RichEdit component in the SPS design.

<table>
<thead>
<tr>
<th>B</th>
<th>Toggle Bold</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Toggle Italic</td>
</tr>
<tr>
<td>U</td>
<td>Toggle Underline</td>
</tr>
<tr>
<td>S</td>
<td>Toggle Strikethrough</td>
</tr>
<tr>
<td></td>
<td>Foreground Color</td>
</tr>
<tr>
<td></td>
<td>Background Color</td>
</tr>
<tr>
<td></td>
<td>Align Left</td>
</tr>
<tr>
<td></td>
<td>Center</td>
</tr>
<tr>
<td></td>
<td>Align Right</td>
</tr>
</tbody>
</table>

The text-styling properties of the RichEdit menu will be applied to the selected text when a RichEdit markup command is clicked. The Authentic View user can, in addition to the font and font-size specified in the Authentic toolbar, additionally specify the font-weight, font-style, font-decoration, color, background color and alignment of the selected text.
11.6.9 Append/Insert/Duplicate/Delete Row

The **Append Row** command appends a row to the current table in Authentic View.

The **Insert Row** command inserts a row into the current table in Authentic View.

The **Duplicate Row** command duplicates the current table row in Authentic View.

The **Delete Row** command deletes the current table row in Authentic View.
11.6.10 Move Row Up/Down

The **Move Row Up** command moves the current table row up by one row in Authentic View.

The **Move Row Down** command moves the current table row down by one row in Authentic View.
11.6.11 Generate HTML, RTF, PDF, Word 2007+ Document

These four commands generate output documents from the Authentic View XML document stored in a PXF file:

- Generate an HTML Document
- Generate an RTF Document
- Generate a PDF Document
- Generate a Word 2007+ Document

They are also available in the Portable XML Form (PXF) toolbar (screenshot below).

![Screenshot of commands](image)

Clicking the individual command or buttons generates HTML, RTF, PDF, or DocX output, respectively.

These buttons are enabled when a PXF file is opened in Authentic View. Individual commands and buttons are enabled if the PXF file was configured to contain the XSLT stylesheet for that specific output format. For example, if the PXF file was configured to contain the XSLT stylesheets for HTML and RTF, then only the commands and toolbar buttons for HTML and RTF output will be enabled while those for PDF and DocX (Word 2007+) output will be disabled.
11.6.12 Trusted Locations

The Trusted Locations command pops up the Trusted Locations dialog (screenshot below), in which you can specify the security settings for scripts in an SPS. When an XML file based on a script-containing SPS is switched to Authentic View, the script will be allowed to run or not depending on the settings you make in this dialog.

The three available options are:

- Authentic scripts are always run when a file is opened in Authentic View.
- Authentic scripts are never run when a file is opened in Authentic View.
- Only Authentic scripts in trusted locations are run. The list of trusted (folder) locations is shown in the bottom pane. Use the Add button to browse for a folder and add it to the list. To remove an entry from the list, select an entry in the Trusted Locations list and click Remove.
11.7 View Menu

The View menu (screenshot below) controls the display of the active Main window and allows you to change the way the document is displayed.
This section provides a description of commands in the View menu.
11.7.1 Authentic View

This command switches the current document to Authentic View.

Authentic View enables you to edit XML documents based on StyleVision Power Stylesheet templates created in Altova’s StyleVision application. These templates (StyleVision stylesheets or SPS files) display XML documents in a graphical format that makes editing the XML document easier (than editing it in a text format with markup).
11.7.2 Browser View

This command switches the current document to Browser View. An XML-enabled browser renders the XML document using information from available CSS and/or XSL stylesheets.

When switching to Browser View, the document is first checked for validity if the Validate upon saving option in the File section of the Options dialog (Tools | Options) is checked. For more information, see the Browser View section of this documentation.
11.8 Browser Menu

The commands in the Browser menu are enabled in Browser View only. The Back and Forward commands, however, is enabled in Schema View also, where it takes you to the previously used command.
11.8.1 Back

The Back command (**shortcut:** Alt + Left arrow) is enabled in Browser View and Schema View.

In Browser View, the **Back** command displays the previously viewed page. The **Backspace** key achieves the same effect. The command is useful if you click a link in your XML document and then want to return to your XML document.

In Schema View, the **Back** command takes you to the previously viewed component or view. It can take you back to up to 500 previously viewed positions.
11.8.2 Forward

The Forward command (shortcut: Alt + Right arrow) is enabled in Browser View. In Schema View it is enabled only after you have used the Back command. The Forward command moves you forward through (i) previously viewed pages in Browser View, and (ii) previous views of schema components in Schema View.
11.8.3 **Stop**

The **Stop** command is enabled in Browser View and instructs the browser to stop loading your document. This is useful if large external files or graphics are being downloaded over a slow Internet connection, and you wish to stop the process.
11.8.4 Refresh

The Refresh (F5) command is enabled in Browser View and updates Browser View by reloading the current document and documents related to the current document (such as CSS and XSL stylesheets, and DTDs).
11.8.5 Fonts

The Fonts command rolls out a sub-menu from which you can select the default font size for rendering the text of your XML document. The selection is available in Browser View only.
11.8.6  Separate Window

The Separate Window command is enabled in Browser View and undocks the Browser View of the document from the other views. As a separate window, Browser View can be displayed side-by-side with an editing view of the document.

To refresh the separated Browser View after making a change in an editing view, press F5 in the editing view. To dock a separate Browser View window back into the window containing the other views, make the Browser View window active and click the Separate Window command.
11.9 Tools Menu

The Tools menu allows you to:

- Check the spelling of your XML documents
- Access the scripting environment of Authentic Desktop. You can create, manage and store your own forms, macros and event handlers
- View the currently assigned macros
- Compare any two files to check for differences
- Compare any two folders to check for differences
- Access customized commands that use external applications. These commands can be created in the Tools tab of the Customize dialog.
- Define global resources
- Change the active configuration for global resources in XMLSpy
- Customize your version of Authentic Desktop: define your own toolbars, keyboard shortcuts, menus, and macros
- Define global Authentic Desktop settings
11.9.1 Spelling

Authentic Desktop's spellchecker with built-in language dictionaries (see note below) is enabled in Authentic View.

Note: The selection of built-in dictionaries that ship with Altova software does not constitute any language preferences by Altova, but is largely based on the availability of dictionaries that permit redistribution with commercial software, such as the MPL, LGPL, or BSD licenses. Many other open-source dictionaries exist, but are distributed under more restrictive licenses, such as the GPL license. Many of these dictionaries are available as part of a separate installer located at http://www.altova.com/dictionaries. It is your choice as to whether you can agree to the terms of the license applicable to the dictionary and whether the dictionary is appropriate for your use with the software on your computer.

This section describes how to use the spellchecker. It is organized into the following subsections:

- Selecting the spellchecker language
- Running the spelling check

Selecting the spellchecker language

The spellchecker language can be set as follows:

1. Click the Tools | Spelling Options menu command.
2. In the XML Spelling Options dialog that pops up, click the MoreSpelling Options button.
3. In the Spelling Options dialog that now pops up (screenshot below), select one of the installed dictionaries from the dropdown list of the Dictionary Language combo box.

4. Click OK to finish.

The dictionary language you selected will be used by the spellchecker for spelling checks. If the
language you want is not already installed, you can download additional language dictionaries. How to do this is described in the section, Adding dictionaries for the spellchecker.

Running the spellchecker
The Tools | Spelling (Shift+F7) command automatically starts checking the currently active XML document. If an unknown word is encountered, the Spelling: Not in Dictionary dialog pops up (screenshot below). Otherwise the spelling check runs through to completion.

![Spelling: English (US)](image)

The various parts of the Spelling: Not in Dictionary dialog and the available options are described below:

Not in Dictionary
This text box contains the word that cannot be found in either the selected language dictionary or user dictionary. The following options are available:

- You can edit the word in the text box manually or select a suggestion from the Suggestions pane. Then click Change to replace the word in the XML document with the edited word. (Double-clicking a suggestion inserts it directly in the XML document.) When a word is shown in the Not in Dictionary text box, it is also highlighted in the XML document, so you can edit the word directly in the document if you like. Clicking Change All will replace all occurrences of the word in the XML document with the edited word.
- You can choose to not make any change and to ignore the spellchecker warning—either just for the current occurrence of the word or for every occurrence of it.
- You can add the word to the user dictionary and so allow the word to be considered correct for all checks from the current check onwards.

Suggestions
This list box displays words resembling the unknown word (supplied from the language and user dictionary).
dictionaries). Double-clicking a word in this list automatically inserts it in the document and continues the spellchecking process.

*Ignore once*
This command allows you to continue checking the document while ignoring the first occurrence of the unknown word. The same word will be flagged again if it appears in the document.

*Ignore all*
This command ignores all instances of the unknown word in the whole document.

*Add to dictionary*
This command adds the unknown word to the user dictionary. You can access the user dictionary (in order to edit it) via the Spelling Options dialog.

*Change*
This command replaces the currently highlighted word in the XML document with the (edited) word in the *Not in Dictionary* text box.

*Change all*
This command replaces all occurrences of the currently highlighted word in the XML document with the (edited) word in the *Not in Dictionary* text box.

*Recheck Document*
The **Recheck Document** button restarts the check from the beginning of the document.

*Options*
Clicking the **Options** button opens a dialog box depending on the current view.

- If the current view is Authentic View, the Spelling Options dialog box is opened.

For more information about these dialog boxes, see the section Spelling Options.

*Close*
This command closes the Spelling dialog box.
11.9.2 Spelling Options

The Tools | Spelling Options command opens the Spelling Options.

Spelling options
The Spelling Options dialog is used to define global spellchecker options.

Always suggest corrections:
Activating this option causes suggestions (from both the language dictionary and the user dictionary) to be displayed in the Suggestions list box. Disabling this option causes no suggestions to be shown.

Make corrections only from main dictionary:
Activating this option causes only the language dictionary (main dictionary) to be used. The user dictionary is not scanned for suggestions. It also disables the User Dictionary button, preventing any editing of the user dictionary.

Ignore words in UPPER case:
Activating this option causes all upper case words to be ignored.

Ignore words with numbers:
Activating this option causes all words containing numbers to be ignored.

Split CamelCase words
CamelCase words are words that have capitalization within the word. For example the word "CamelCase" has the "C" of "Case" capitalized, and is therefore said to be CamelCased. Since CamelCased words are rarely found in dictionaries, the spellchecker would flag them as errors. To
avoid this, the *Split CamelCase words* option splits CamelCased words into their capitalized components and checks each component individually. This option is checked by default.

**Dictionary Language**

Use this combo box to select the dictionary language for the spellchecker. The default selection is US English. Other language dictionaries are available for download free of charge from the [Altova website](https://www.altova.com).

**Adding dictionaries for the spellchecker**

For each dictionary language there are two Hunspell dictionary files that work together: a .aff file and .dic file. All language dictionaries are installed in a *Lexicons* folder at the following location:

C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons.

**Within the Lexicons folder, different language dictionaries are each stored in different folder:**
<language name>\<dictionary files>. For example, files for the two English-language dictionaries (English (British) and English (US)) will be stored as below:

C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)\en_GB.aff
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)\en_GB.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.dic

In the Spelling Options dialog, the dropdown list of the *Dictionary Language* combo box displays the language dictionaries. These dictionaries are those available in the *Lexicons* folder and have the same names as the language subfolders in the *Lexicons* folder. For example, in the case of the English-language dictionaries shown above, the dictionaries would appear in the Dictionary Language combo box as: *English (British)* and *English (US)*.

All installed dictionaries are shared by the different users of the machine and the different major versions of Altova products (whether 32-bit or 64-bit).

You can add dictionaries for the spellchecker in two ways, neither of which require that the files be registered with the system:

- **By adding Hunspell dictionaries into a new subfolder of the *Lexicons* folder.** Hunspell dictionaries can be downloaded, for example, from [https://wiki.openoffice.org/wiki/Dictionaries](https://wiki.openoffice.org/wiki/Dictionaries) or [http://extensions.services.openoffice.org/en/dictionaries](http://extensions.services.openoffice.org/en/dictionaries). (Note that OpenOffice uses the zipped OXT format. So change the extension to .zip and unzip the .aff and .dic file to the language folders in the *Lexicons* folder. Also note that Hunspell dictionaries are based on Myspell dictionaries. So Myspell dictionaries can also be used.)
- **By using the Altova dictionary installer**, which installs a package of multiple language dictionaries by default to the correct location on your machine. The installer can be downloaded via the link in the Dictionary language pane of the Spelling Options dialog (see screenshot below). Installation of the dictionaries must be done with administrator rights, otherwise installation will fail with an error.
Note: It is your choice as to whether you agree to the terms of the license applicable to the dictionary and whether the dictionary is appropriate for your use with the software on your computer.

Working with the user dictionary
Each user has one user dictionary, in which user-allowed words can be stored. During a spell check, spellings are checked against a word list comprising the words in the language dictionary and the user dictionary. You can add words to and delete words from the user dictionary via the User Dictionary dialog (screenshot below). This dialog is accessed by clicking the User Dictionary button in the Spelling Options dialog (see second screenshot in this section).

To add a word to the user dictionary, enter the word in the Word text box and click Add. The word will be added to the alphabetical list in the Dictionary pane. To delete a word from the dictionary, select the word in the Dictionary pane and click Delete. The word will be deleted from the Dictionary pane. When you have finished editing the User Dictionary dialog, click OK for the changes to be saved to the user dictionary.

Words may also be added to the User Dictionary during a spelling check. If an unknown word is encountered during a spelling check, then the Spelling dialog pops up prompting you for the action you wish to take. If you click the Add to Dictionary button, then the unknown word is added to the user dictionary.

The user dictionary is located at: C:\Users\<user>\Documents\Altova\SpellChecker\Lexicons\user.dic
11.9.3 Scripting Editor

The Scripting Editor command opens the Scripting Editor window. How to work with the Scripting Editor is described in the Scripting section of this documentation.

Note: The .NET Framework version 2.0 or higher will have to be installed on your machine in order for the Scripting Editor to run.
11.9.4 Macros

Mousing over the Macros command rolls out a submenu containing the macros defined in the Scripting Project that is currently active in Authentic Desktop (screenshot below). The active Scripting Project is specified in the Scripting section of the Options dialog.

![Macros Submenu](screenshot)

Clicking a macro in the submenu (see screenshot above) runs the macro.
11.9.5 User-defined Tools

Placing the cursor over the User-defined Tools command rolls out a sub-menu containing custom-made commands that use external applications. You can create these commands in the Tools tab of the Customize dialog. Clicking one of these custom commands executes the action associated with this command.

The User-Defined Tools | Customize command opens the Tools tab of the Customize dialog (in which you can create the custom commands that appear in the menu of the User-Defined Tools command.)
11.9.6 Global Resources

The Global Resources command pops up the Global Resources dialog (screenshot below), in which you can:

- Specify the Global Resources XML File to use for global resources.
- Add file, folder, and database global resources (or aliases)
- Specify various configurations for each global resource (alias). Each configuration maps to a specific resource.

How to define global resources is described in detail in the section, Defining Global Resources.

**Note:** The Altova Global Resources dialog can also be accessed via the Global Resources toolbar (Tools | Customize | Toolbars | Global Resources).
11.9.7 Active Configuration

Mousing over the **Active Configuration** menu item rolls out a submenu containing all the configurations defined in the currently active [Global Resources XML File](#) *(screenshot below)*.

![Global Resources Menu](image)

The currently active configuration is indicated with a bullet. In the screenshot above the currently active configuration is **Default**. To change the active configuration, select the configuration you wish to make active.

**Note:** The active configuration can also be selected via the [Global Resources toolbar](#) *(Tools | Customize | Toolbars | Global Resources)*.
11.9.8 Customize

The Customize command lets you customize application menus and toolbars to suit your personal needs. Clicking the command pops up the Customize dialog, which has the following tabs:

- **Commands**: All application and macro commands can be dragged from this tab into menu bars, menus and toolbars.
- **Toolbars**: Toolbars can be activated, deactivated, and reset individually.
- **Tools**: Commands that open external programs from within the interface can be added to the interface.
- **Keyboard**: Keyboard shortcuts can be created for individual application and macro commands.
- **Menu**: Menu bars and context menus to be customized are selected and made active in this tab. Works together with the Commands tab.
- **Macros**: Macros can have new commands associated with them.
- **Plug-ins**: Plug-ins can be activated and integrated in the interface.
- **Options**: Display options for toolbars are set in this tab.

This section also describes the context menu that appears when the Customize dialog is open and menu bar, menu, or toolbar items are right-clicked.

**Commands**

The Commands tab allows you customize your menus and toolbars. You can add application commands to menus and toolbars according to your preference. Note, however, that you cannot create new application commands or menus yourself.

To add a command to a toolbar or menu:
1. Select the menu item **Tools | Customize**. The Customize dialog appears.
2. Select the **All Commands** category in the **Categories** list box. The available commands appear in the **Commands** list box.
3. Click on a command in the **Commands** list box and drag it to an to an existing menu or toolbar. An I-beam appears when you place the cursor over a valid position to drop the command.
4. Release the mouse button at the position you want to insert the command.

Note the following points.

- When you drag a command, a small button appears at the tip of mouse pointer: This indicates that the command is currently being dragged.
- An "x" below the pointer indicates that the command cannot be dropped at the current cursor position.
- If the cursor is moved to a position at which the command can be dropped (a toolbar or menu), the "x" disappears and an I-beam indicates the valid position.
- Commands can be placed in menus or toolbars. If you have created your own toolbar, you can use this customization mechanism to populate it.
- Moving the cursor over a closed menu, opens that menu, allowing you to insert the command anywhere in that menu.

Adding commands to context menus

You can also add commands to context menus by dragging commands from the **Commands** list box into the context menu. The procedure is as follows:

1. In the Customize dialog, click the **Menu** tab.
2. In the Context Menu pane, select a context menu from the combo box. The selected context menu pops up.
3. In the Customize dialog, switch back to the **Commands** tab.
4. Drag the command you wish to create from the **Commands** list box and drop it into the desired location in the context menu.

Deleting a command or menu

To delete a command from a menu, context menu (see above for details of accessing context menus), or toolbar, or to delete an entire menu, do the following.

1. Open the Customize dialog (**Tools | Customize**). The Customize dialog appears.
2. With the Customize dialog open (and any tab selected), right-click a menu or a menu command, and then select **Delete** from the context menu that pops up. Alternatively, drag the menu or menu command till an "x" icon appears below the mouse pointer, and then drop the menu or menu command. The menu or menu command will be deleted.

To re-instate deleted menu commands, use the mechanisms described in this section. To re-instate a deleted menu, go to **Tools | Customize | Menu**, and click the **Reset** button in the **Application Frame Menus** pane. Alternatively, go to **Tools | Customize | Toolbars**, select Menu Bar, and click the **Reset** button.
Toolbars

The Toolbars tab allows you: (i) to activate or deactivate specific toolbars (that is, to decide which ones to display in the interface); (ii) to set what icons are displayed in each toolbar; and (iii) to create your own specialized toolbars.

The toolbars contain icons for the most frequently used menu commands. Information about each icon is displayed in a tooltip and in the Status Bar when the cursor is placed over the icon. You can drag a toolbar to any location on the screen, where it will appear as a floating window.

Note: To add a command to a toolbar, drag the command you want from the Commands list box in the Commands tab to the toolbar. To delete a command from a toolbar, open the Customize dialog, and with any tab selected, drag the command out of the toolbar (see Commands for more details).

Note: Toolbar settings defined in a particular view are, by default, valid for that view only. To make the settings apply to all views, click the check box at the bottom of the dialog.

The following functionality is available:

- **To activate or deactivate a toolbar**: Click its check box in the Toolbars list box.

- **To apply changes to all views**: Click the check box at the bottom of the dialog. Otherwise, changes are applied only to the active view. Note that only changes made after clicking the All Views check box will apply to all views.

- **To add a new toolbar**: Click the New button and give the toolbar a name in the Toolbar Name dialog that pops up. From the Commands tab drag commands into the new toolbar.
- **To change the name of an added toolbar**: Select the added toolbar in the Toolbars pane, click the **Rename** button, and edit the name in the Toolbar Name dialog that pops up.

- **To reset the Menu bar**: Select the **Menu Bar** item in the Toolbars pane, and then click **Reset**. This resets the Menu bar to the state it was in when the application was installed.

- **To reset all toolbar and menu commands**: Click the **Reset All** button. This resets all toolbars and menus to the states they were in when the application was installed.

- **To delete a toolbar**: Select the toolbar you wish to delete in the Toolbars pane and click **Delete**.

- **To show text labels of commands in a particular toolbar**: Select that toolbar and click the **Show Text Labels** check box. Note that text labels have to be activated for each toolbar separately.

### Tools

The **Tools** tab allows you to set up commands to use external applications from within Authentic Desktop. These commands will be added to the **Tools | User-defined Tools** menu. For example, the active file in the main window of Authentic Desktop can be opened in an external application, such as Notepad, by clicking a command in the **Tools | User-defined Tools** menu that you created.

To set up a command to use an external application, do the following:

1. In the **Menu Contents** pane (see screenshot above), click the **New** icon in the title bar of the pane and, in the item line that is created, enter the name of the menu command you want. In the screenshot above, we have entered a single menu command, **Open in**
Notepad. We plan to use this command to open the active document in the external Notepad application. More commands can be added to the command list by clicking the New icon. A command can be moved up or down the list relative to other commands by using the Move Item Up and Move Item Down icons. To delete a command, select it and click the Delete icon.

2. To associate an external application with a command, select the command in the Menu Contents pane. Then, in the Command field, enter the path to, or browse for, the executable file of the external application. In the screenshot above, the path to the Notepad application has been entered in the Command field.

3. The actions available to be performed with the external application are displayed when you click the flyout button of the Arguments field (see screenshot above). These actions are described in the list below. When you select an action, a code string for the action is entered in the Arguments field.

4. If you wish to specify a current working directory, enter it in the Initial Directory field.

5. Click Close to finish.

The command/s you created will appear in the Tools | User-defined Tools menu, and in the context menu of Project window files and folders—in the User-defined Tools submenu.

When you click the command (in the Tools | User-defined Tools menu) that you created, the action you associated with the command will be executed. The command example shown in the screenshot above does the following: It opens, in Notepad, the document that is active in the Main Window of Authentic Desktop. The external application command is also available in the context menu of files in the Project window (right-click a file in the Project window to display that file's context menu). Via the Project Window you can also open multiple files (for applications that allow this) by making a multi-selection and then selecting the command from the context menu.

Arguments

The Arguments field specifies the action to be executed by the external application command. The following arguments are available.

- **Active Document File Path**: The command in the Tools | User-defined Tools menu opens the document that is active in Authentic Desktop in the external application. The command in the context menu of a file in the Project window opens the selected file in the external application.
- **Project File Path**: Opens the Authentic Desktop project file (the .spp file) in the external application.

Initial directory

The Initial Directory entry is optional and is a path that will be used as the current directory.

Keyboard

The Keyboard tab allows you to create new keyboard shortcuts, or change existing shortcuts, for any application command.
To assign a new shortcut to a command, or to change an existing shortcut, do the following.

1. Select the All Commands category in the Category combo box. Note that if a macro has been selected as an Associated Command, then macros are also available for selection in the Category combo box and a shortcut for the macro can be set.
2. In the Commands list box, select the command to which you wish to assign a new shortcut or select the command the shortcut of which you wish to change.
3. Click in the Press New Shortcut Key text box, and press the shortcut you wish to assign to that command. The shortcut appears in the Press New Shortcut Key text box. If the shortcut has not yet been assigned to any command, the Assign button is enabled. If the shortcut has already been assigned to a command, then that command is displayed below the text box and the Assign button is disabled. (To clear the Press New Shortcut Key text box, press any of the control keys, Ctrl, Alt or Shift).
4. Click the Assign button to assign the shortcut. The shortcut now appears in the Current Keys list box. You can assign multiple shortcuts to a single command.
5. Click the Close button to confirm.

Deleting a shortcut
A shortcut cannot be assigned to multiple commands. If you wish to delete a shortcut, click it in the Current Keys list box and then click the Remove button.

Set accelerator for
Currently, accelerators can be set only as default. No other mode is available.
Default keyboard shortcuts

The default shortcuts of commonly used commands are listed below. An overview of all the
application’s menu commands is available in the Keyboard Map (Help | Keyboard Map).

- **Function-key shortcuts (incl. for validation and transformation)**

<table>
<thead>
<tr>
<th>Key Sequence</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Help Menu</td>
</tr>
<tr>
<td>F1 + Alt</td>
<td>Open Last File</td>
</tr>
<tr>
<td>F3</td>
<td>Find Next</td>
</tr>
<tr>
<td>F4 + CTRL</td>
<td>Close Active Window</td>
</tr>
<tr>
<td>F4 + Alt</td>
<td>Close Authentic Desktop</td>
</tr>
<tr>
<td>F5</td>
<td>Refresh</td>
</tr>
<tr>
<td>F6 + CTRL</td>
<td>Cycle through Open Windows</td>
</tr>
<tr>
<td>F7</td>
<td>Check Well-formedness</td>
</tr>
<tr>
<td>F8</td>
<td>Validate</td>
</tr>
<tr>
<td>F10</td>
<td>XSL Transformation</td>
</tr>
<tr>
<td>F10 + CTRL</td>
<td>XSL:FO Transformation</td>
</tr>
</tbody>
</table>

- **File and Application commands**

<table>
<thead>
<tr>
<th>Key Sequence</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt + F1</td>
<td>Open Last File</td>
</tr>
<tr>
<td>CTRL + O</td>
<td>File Open</td>
</tr>
<tr>
<td>CTRL + N</td>
<td>File New</td>
</tr>
<tr>
<td>CTRL + P</td>
<td>File Print</td>
</tr>
<tr>
<td>CTRL + S</td>
<td>File Save</td>
</tr>
<tr>
<td>CTRL + F4</td>
<td>Close Active Window</td>
</tr>
<tr>
<td>CTRL + F6</td>
<td>Cycle through Open Windows</td>
</tr>
<tr>
<td>CTRL + TAB</td>
<td>Switch between Open Documents</td>
</tr>
<tr>
<td>Alt + F4</td>
<td>Close Authentic Desktop</td>
</tr>
</tbody>
</table>

- **Numeric keypad shortcuts (to expand/collapse nodes)**

<table>
<thead>
<tr>
<th>Key Sequence</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num +</td>
<td>Expand</td>
</tr>
<tr>
<td>Num *</td>
<td>Expand Fully</td>
</tr>
<tr>
<td>Num –</td>
<td>Collapse</td>
</tr>
<tr>
<td>CTRL + Num –</td>
<td>Collapse Unselected</td>
</tr>
</tbody>
</table>

- **Miscellaneous keys**

<table>
<thead>
<tr>
<th>Key Sequence</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up/Down Arrow Keys</td>
<td>Move Cursor or Selection Bar</td>
</tr>
<tr>
<td>Esc</td>
<td>Abandon Edits or Close Dialog Box</td>
</tr>
<tr>
<td>Return</td>
<td>Confirm Selection</td>
</tr>
</tbody>
</table>
Menu

The Menu tab allows you to customize the two main menu bars (default and application menu bars) as well as the application's context menus.

Customizing the default menu bar and application menu bar

The default menu bar is the menu bar that is displayed when no document is open in the main window. The application menu bar is the menu bar that is displayed when one or more documents are open in the main window. Each menu bar can be customized separately, and customization
changes made to one do not affect the other.

To customize a menu bar, select it in the Show Menus For combo box (see screenshot above). Then switch to the Commands tab of the Customize dialog and drag commands from the Commands list box to the menu bar or into any of the menus.

Deleting commands from menus and resetting the menu bars
To delete an entire menu or a command inside a menu, select that menu or menu command, and then either (i) right-click and select Delete, or (ii) drag away from the menu bar or menu, respectively.

You can reset each of these two menu bars (default and application menu bars) to its original installation state by selecting the menu in the Show Menus For combo box and then clicking the Reset button below the combo box.

Customizing the application's context menus
Context menus are the menus that appear when you right-click certain objects in the application's interface. Each of these context menus can be customized by doing the following:

1. Select the context menu you want in the Select Context Menu combo box. This pops up the context menu.
2. Switching to the Commands tab of the Customize dialog.
3. Drag a command from the Commands list box into the context menu.
4. If you wish to delete a command from the context menu, right-click that command in the context menu, and click Delete. Alternatively, you can drag the command you want to delete out of the context menu.

You can reset any context menu to its original installation state by selecting it in the Select Context Menu combo box and then clicking the Reset button below the combo box.

Menu shadows
Click the Menu shadows check box to give all menus shadows.

Macros
The Macros tab allows you to create application commands for macros that were created using Authentic Desktop's Scripting Editor. These application commands (which run the macros associated with them) can subsequently be made available in menus and toolbars, either from the Macros tab directly or by using the mechanisms available in the Commands tab of the Customize dialog. As application commands, they can also be assigned shortcuts in the Keyboard tab of the Customize dialog.

How macros work in Authentic Desktop
Macros in Authentic Desktop work as follows:

- Altova scripting projects (.asprj files) are created in Authentic Desktop's Scripting Editor. It is these scripting projects that can contain the macros used in Authentic Desktop.
Two scripting projects can be active at a time in Authentic Desktop: (i) An application scripting project, which is specified in the Scripting section of the Options dialog, and (ii) the scripting project of the active Authentic Desktop project, which is specified in the Script Settings dialog (Project | Script Settings).

The macros in these two scripting projects are available in the application: in the Project | Macros menu (from where the macros can be run), and in the Macros tab of the Customize dialog (screenshot below), in which they can be set as application commands. After a macro has been set as an application command, the command can be placed in a menu and/or toolbar.

Creating an application command for a macro

In Scripting Editor (Tools | Scripting Editor) create the macro you wish and save it to a scripting project. Specify this file to be either the application scripting project (via the Scripting section of the Options dialog) or the active application project's scripting project (via the application project's Script Settings dialog (Project | Script Settings)). The macros in the scripting project will now appear in the Macros pane of the Macros tab (see screenshot below).

To create an application command for a macro, select the macro in the Macros pane, set the text of the command in the Display Text text box, and click Add Command (see screenshot below). A command associated with the selected macro will be added to the Associated Commands list box.

To edit the icon of an associated command, select the command and click Edit Icon. To delete an associated command, click Remove.

Placing a macro-associated command in a menu or toolbar

There are two ways to place a macro-associated command in a menu or toolbar:
Drag the command from the Associated Commands list box to the desired location in the menu or toolbar.

Use the mechanisms available in the Commands tab of the Customize dialog.

In either case, the command will be created at the desired location. Clicking on the command in the menu or toolbar will execute the macro.

Note: If a macro has been set as an associated command, you can set a keyboard shortcut for it. In the Keyboard tab of the Customize dialog, select Macros in the Category combo box, then select the required macro, and set the shortcut. You must set a macro as an associated command in order for it to be available to be created as a keyboard shortcut.

Plug-Ins

The Plug-Ins tab allows you to integrate plug-ins and to place commands, where these have been so programmed, in an application menu and/or toolbar. In the Plug-In tab (screenshot below), click Add Plug-In, and browse for the plug-in's DLL file (see 'Creating plug-ins' below). Click OK to add the plug-in. Multiple plug-ins can be added.

After a plug-in has been added successfully, a description of the plug-in appears in the dialog and the Remove Plug-In button becomes enabled. If the plug-in code creates toolbars and menus, these will be immediately visible in the application interface. To remove a plug-in select it and click Remove Plug-In.

Creating plug-ins

Source code for sample plug-ins has been provided in the application's (My) Documents folder: Examples\IDEPlugin folder. To build a plug-in from such source code, do the following:
1. Open the solution you want to build as a plug-in in Visual Studio.
2. Build the plug-in with the command in the Build menu.
3. The plug-in's DLL file that will be created in the Bin or Debug folder. This DLL file is the file that must be added as a plug-in (see above).

For more information about plug-ins, see the section IDE Plugins.

Options
The Options tab allows you to define general environment settings.

Click the check boxes to toggle on the following options:

- **Show ScreenTips on toolbar**: Displays a popup when the mouse pointer is placed over an icon in any toolbar. The popup contains a short description of the icon function, as well as the associated keyboard shortcut, if one has been assigned and if the Show shortcut keys option has been checked.
- **Show shortcut keys in ScreenTips**: Defines whether shortcut information will be shown in screen tips.
- **Large icons**: Toggles the size of toolbar icons between standard and large.

Customize Context Menu
The Customize context menu (screenshot below) is the menu that appears when you have the Customize dialog open and then right-click an application menu, menu command, or toolbar icon.
The following functionality is available:

- **Reset to Default**: Currently no function.
- **Copy Button Image**: Copies the icon you right-click to the clipboard.
- **Delete**: Deletes the selected menu, menu command, or toolbar icon. For information about how to restore deleted items, see below.
- **Button Appearance**: Pops up the Button Appearance dialog (see screenshot below), in which you can set properties that define the appearance of the selected toolbar icon. See the description below for details.
- **Image, Text, Image and Text**: Mutually exclusive options that determine whether the selected toolbar icon will be an icon only, text only, or both icon and text. You can select one of these options to make the change. Alternatively, you can make this change in the Button Appearance dialog.
- **Start Group**: Inserts a vertical group-divider to the left of the selected toolbar icon. This makes the selected toolbar icon the first of a group of icons.

### The Button Appearance dialog

Right-click a toolbar icon and click **Button Appearance** to get the Button Appearance dialog (screenshot below). Via this dialog you can edit the toolbar icon image, as well as its text. Currently only toolbar icons for macros and from plug-ins can be edited using this dialog.
The following editing functionality is available for the selected toolbar icon (the one that was right-clicked to get the Customize context menu):

- **Image only, Text only, Image and text**: Select the desired radio button to specify what form the toolbar icon will take.
- **Image editing**: When *Image only* or *Image and text* is selected, then the image editing options are enabled. Click **New** to create a new image that will be added to the user-defined images in the images pane. Select an image and click **Edit** to edit it.
• **Image selection**: Select an image from the Images pane and click OK to use the selected image as the toolbar icon.

• **Text editing and selection**: When **Text only** or **Image and text** is selected, then the **Button Text** text box is enabled. Enter or edit the text and click OK to make this the text of the toolbar icon.

**Note**: The Button Appearance dialog can also be used to edit the text of menu commands. Right-click the menu command (with the Customize dialog open), click **Button Appearance**, and then edit the menu command text in the **Button Text** text box.

**Restoring deleted menus, menu commands, and toolbar icons**

If a menu, menu command, or toolbar icon has been deleted by using the **Delete** command in the Customize context menu, these can be restored as follows:

• **Menus**: Go to **Tools | Customize | Menu**, and click the **Reset** button in the **Application Frame Menus** pane. Alternatively, go to **Tools | Customize | Toolbars**, select Menu Bar, and click the **Reset** button.

• **Menu commands**: Go to **Tools | Customize | Commands**, and drag the command from the Commands list box into the menu.

• **Toolbar icons**: Go to **Tools | Customize | Commands**, and drag the command from the Commands list box into the toolbar.
11.9.9 **Restore Toolbars and Windows**

The **Restore Toolbars and Windows** command closes down Authentic Desktop and re-starts it with the default settings. Before it closes down a dialog pops up asking for confirmation about whether Authentic Desktop should be closed (*screenshot below*).

![Screenshot](image.png)

This command is useful if you have been resizing, moving, or hiding toolbars or windows, and would now like to have all the toolbars and windows as they originally were.
11.9.10 Options

The **Tools | Options** command enables you to define global application settings. These settings are organized in sections (see left pane in screenshot below). For example, the **File section** (shown in the screenshot below) contains options that specify how you want Authentic Desktop to open and save files. To specify options of a particular section, select that section in the left pane and specify the property values you want. The **OK** button saves changes to the registry and closes the dialog. The **Apply** button causes changes to be displayed in currently open documents.

Each section of the Options dialog is described in detail in its sub-section of this section.

**File**

The **File** section defines the way Authentic Desktop opens and saves documents. Related settings are in the **Encoding section**.

**Automatic reload of changed files**

If you are working in a multi-user environment, or if you are working on files that are dynamically generated on a server, you can watch for changes to files that are currently open in the interface. Each time Authentic Desktop detects a change in an open document, it will prompt you about whether you want to reload the changed file.

**Validation**

If you are using DTDs or schemas to define the structure of your XML documents, you can automatically check the document for validity whenever it is opened or saved. During Open and
Save operations, you have the option of validating files only if the file-size is less than a size you specify in MB. If the document is not valid, an error message will be displayed. If it is valid, no message will be displayed and the operation will proceed without any notification. Authentic Desktop can also cache these files in memory to save any unnecessary reloading (e.g. when the schema being referred to is accessed through a URL). If your schema location declaration uses an URL, disable the "cache DTD/Schema files in memory" option to have changes made to the schema appear immediately, and not use the cached version of the schema.

**XML Schema Version**

The XSD mode that is enabled in Schema View depends on both (i) the presence/absence—and, if present, the value—of the `/xs:schema/@vc:minVersion` attribute of the XSD document, and (ii) the XML Schema Version option selected in the File section of the Options dialog (Tools | Options, screenshot below).


<table>
<thead>
<tr>
<th>XML Schema Version</th>
<th>vc:minVersion attribute</th>
<th>XSD mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always v1.0</td>
<td>Is absent, or is present with any value</td>
<td>1.0</td>
</tr>
<tr>
<td>Always v1.1</td>
<td>Is absent, or is present with any value</td>
<td>1.1</td>
</tr>
<tr>
<td>Value of @vc:minVersion</td>
<td>Attribute has value of 1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>Value of @vc:minVersion</td>
<td>Attribute is absent, or attribute is</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>present with a value other than 1.1</td>
<td></td>
</tr>
</tbody>
</table>

**Project**

When you start Authentic Desktop, you can open the last-used project automatically.

**Save File**

When saving an XML document, Authentic Desktop includes a short comment `<!-- Edited with Authentic Desktop http://www.altova.com -->` near the top of the file. This option can only be deactivated by licensed users, and takes effect when editing or saving files in the Enhanced Grid or Schema Design View.

If a StyleVision Power Stylesheet is associated with an XML file, the 'Authentic: save link to design file' option will cause the link to the StyleVision Power Stylesheet to be saved with the XML file.
Line breaks
When you open a file, the character coding for line breaks in it are preserved if Preserve old is selected. Alternatively, you can choose to code line breaks in any of three codings: CR&LF (for PC), CR (for MacOS), or LF (for Unix).

No output formatting for
In Text View, the indentation of an element can be made to reflect its position in the element hierarchy (see Save File). You can, however, override this indentation for individual elements. To do this, enter the element name in the No output formatting for field. All elements entered in this field will be formatted such that their descendant elements have no whitespace between them (see screenshots).

Hierarchical indentation for all elements:

```
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>
```

No output formatting has been specified for element `xs:restriction`:

```
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>
```

Save and exit
After making the settings, click OK to finish.

File Types
The File Types section (screenshot below) allows you to customize the behavior of Authentic Desktop on a per-file-type basis.
Choose a file type from the File Types list box, and then customize the functions for that particular file type as described below.

**Windows Explorer settings**
You can define the file type description and MIME-compliant content type used by Windows Explorer and whether Authentic Desktop is to be the default editor for documents of this file type.

**Conformance**
Authentic Desktop provides specific intelligent editing features, as well as other features, for different file types. Authentic Desktop sets the features for a particular file type on the basis of the conformance you set in this option. A large number of file types are defined with a default conformance that is appropriate for the file type. We recommend that you do not modify these settings unless you are adding a new file type or deliberately wish to set a file type to another kind of conformance.

**Default view**
This group lets you define the default view to be used for each file type.

**Text View**
This check box lets you set syntax-coloring for particular file types.

**Disable automatic validation**
This option enables you to disable automatic validation per file type. Automatic validation typically
takes place when a file is opened or saved, or when a view is changed.

**Save empty elements in short \(<\mathrm{E}/\) format**

Some applications that use XML documents or output generated from XML documents may have problems understanding the short \(<\mathrm{Element}/\) form for empty elements defined in the XML 1.0 Specification. You can instruct Authentic Desktop to save elements in the longer (but also valid) \(<\mathrm{Element}/>\) form.

**Add new file extension**

Adds a new file type to the File types list. You must then define the settings for this new file type using the other options in this tab.

**Delete selected file extension**

Deletes the currently selected file type and all its associated settings.

**Save and exit**

After making the settings, click **OK** to finish.

**Encoding**

The **Encoding** section specifies options for file encodings.

The default encoding for new XML files can be set by selecting an option from the dropdown list. A new document is created with an XML declaration containing the encoding value you specify here. If a two- or four-byte encoding is selected as the default encoding (i.e. UTF-16, UCS-2, or UCS-4) you can also choose between little-endian and big-endian byte-ordering.
The encoding of existing XML files will be retained and can only be changed with the **File | Encoding** command.

**Open XML files with unknown encoding as**
If the encoding of an XML file cannot be determined or if the XML document has no encoding specification, the file will be opened with the encoding you select in this combo box.

**Open non-XML files in**
Existing and new non-XML files are opened with the encoding you select in this combo box. You can change the encoding of the document by using the **File | Encoding** command.

**BOM (Byte Order Mark)**
When a document with two-byte or four-byte character encoding is saved, the document can be saved either with (i) little-endian byte-ordering and a little-endian BOM (**Always create BOM if not UTF-8**); or (ii) the detected byte-ordering and the detected BOM (**Preserve detected BOM on saving**).

**Save and exit**
After making the settings, click **OK** to finish.

**View**
The **View** section enables you to customize the XML documents presentation in Authentic Desktop.

**Pretty-print**
When you select **Edit | Pretty-Print XML Text** in Text View or switch from another view to Text View, the XML document will be "pretty-printed". The pretty-printing will be with or without indentation according to whether the **Use Indentation** option in this dialog is checked or not.

**Program logo**
You can turn off the splash screen on program startup to speed up the application. Also, if you have a purchased license (as opposed to, say, a trial license), you will have the option of turning off the program logo, copyright notice, and registration details when printing a document from XMLSpy.

**Window title**
The window title for each document window can contain either the file name only or the full path name.

**Save and exit**
After making the settings, click **OK** to finish.
**XSL**

The **XSL** section *(screenshot below)* enables you to define options for **XSLT transformations** and **XSL-FO transformations** carried out from within the application.

--

**XSL**

- Built-in RaptorXML XSLT engine (Important: the built-in engine is always used for XSLT debugging)
  - Validate XML files used in transformation

- Microsoft® XML Parser (MSXML):  
  - v3.0  
  - v4.0  
  - v6.0  
  - Choose version automatically

- External XSL transformation program:

Please enter the command line for executing an external XSL transformation program in the form:
```
Program.exe %1 %2 %3
```
where %1 will be replaced with the XML input file name, %2 with the output file name and %3 (optional) with XSL style-sheet file name. Feel free to add any other parameters that are required by the external program.

- Show external program output in Messages window after transformation
- Show external program error output in Messages window after transformation

Default file extension of output file:  
- Reuse output window

Please enter path to XSL-FO transformation engine (if using FOP enter path to fop.bat):
```
C:\ProgramData\Altova\SharedBetweenVersions\Apache FOP 2.2\fop.bat
```

- Use XSL engine selected above to perform XSLT part and then XSL-FO engine for FO part
- Use XSL-FO engine for both XSLT and FO parts of transformation

---

**XSLT transformations**

Authentic Desktop contains the Altova RaptorXML XSLT 1.0, XSLT 2.0, and XSLT 3.0 engines, which you can use for XSLT transformations as well as for validating the XML files used in transformations. The appropriate XSLT engine (1.0, 2.0, or 3.0) is used (according to the value of the version attribute of the xsl:stylesheet or xsl:transform element).

For transforming XML documents using XSLT, you could use one of the following:

- The built-in Altova XSLT Engines (XSLT 1.0, XSLT 2.0, and XSLT 3.0).
- The MSXML 3.0, 4.0, or 6.0 parser (which is pre-installed). If you know which version of the MSXML parser is running on your machine, you could select it; otherwise, you should let the application select the version automatically. (The Choose version automatically option is active by default.) In this case, the application tries to select the most recent available version.
- An external XSLT processor of your choice. You must specify the command line string that the external XSLT processor uses to run a transformation. The following variables are available for building the command line string:
  
  \%1 = XML document to process
%2 = Output file to generate
%3 = XSLT stylesheet to use (if the XML document does not contain a reference to a stylesheet)

For example, say you have a processor that uses the following command to run an XSLT transformation:

```
myxsltengine.exe -o output.xml input.xml stylesheet.xslt parameter-name=parameter-value
```

Then, in Authentic Desktop, you can use the variables listed above to generate this command. Select the External XSL Transformation Program radio button, and enter the following line in the text box:

```
c:\myxsltengine\myxsltengine.exe -o %2 %1 %3 parameter-name=parameter-value
```

Check the respective check boxes to show the output and error messages of the external program in the Messages Window in Authentic Desktop.

The Reuse output window option causes subsequent transformations to display the result document in the same output window. If the XML file belongs to a project and Reuse output window option is disabled, the setting only takes effect if the Save in folder output file path (screenshot below) in the relevant project properties is also disabled.

---

**XSL-FO transformations**

FO documents are processed using an FO processor, and the path to the executable of the FO processor must be specified in the text box for the XSL-FO transformation engine. The transformation is carried out using the XSL/XQuery | XSL-FO Transformation menu command. If the source file (the active document when the command is executed in the IDE) is an XSL-FO document, the FO processor is invoked for the transformation. If the source document is an XML document, an XSLT transformation is required to first convert the XML document to an XSL-FO document. This XSLT transformation can be carried out either by the XSLT engine you have specified as the default engine for the application (see above), or by the XSLT engine that might be built into the FO processor you have specified as the default FO processor for the application. To select between these two options, click the appropriate radio button.

After making the settings, click OK to finish.

**Note:** Unless you deselected the option to install the FOP processor of the Apache XML Project, it will have been installed in the folder `C:\ProgramData\Altova\SharedBetweenVersions`. If installed, the path to it will automatically have been entered in the XSL-FO Engine input box. You can set the path to any FO processor you wish to use. Note, however, that the same path will be used by other Altova products that use FO processors and have settings to select the FO processor (StyleVision and Authentic
Save and exit
After making the settings, click OK to finish.

Java
On the Java tab, you can optionally enter the path to a Java VM (Virtual Machine) on your file system. Note that adding a custom Java VM path is not always necessary. By default, Authentic Desktop attempts to detect the Java VM path automatically by reading (in this order) the Windows registry and the JAVA_HOME environment variable. The custom path added on this dialog box will take priority over any other Java VM path detected automatically.

You may need to add a custom Java VM path, for example, if you are using a Java virtual machine which does not have an installer and does not create registry entries (for example, Oracle's OpenJDK). You might also want to set this path if you need to override, for whatever reason, any Java VM path detected automatically by Authentic Desktop.

Note the following:

- The Java VM path is shared between Altova desktop (not server) applications. Consequently, if you change it in one application, it will automatically apply to all other Altova applications.
- The path must point to the jvm.dll file from the \bin\server or \bin\client directory, relative to the directory where the JDK was installed.
- The Authentic Desktop platform (32-bit, 64-bit) must be the same as that of the JDK.
- After changing the Java VM path, you may need to restart Authentic Desktop for the new settings to take effect.
Scripting

The **Scripting** section (*screenshot below*) allows you to enable the **Scripting Environment** on application startup. Check the **Activate Scripting** check box to do this. You can then specify the Global Scripting Project file (*see screenshot below*).

To set a global scripting project for Authentic Desktop, check the **Activate Scripting** check box and then browse for the Altova Scripting Project (*.asprj*) file you want. You can also specify: (i) whether Auto-Macros in the scripting project should be automatically executed when Authentic Desktop starts, and (ii) whether application event handler scripts in the project should be automatically executed or not; check or uncheck the respective check boxes accordingly.

**Save and exit**

After making the settings, click **OK** to finish. Macros in the Global Scripting Project will then be displayed in the submenu of the **Macros** command.

Source Control

The **Source Control** section (*screenshot below*) enables you to specify the source control provider, and the settings and default logon ID for each source control provider.
Source Control Plugin

The current source control plugin can be selected from among the currently installed source control systems. These systems are listed in the dropdown list of the combo box. After selecting the required source control, specify the login ID for it in the next text box. The Advanced button pops up a dialog specific to the selected source control plugin, in which you can define settings for that source control plugin. These settings are different for different source control plugins.

User preferences

A range of user preferences is available, including the following:

- Status updates can be performed in the background after a user-defined interval of time, or they can be switched off entirely. Very large source control databases could consume considerable CPU and network resources. The system can be speeded up, however, by disabling background status updates or increasing the interval between them.
- When opening and closing projects, files can be automatically checked out and checked in, respectively.
- The display of the Check Out and Check In dialogs can be suppressed.
- The Reset button is enabled if you have checked/activated the Don't show this again option in one of the dialog boxes. On clicking the Reset button, the Don't show this again prompt is re-enabled.

Save and exit

After making the settings, click OK to finish.
Network Proxy

The Network Proxy section (screenshot below) enables you to configure custom proxy settings. The default is to use the system’s proxy settings, so the settings will work without user intervention. If you wish to specify an alternative network proxy, use one of the options to define the setting you want.

**Note:** The network proxy settings are shared among all Altova MissionKit applications. So, if you change the settings in one application, all MissionKit applications will be affected.

Use system proxy settings

Uses the Internet Explorer (IE) settings configurable via the system proxy settings. It also queries the settings configured with `netsh.exe winhttp`.

Automatic proxy configuration

The following options are provided:

- **Auto-detect settings**: Looks up a WPAD script (http://wpad.LOCALDOMAIN/wpad.dat) via DHCP or DNS, and uses this script for proxy setup.
- **Script URL**: Specify an HTTP URL to a proxy-auto-configuration (.pac) script that is to be used for proxy setup.
- **Reload**: Resets and reloads the current auto-proxy-configuration. This action requires Windows 8 or newer, and may need up to 30s to take effect.
**Manual proxy configuration**

Manually specify the fully qualified host name and port for the proxies of the respective protocols. A supported scheme may be included in the host name (for example: `http://hostname`). It is not required that the scheme is the same as the respective protocol if the proxy supports the scheme.

The following options are provided:

- *Use this proxy for all protocols*: Uses the host name and port of the HTTP Proxy for all protocols.
- *No Proxy for*: A semi-colon (`;`) separated list of fully qualified host names, domain names, or IP addresses for hosts that should be used without a proxy. IP addresses may not be truncated and IPv6 addresses have to be enclosed by square brackets (for example: `[2606:2800:220:1:248:1893:25c8:1946]`). Domain names must start with a leading dot (for example: `.example.com`).
- *Do not use the proxy server for local addresses*: If checked, adds `<local>` to the *No Proxy for* list. If this option is selected, then the following will not use the proxy: (i) `127.0.0.1`, (ii) `::1`, (iii) all host names not containing a dot character (`.`).

**Current proxy settings**

Provides a verbose log of the proxy detection. It can be refreshed with the **Refresh** button to the right of the **Test URL** field (for example, when changing the test URL, or when the proxy settings have been changed).

- **Test URL**: A test URL can be used to see which proxy is used for that specific URL. No I/O is done with this URL. This field must not be empty if proxy-auto-configuration is used (either through **Use system proxy settings** or **Authomatic proxy configuration**).
11.10 Window Menu

To organize the individual document windows in an Authentic Desktop session, the Window menu contains standard commands common to most Windows applications. You can cascade the open document windows, tile them, or arrange document icons once you have minimized them. You can also switch the various Entry Helper windows on or off, or switch to an open document window directly from the menu.
11.10.1 Cascade

This command rearranges all open document windows so that they are all cascaded (i.e. staggered) on top of each other.
11.10.2 Tile Horizontally

This command rearranges all open document windows as horizontal tiles, making them all visible at the same time.
11.10.3 Tile Vertically

This command rearranges all open document windows as **vertical tiles**, making them all visible at the same time.
11.10.4 Project Window

This command lets you switch the Project Window on or off.

This is a dockable window. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar, to allow docking or hide the window.
11.10.5 Info Window

This command lets you switch the Info Window on or off.

This is a dockable window. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar, to allow docking or hide the window.
11.10.6 Entry Helpers

This command lets you switch all three Entry-Helper Windows on or off.

All three Entry helpers are dockable windows. Dragging on a title bar detaches it from its current position and makes it a floating window. Click right on the title bar to allow docking or hide the window.
11.10.7 Output Windows

The Output Windows are a set of tabbed output windows, such as the Messages window (which displays messages like validation results), the Find in Files window, and the XPath window (which shows XPath evaluation results). The initial setting is for them to open at below the Main Window. The Output Windows command lets you switch the Output Windows on or off.

The Output Windows window is dockable. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar to allow docking or to hide the window.

For a complete description of Output Windows see Output Windows in the section, Text View.
11.10.8 Project and Entry Helpers

This command toggles on and off the display of the Project Window and the Entry Helpers together.
11.10.9 All On/Off

This command lets you switch all dockable windows on, or off:

- the Project Window
- the Info Window
- the three Entry-Helper Windows
- the Output Windows

This is useful if you want to hide all non-document windows quickly, to get the maximum viewing area for the document you are working on.
11.10.10 Currently Open Window List

This list shows all currently open windows, and lets you quickly switch between them.

You can also use the Ctrl-TAB or CTRL F6 keyboard shortcuts to cycle through the open windows.
11.11 Help Menu

The **Help** menu contains commands required to get help or more information about Authentic Desktop, as well as links to information and support pages on the Altova web server.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents...</td>
<td>F1</td>
</tr>
<tr>
<td>Index...</td>
<td></td>
</tr>
<tr>
<td>Search...</td>
<td></td>
</tr>
<tr>
<td>Keyboard Map...</td>
<td></td>
</tr>
<tr>
<td>Software Activation...</td>
<td></td>
</tr>
<tr>
<td>Order Form...</td>
<td></td>
</tr>
<tr>
<td>Registration...</td>
<td></td>
</tr>
<tr>
<td>Check for Updates...</td>
<td></td>
</tr>
<tr>
<td>Support Center...</td>
<td></td>
</tr>
<tr>
<td>FAQ on the Web...</td>
<td></td>
</tr>
<tr>
<td>Download Components and Free Tools...</td>
<td></td>
</tr>
<tr>
<td>Authentic on the Internet..</td>
<td></td>
</tr>
<tr>
<td>Authentic Training...</td>
<td></td>
</tr>
<tr>
<td>About Authentic...</td>
<td></td>
</tr>
</tbody>
</table>

The **Help** menu also contains the [Registration dialog](#), which lets you enter your license key-code once you have purchased the product.
11.11.1 Table of Contents, Index, Search

- **Table of Contents**
  
  *Description*
  
  Opens the onscreen help manual of Authentic Desktop with the Table of Contents displayed in the left-hand-side pane of the Help window. The Table of Contents provides an overview of the entire Help document. Clicking an entry in the Table of Contents takes you to that topic.

- **Index**
  
  *Description*
  
  Opens the onscreen help manual of Authentic Desktop with the Keyword Index displayed in the left-hand-side pane of the Help window. The index lists keywords and lets you navigate to a topic by double-clicking the keyword. If a keyword is linked to more than one topic, a list of these topics is displayed.

- **Search**
  
  *Description*
  
  Opens the onscreen help manual of Authentic Desktop with the Search dialog displayed in the left-hand-side pane of the Help window. To search for a term, enter the term in the input field, and press **Return**. The Help system performs a full-text search on the entire Help documentation and returns a list of hits. Double-click any item to display that item.
11.11.2 Keyboard Map

The **Help | Keyboard Map** command causes an information box to be displayed that contains a menu-by-menu listing of all commands in Authentic Desktop. Menu commands are listed with a description and shortcut keystrokes for the command.

<table>
<thead>
<tr>
<th>Command</th>
<th>Keys</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileClose</td>
<td></td>
<td>Close the active document</td>
</tr>
<tr>
<td>FileCloseAll</td>
<td></td>
<td>Close all open documents</td>
</tr>
<tr>
<td>FileEncoding</td>
<td></td>
<td>Set or change the character-set encoding for the cur...</td>
</tr>
<tr>
<td>FileExit</td>
<td></td>
<td>Quit the application; prompts to save documents</td>
</tr>
<tr>
<td>FileNew</td>
<td>Ctrl+N</td>
<td>Create a new document</td>
</tr>
<tr>
<td>FileOpen</td>
<td>Ctrl+O</td>
<td>Open an existing document</td>
</tr>
<tr>
<td>FileOpenURL</td>
<td></td>
<td>Open an existing document directly from a URL</td>
</tr>
<tr>
<td>FilePrint</td>
<td>Ctrl+P</td>
<td>Print the active document</td>
</tr>
<tr>
<td>FilePrintPreview</td>
<td></td>
<td>Print Preview</td>
</tr>
<tr>
<td>FilePrintSetup</td>
<td></td>
<td>Change the printer and printing options</td>
</tr>
<tr>
<td>FileReload</td>
<td></td>
<td>Reload open file</td>
</tr>
<tr>
<td>FileSave</td>
<td>Ctrl+S</td>
<td>Save the active document</td>
</tr>
<tr>
<td>FileSaveAll</td>
<td></td>
<td>Save all open documents</td>
</tr>
<tr>
<td>FileSaveAs</td>
<td></td>
<td>Save the active document with a new name</td>
</tr>
<tr>
<td>FileSaveToURL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileSendByMail</td>
<td></td>
<td>Send document by e-mail</td>
</tr>
</tbody>
</table>

To view commands in a particular menu, select the menu name in the Category combo box. You can print the command by clicking the printer icon.
11.11.3 Activation, Order Form, Registration, Updates

- Software Activation

**Description**

After you download your Altova product software, you can license—or activate—it using either a free evaluation key or a purchased permanent license key.

- **Free evaluation key.** When you first start the software after downloading and installing it, the Software Activation dialog will pop up. In it is a button to request a free evaluation key-code. Enter your name, company, and e-mail address in the dialog that appears, and click Request Now! The evaluation key is sent to the e-mail address you entered and should reach you in a few minutes. Now enter the key in the key-code field of the Software Activation dialog box and click OK to start working with your Altova product. The software will be unlocked for a period of 30 days.

- **Permanent license key.** The Software Activation dialog contains a button to purchase a permanent license key. Clicking this button takes you to Altova's online shop, where you can purchase a permanent license key for your product. There are two types of permanent license: single-user and multi-user. Both will be sent to you by e-mail. A single-user license contains your license-data and includes your name, company, e-mail, and key-code. A multi-user license contains your license-data and includes your company name and key-code. Note that your license agreement does not allow you to install more than the licensed number of copies of your Altova software on the computers in your organization (per-seat license). Please make sure that you enter the data required in the registration dialog exactly as given in your license e-mail.

**Note:** When you enter your license information in the Software Activation dialog, ensure that you enter the data exactly as given in your license e-mail. For multi-user licenses, each user should enter his or her own name in the Name field.

---

Your license email and the different ways to license (activate) your Altova product

The license email that you receive from Altova will contain:

- Your license details (name, company, email, key-code)
- As an attachment, a license file with a .altova_licenses file extension

To activate your Altova product, you can do one of the following:

- Enter the email-supplied license details in the Altova product's Software Activation dialog, and click OK.
- Save the license file (.altova_licenses) to a suitable location, double-click the license file, enter any requested details in the dialog that appears, and finish by clicking Apply Keys.
- Save the license file (.altova_licenses) to any suitable location, and upload it from this location to the license pool of your Altova LicenseServer. You can then either: (i) acquire the license from
Your Altova product via the product's Software Activation dialog, or (ii) assign the license to the product from Altova LicenseServer. For more information about licensing via LicenseServer, read the rest of this topic.

The Software Activation dialog (screenshot below) can be accessed at any time by clicking the Help | Software Activation command.

You can activate the software by either:

- Entering the license key information (click Enter a New Key Code), or
- Acquiring a license via an Altova LicenseServer on your network (click Use Altova LicenseServer, located at the bottom of the Software Activation dialog). Select the machine on which the LicenseServer you want to use has been installed. Note that the auto-discovery of License Servers works by means of a broadcast sent out on the LAN. As these broadcasts are limited to a subnet, License Server must be on the same subnet as the client machine for auto-discovery to work. If auto-discovery does not work, then type in the name of the server. The Altova LicenseServer must have a license for your Altova product in its license pool. If a license is available in the LicenseServer pool, this is indicated in the Software Activation dialog (screenshot below), and you can click Save to acquire the license.

After a machine-specific (aka installed) license has been acquired from a LicenseServer, it cannot be returned to the LicenseServer for a period of seven days. After that time, you can return the machine license to LicenseServer (click Return License) so that this license can be acquired from LicenseServer by another client. (A LicenseServer administrator, however, can unassign an acquired license at any time via the administrator's Web UI of LicenseServer.)
Note that the returning of licenses applies only to machine-specific licenses, not to concurrent licenses.

**Check out license**
You can check out a license from the license pool for a period of up to 30 days so that the license is stored on the product machine. This enables you to work offline, which is useful, for example, if you wish to work in an environment where there is no access to your Altova LicenseServer (such as when your Altova product is installed on a laptop and you are traveling). While the license is checked out, LicenseServer displays the license as being in use, and the license cannot be used by any other machine. The license automatically reverts to the checked-in state when the check-out period ends. Alternatively, a checked-out license can be checked in at any time via the **Check in** button of the Software Activation dialog.

To check out a license, do the following: (i) In the Software Activation dialog, click **Check out License** (see screenshot above); (ii) In the License Check-out dialog that appears, select the check-out period you want and click **Check out**. The license will be checked out. The Software Activation dialog will display the check-out information, including the time when the check-out period ends. The **Check out License** button in the dialog changes to a **Check In** button. You can check the license in again at any time by clicking **Check In**. Because the license automatically reverts to the checked-in status, make sure that the check-out period you select adequately covers the period during which you will be working offline.

**Note:** For license check-outs to be possible, it must be enabled on the LicenseServer. If this functionality has not been enabled, you will get an error message to this effect. In this event, contact your LicenseServer administrator.

**Copy Support Code**
Click **Copy Support Code** to copy license details to the clipboard. This is the data that you will need to provide when requesting support via the online support form.

Altova LicenseServer provides IT administrators with a real-time overview of all Altova licenses on a network, together with the details of each license, as well as client assignments and client usage of licenses. The advantage of using LicenseServer therefore lies in administrative features it offers for large-volume Altova license management. Altova LicenseServer is available free of cost from the Altova website. For more information about Altova LicenseServer and licensing via Altova LicenseServer, see the Altova LicenseServer documentation.

**Order Form**

**Description**
When you are ready to order a licensed version of the software product, you can use either the **Order license key** button in the Software Activation dialog (see previous section) or the **Help | Order Form** command to proceed to the secure Altova Online Shop.
Registration

- Description
  Opens the Altova Product Registration page in a tab of your browser. Registering your Altova software will help ensure that you are always kept up to date with the latest product information.

Check for Updates

- Description
  Checks with the Altova server whether a newer version than yours is currently available and displays a message accordingly.
11.11.4 Other Commands

- Support Center
  - Description
  A link to the Altova Support Center on the Internet. The Support Center provides FAQs, discussion forums where problems are discussed, and access to Altova's technical support staff.

- FAQ on the Web
  - Description
  A link to Altova's FAQ database on the Internet. The FAQ database is constantly updated as Altova support staff encounter new issues raised by customers.

- Download Components and Free Tools
  - Description
  A link to Altova's Component Download Center on the Internet. From here you can download a variety of companion software to use with Altova products. Such software ranges from XSLT and XSL-FO processors to Application Server Platforms. The software available at the Component Download Center is typically free of charge.

- Authentic Desktop on the Internet
  - Description
  A link to the Altova website on the Internet. You can learn more about Authentic Desktop and related technologies and products at the Altova website.

- About Authentic Desktop
  - Description
  Displays the splash window and version number of your product. If you are using the 64-bit version of Authentic Desktop, this is indicated with the suffix (x64) after the application name. There is no suffix for the 32-bit version.
11.12 Command Line

Certain Authentic Desktop actions can be carried out from the command line. These commands are listed below:

Open a file

Command: `authentic.exe file.xml`

*Action*: Opens the file, *file.xml*, in Authentic Desktop

Open multiple files

Command: `authentic.exe file1.xml file2.xml`

*Action*: Opens the files, *file1.xml* and *file2.xml*, in Authentic Desktop

Assign an SPS file to an XML file for Authentic View editing

Command: `authentic.exe myxml.xml /sps mysps.sps`

*Action*: Opens the file, *myxml.xml* in Authentic View with *mysps.sps* as its SPS file. The /sps flag specifies that the SPS file that follows is to be used with the XML file that precedes the /sps flag (for Authentic View editing).

Open a new XML template file via an SPS file

Command: `authentic.exe mysps.sps`

*Action*: Opens a new XML file in Authentic View. The display will be based on the SPS and the new XML file will have a skeletal structure based on the SPS schema. The name of the newly created XML file must be assigned when saving the XML file.
Chapter 3
Programmers' Reference
Programmers' Reference

Authentic Desktop is an Automation Server. It exposes programmable objects to other applications called Automation Clients. As a result, an Automation Client can directly access the objects and functionality that the Automation Server makes available. An Automation Client of Authentic Desktop, can use the XML validation functionality of Authentic Desktop. Developers can thus enhance their applications with the ready-made functionality of Authentic Desktop.

The programmable objects of Authentic Desktop are made available to Automation Clients via the Application API of Authentic Desktop, which is a COM API. The object model of the API and a complete description of all available objects are provided in this documentation (see the section Application API).

The API can be accessed from within the following environments:

- **Scripting Editor**
- **IDE Plug-ins**
- **External programs**
- **ActiveX Integration**

Each of these environments is described briefly below.

**Scripting Editor: Customizing and modifying Authentic Desktop functionality**

You can customize your installation of Authentic Desktop by modifying and adding functionality to it. You can also create Forms for user input and modify the user interface so that it contains new menu commands and toolbar shortcuts. All these features are achieved by writing scripts that interact with objects of the Application API. To aid you in carrying out these tasks efficiently, Authentic Desktop offers you an in-built Scripting Editor. A complete description of the functionality available in the Scripting Editor and how it is to be used is given in the **Scripting Editor** section of this documentation. The supported programming languages are **JScript** and **VBScript**.

**IDE Plug-ins: Creating plug-ins for Authentic Desktop**

Authentic Desktop enables you to create your own plug-ins and integrate them into Authentic Desktop. You can do this using Authentic Desktop's special interface for plug-ins. A description of how to create plug-ins is given in the section **Authentic Desktop IDE Plug-ins**.

An application object gets passed to most methods that must be implemented by an IDE plug-in and gets called by the application. Typical languages used to implement an IDE plug-in are **C#** and **C++**. For more information, see the section **Authentic Desktop IDE Plugins**.

**External programs**

Additionally, you can manipulate Authentic Desktop with external scripts. For example, you could write a script to open Authentic Desktop at a given time, then open an XML file in Authentic Desktop, validate the file, and print it out. External scripts would again make use of the Application API to carry out these tasks. For a description of the Application API, see the section **Application API**.
Using the Application API from outside Authentic Desktop requires an instance of Authentic Desktop to be started first. How this is done depends on the programming language used. See the section, Programming Languages, for information about individual languages.

Essentially, Authentic Desktop will be started via its COM registration. Then the Application object associated with the Authentic Desktop instance is returned. Depending on the COM settings, an object associated with an already running Authentic Desktop can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript and VBScript script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C# is a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#.
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for Applications, Perl, and Python.

**ActiveX Integration**
A special case of accessing the Application API is via the Authentic Desktop ActiveX control. This feature is only available if the Authentic Desktop integration package is installed. Every ActiveX Control has a property that returns a corresponding COM object for its underlying functionality. The manager control provides an Application object, the document control a Document object, and the placeholder object, in cases where it contains the project tree, returns the Project object. The methods supported by these objects are exactly as described in the Interfaces section of the Application API. Care must be taken not to use methods that do not make sense in the context of ActiveX control integration. For details see ActiveX Integration.

**About Programmers' Reference**
The documentation contained in the Programmers’ Reference for Authentic Desktop consists of the following sections:

- **Scripting Editor**: a user reference for the Scripting Environment available in Authentic Desktop
- **IDE Plug-ins**: a description of how to create plug-ins for Authentic Desktop
- **Application API**: a reference for the Application API
- **ActiveX Integration**: a guide and reference for how to integrate the Authentic Desktop GUI and Authentic Desktop functionality using an ActiveX control
1 **Scripting Editor**

The Scripting Editor of Authentic Desktop uses the Form Editor components of the Microsoft .NET Framework, and thus provides access to the Microsoft .NET Framework. This means that JScripts and VBScripts not only work with the Authentic Desktop API—which is a COM API and the API of Authentic Desktop—but can also access and use classes of the Microsoft .NET framework.

You can therefore create and use your own macros and forms within Authentic Desktop, and thus add to and modify the functionality of your installation of Authentic Desktop.

**Note:** Visual Basic is not supported as a language in the scripting environment. Only VBScript and JavaScript are. Ensure that you use VBScript syntax and not Visual Basic syntax in the scripting environment.

**Note:** Microsoft’s .NET Framework 2.0 or higher is a system prerequisite for Scripting Editor, and it must be installed before Authentic Desktop is installed.

The **Scripting Editor** (screenshot below) opens in a separate window and is accessed via the **Tools | Scripting Editor** menu command in the Authentic Desktop GUI. The programming languages that can be used in the Scripting Environment are **JScript** and **VBScript**. The scripting language can be changed by right-clicking the Project item in the Project window, selecting **Scripting Language**, and selecting the language you want.
What you can do with the Scripting Editor

In the Scripting Editor, you can create Forms, Event Handlers, and Macros to build up a Scripting Project. A Scripting Project can then be set as the Global Scripting Project for Authentic Desktop, thus enabling scripts in the Scripting Project to be used in the application. Additionally, different Scripting Projects can be assigned to different Authentic Desktop projects, thus allowing different scripts to be used for different Authentic Desktop projects.

Every script project can define the .NET runtime version it wants to use. An application can handle multiple scripting projects with different .NET runtime versions simultaneously, but the appropriate .NET version must be installed. For example, script projects with .NET 4.0 will only run on computers having .NET 4.0 installed.

Documentation about the Scripting Editor

The documentation describing the Scripting Environment (this section) is organized into the following parts:

- **An overview**, which provides a high level description of the Scripting Editor and Scripting Projects.
- **A list of steps required to create a Scripting Project**.
- **An explanation of Global Declarations**, together with an example.
- **A description of how to create Forms**.
- **A discussion of Authentic Desktop-specific event handlers**.
- **An explanation of how to use macros** in the Scripting Editor and in Authentic Desktop.
1.1 Overview

The Scripting Editor provides an interface in which you can: (i) graphically design Forms while assigning scripts for components in the Form; (ii) create Event Handlers, and (iii) create Macros.

These Forms, Event Handlers, and Macros are organized into scripting projects, which are then assigned to Authentic Desktop application projects and can be used in the application.

Variables and functions can be defined in a Global Declarations script, which is always executed before Macro or Event Handler scripts.

This section gives an overview of the Scripting Editor and Scripting Projects. It is organized into the following sections:

- **Scripting Projects in Authentic Desktop**, which describes how the scripting projects you create with the Scripting Editor will be used in Authentic Desktop.
- **The Scripting Editor GUI**, which provides a detailed look at the different parts of the Scripting Editor GUI and how they are to be used.
- **Components of a Scripting Project**, which explains the different components that go to make up a scripting project.

The details about the creation of the various components (*Global Declarations*, *Forms*, *Event Handlers*, and *Macros*) are described in their respective sections.

**.NET assemblies**

Every scripting project can have references to .NET assemblies—in addition to the default references. .NET assemblies can be added for the whole scripting project or for individual macros (by using the new CLR.LoadAssembly command in the source code; see *Built-in Commands*).

Assemblies can be added, for example, from the Global Assembly Cache.

To add an assembly, right-click the project or macro, and, from the context menu that pops up, select **Add .NET Assembly | Assembly from Global Cache (GAC)**.

This works in the same way as with Visual Studio and allows access not only to the complete Microsoft .NET Framework but also to any user-defined assembly.
1.1.1 **Scripting Projects in Authentic Desktop**

All scripts and scripting information created in the Scripting Editor are stored in **Altova Scripting Projects** (.asprj files).

You can create any number of Altova Scripting Projects. After a scripting project has been created, it can be used in the following ways:

- It can be set as the global scripting project for Authentic Desktop. Scripts in the global scripting project can then be called from within the application, and macros of the Global Scripting Project can be used for all Authentic Desktop projects.
- It can be assigned to an Authentic Desktop project (as an application project). When an Authentic Desktop project is open in Authentic Desktop, scripts in the associated scripting project can be called.

Your Authentic Desktop package contains a sample scripting project called **SampleScripts.asprj**. This file contains global declarations for a few standard tasks and is located in the folder: `C:\Users\<username>\Documents\Altova\Authentic Desktop2019\AuthenticExamples`.

**Setting the global scripting project of an application**
The global scripting project of an application is set in the Scripting tab of the Options dialog of Authentic Desktop (screenshot below, **Tools | Options**).

To set a global scripting project for Authentic Desktop, check the **Activate Scripting** check box and then browse for the Altova Scripting Project (.asprj) file you want. You can also specify: (i) whether Auto-Macros in the scripting project should be automatically executed when Authentic Desktop starts, and (ii) whether application event handler scripts in the project should be automatically executed or not; check or uncheck the respective check boxes accordingly.

**Note:** Nested script execution is possible, i.e. Macros can call other macros, and events are received during macro, or event, execution.

**Assigning a scripting project to an Authentic Desktop project**
A scripting project is assigned to an Authentic Desktop project as follows:
1. In the Authentic Desktop GUI, open the required application project.
2. Select the menu command **Project | Script Settings**. The Scripting dialog (screenshot below) opens.

![Scripting Dialog](image)

3. Check the **Activate Project Scripts** check box and select the required scripting project (.asprj file). If you wish to run Auto-Macros when the Authentic Desktop project is loaded, check the **Run Auto-Macros when Authentic Desktop project is loaded** check box.

4. Click **OK** to finish.

**Note:** To deactivate (that is, unassign) the scripting project of an Authentic Desktop project, uncheck the **Activate Project Scripts** check box.
1.1.2 The Scripting Editor GUI

The Scripting Editor GUI is shown below. It has the following parts:

- A toolbar
- A Scripting Project Tree pane (top left-hand side)
- A Properties and Events pane (bottom left)
- A Main Window with Design and Source tabs
- A Form Object Palette (right-hand side)

Scripting Editor toolbar

The Scripting Editor toolbar contains icons for:

- Standard file commands such as **New**, **Open**, **Save**, and **Print**. These commands are used to create new scripting projects, open existing scripting projects, and save and print scripting projects.
- Standard editing commands such as **Copy**, **Paste**, **Undo**, **Redo**, **Find**, and **Replace**. Note that the **Find** and **Replace** commands are applied to code in the Source tab of the Scripting Editor.

Scripting Project Tree

The Scripting Project Tree (**screenshot below**) shows the various components of the scripting project, structured along four main branches: (i) Global Declarations, (ii) Forms, (iii) Events, and (iv) Macros.
The Scripting Project Tree provides access to each component of the scripting project. For example, in order to display and edit a particular Form, expand the Forms folder in the tree (see screenshot above), right-click the Form you wish to display or edit, and click Open from the context menu that pops up.

A quicker way to open a Form, Event, macro, or the Global Declarations script, is to double-click the respective icon, or text. To delete a Form or Macro from the scripting project, right-click the component and select the Delete command from the context menu.

The Scripting Project Tree pane contains a toolbar with icons (screenshot below).

The icons, from left to right, are for: (i) creating a new macro, (ii) creating a new form, (iii) running a macro, and (iv) debugging a macro. These commands are also available in the context menu that appears when you right-click any component in the Scripting Project Tree.

**Properties and Events**
The Properties and Events pane (screenshot below) displays the following:

- Form properties, when the Form is selected
- Object properties, when an object in a Form is selected. (The screenshot below shows, at left, the properties of the object selected in the Form at right.)
- Form events, when a Form is selected
- Object events, when an object in a Form is selected
To switch between the properties and events of the selected component, click, respectively, the **Properties** icon (third from left in the Properties and Events toolbar, *see screenshot above*) and the **Events** icon (fourth from left).

The first and second icons from left in the toolbar are, respectively, the **Categorized** and **Alphabetical** icons. These display the properties or events either organized by category or organized in ascending alphabetical order.

When a property or event is selected, a short description of it is displayed at the bottom of the Properties and Events pane.

**Main Window**

The Main Window displays one component at a time and has one or two tabs depending on what is being displayed. If a Global Declarations script, an Event, or a Macro is being displayed, then a single tab, the Source tab, displays the source code of the selected component.

The Source tab supports:

- syntax coloring
- source code folding
- setting/deleting bookmarks using **CTRL+F2**
- autocompletion entry helper with parameter info
- Goto Brace, Goto Brace Extend
- Zoom In / Zoom Out
- full method/property signature shown next to the autocompletion entry helper

- brace highlighting during code entry
  
  ```javascript
  if ( x == y.GetNames(a, b, c) )
  ```

- mouse over popups; placing the mouse over a known method or property, displays its signature (and documentation if available)

If a **Form** is being displayed, then the Main Window has two tabs: a Design tab showing and enabling the layout of the Form, and a Source tab containing the source code for the Form. Content in both the Design tab and Source tab can be edited.

**Note:** Since JScript and VB Script are untyped languages, entry helpers and auto-completion is supported only in cases of "fully qualified constructs" and "predefined" names.

If names start with `objDocument`, `objProject`, `objXMLData`, or `objAuthenticRange`, members of the corresponding interface will be shown. Auto-completion entry helper and parameter info are shown during editing, but can also be obtained on demand by pressing **Ctrl+Space**.

**Form Object Palette**

The Form Object Palette contains all the objects that are available for designing Forms and looks something like the screenshot below. Registered ActiveX controls can be added to the Form Object Palette by right-clicking the pane and selecting the **Add ActiveX Control** command.
To insert an object from the Form Object Palette click the object you want in the palette, then click at the location in the Form where you wish to insert the object. The object will be placed at this location. In many cases you will need to supply some properties of the object via the Properties and Events pane. You can drag the object to other locations as well as resize it. Further, a number of editing commands, such as centering and stacking objects, can be accessed via the context menu of the selected Form object.

Some Form objects, such as Timer, are not added to the Form but are created as Tray Components in a tray at the bottom of the Main Window. You can select the object in the tray and set properties and event handlers for the object via the Properties and Events pane. For an example of how Tray Components are handled, see Form usage and commands.
1.1.3 Components of a Scripting Project

An Altova Scripting Project consists of the following four major components:

- **Global Declarations**, a component which contains definitions of variables and functions that are available to, and can be used by, all Forms, Macros, and Event Handler scripts in the scripting project.

- **Forms**, a component which contains all the Forms defined in the scripting project.

- **Events**, a component which contains Event Handler scripts for all application-based—as opposed to Form-based—events.

- **Macros**, a component which contains all the Macros defined in the scripting project.

These components are displayed in and accessed via the Scripting Project Tree of the Scripting Editor (screenshot below).

![Scripting Project Tree](screenshot below)

Given below is a brief description of each of these components.

**Global Declarations**

The Global Declarations component is a script that contains variables and functions that can be used by Forms, Event Handlers, and Macros. The functions make use of the XMLSpy API to access Authentic Desktop functionality. Creating a variable or function in the Global Declarations module enables it to be accessed from all the Forms, Event Handlers and Macros in the scripting project.

To add a variable or function, open the Global Declarations component (by right-clicking it in the Scripting Project Tree and selecting **Open**) and edit the Global Declarations script in the Main Window. In this script, add the required variable or function.

**Forms**

In the Scripting Editor, you can build a Form graphically using a palette of Form objects such as text input fields and buttons. For example, you can create a Form to accept the input of an element name and to then remove all occurrences of that element from the active XML document.

For such a Form, a function script can be associated with a text box so as to take an input variable, and an Event Handler can be associated with a button to start execution of the delete functionality, which is available in the XMLSpy API. A Form is invoked by a call to it either within a function (in the Global Declarations script) or directly in a Macro. For details of how to create and edit Forms, see the **Forms** section.

**Event handling**

Event Handler scripts can be associated with a variety of available events. You can control events
that occur both within Forms (Form events) and within the general application interface (application events). The script associated with an event is executed immediately upon the triggering of that event.

Most events have parameters which provide detailed information about the event. The return value from the script typically instructs the application about how to continue its processing (for example, the application may not allow editing).

An Event Handler runs when the relevant event occurs in the Form or in Authentic Desktop. For details about how to create event handlers, see Event Handlers.

**Macros**

Macros are used to implement complex or repetitive tasks. Macros do not use either parameters or return values.

In a Macro, it is possible to access all variables and functions declared in the Global Declarations and to display Forms for user input.

For a simple example of creating a Macro, see Writing a Macro. Also see Running Macros for a description of the ways in which a Macro can be called. A Macro is run from within the Authentic Desktop interface by clicking Tools | Macros | [MacroName]
1.2 Creating a Scripting Project

The broad steps for creating a Scripting Project are as follows:

1. Open the Scripting Editor by clicking the command Tools | Scripting Editor.
2. In the Scripting Editor, open a new scripting project by clicking the New icon in the Scripting Editor toolbar. The Project Settings dialog (screenshot below) pops up. You can also access this dialog by right-clicking a project in the Scripting Project Tree pane (in the top left part of the Scripting Editor window) and clicking the command Project Settings.

   ![Project Settings dialog]

   Select either JScript or VBScript in the first combo box and the .NET Framework in the second combo box. To enable higher .NET Frameworks (such as .NET Framework 4.5 on Windows 8), check the Automatically use higher .NET Framework when specified Target Framework is not available on target computer check box. Then click OK. The new Scripting Project is created.

3. Click the Save icon in the Scripting Editor toolbar to save the Scripting Project as a .asprj file.
4. A Scripting Project can be considered to be made up of several components that work together. These components will typically be a combination of: Global Declarations, Forms, Events, and Macros. They can be created in any order, but you should clearly understand how they work together. The way each type of component is called and executed is described below. How to create each type of component is described in the respective sections about the component type.
5. After you have finished creating all the required components, save the Scripting Project (by clicking the Save icon in the Scripting Editor toolbar).
6. Close the Scripting Editor.

**Note:** Right-clicking the Project folder and selecting Project Settings lets you change the scripting language at any time.

**How Forms, Event Handlers, and Macros are called and executed**
Forms, Event Handlers, and Macros are all created in the Scripting Editor. However, the way they are called and executed is different for each and has a bearing on how you create your scripting projects.

- A Form is invoked by a call to it either within a function in the Global Declarations script or directly in a Macro.
- An Event Handler runs when the relevant event occurs in Authentic Desktop. If an Event Handler for a single event is defined in both the Global Scripting Project and the Authentic Desktop-project-specific Scripting Project, then the event handler for the project-specific Scripting Project is executed first and that for the Global Scripting Project immediately afterwards.
- A Macro is executed from within the Authentic Desktop interface by clicking **Tools | Macros | [MacroName]**. In a Macro, it is possible to access all variables and functions declared in the Global Declarations and to display Forms for user input.
### 1.3 Global Declarations

The Global Declarations component is present by default in every Scripting Project (see screenshot below), and therefore does not have to be created. In order to add variables and functions to the Global Declarations script of a Scripting Project, you need to open the Global Declarations script and add the code fragment to the Global Declarations script. See Components of a Scripting Project and Creating a Scripting Project for more information.

To open the Global Declarations script of a Scripting Project, right-click the Global Declarations item in the Scripting Project Tree (screenshot above), and select Open. The Global Declarations script opens in the Main Window.

**Note:** Every time a macro is executed or an event handler is called, global declarations are re-initialized.

Given below is an example function. Remember that creating a variable or function in the Global Declarations script makes this variable or function accessible to all Forms, Event Handlers, and Macros.

**Example function**

A function called `RemoveAllNamespaces` would have code like this:

```javascript
function RemoveAllNamespaces(objXMLData)
{
    if(objXMLData == null)
        return;

    if(objXMLData.HasChildren) {
        var objChild;

        // spyXMLDataElement := 4
        objChild = objXMLData.GetFirstChild(4);

        while(objChild) {
            RemoveAllNamespaces(objChild);

            try{
                var nPos,txtName;
                txtName = objChild.Name;

                if((nPos = txtName.indexOf(":")) >= 0) {
                    objChild.Name = txtName.substring(nPos+1);
                }
            }
        }
    }
}
```
objChild = objXMLData.GetNextChild();
}
    **catch** (Err) {
        objChild = null;
    }
}
}

**Note:**

- It is possible to define local variables and helper functions within macros and event handlers. Example:

```javascript
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper() {
    txtMessage = txtLocal;
    Application.ShowForm("MsgBox");
}
function On_BeforeStartEditing(objXMLData) {
    txtLocal = "On_BeforeStartEditing()";
    Helper();
}
```

- Recursive functions are supported.
1.4 Forms

Creating and editing Forms in the Scripting Editor consists of the following steps:

1. **Creating a New Form.** The new Form is created and named, and has properties defined for it.
2. **Designing the Form.** A Form is designed by adding Form Objects to it and assigning values for the different Form Objects.
3. **Scripting Form Events.** Scripts are assigned to Form-related events.
1.4.1 Creating a New Form

Creating a new Form in the Scripting Editor involves the following steps:

1. Creating a new Form and naming it
2. Specifying the properties of the Form

Creating a new Form and naming it
To add a new Form to a scripting project, click the Add Form icon (highlighted in screenshot below) in the toolbar of the Project Overview pane. Enter the name of the new Form.

A new Form is added to the project. It appears in the Main Window and an entry for it is created in the Scripting Project Tree pane, under the Forms heading. Press the F2 function key to rename the form, or right click the form name and select Rename from the context menu. In the screenshot below, we have named the new Form Registration.

Form properties
The properties of the Form, such as its size, background color, and font properties, can be set in the Properties pane. The screenshot below shows the size and background-color property values in bold, in the Layout and Appearance categories, respectively.
Testing a Form

You can test a form in the Scripting Editor by right-clicking it in the Project Overview pane and selecting the **Test Form** Command.
1.4.2 Form Design and Form Objects

Designing a Form consists of the following steps:

- Placing an object from the Form Object Palette in the Form design.
- Assigning values for the properties of individual Form Objects.
- Assigning scripts for Form-based events.

The Form Object Palette

The Form Object Palette contains all the objects that are available for designing Forms and looks something like the screenshot below. Registered ActiveX controls can be added to the Form Object Palette by right-clicking the pane and selecting the Add ActiveX Control command.

To insert an object from the Form Object Palette click the object you want in the palette, then click at the location in the Form where you wish to insert the object. The object will be placed at this location. In many cases you will need to supply some properties of the object via the Properties and Events pane. You can drag the object to other locations as well as resize it. Further, a number of editing commands, such as centering and stacking objects, can be accessed via the context menu of the selected Form object.

Some Form objects, such as Timer, are not added to the Form but are created as Tray Components in a tray at the bottom of the Main Window. You can select the object in the tray and set properties and event handlers for the object via the Properties and Events pane. For an example of how Tray Components are handled, see Form usage and commands.

Some of the most commonly used objects are described below:

- **Label**: Adds text fields such as captions or field descriptions.
- **Button**: Adds a button. It is possible to assign bitmaps as background images for these buttons.
Check Box: Adds a check box, which enables Yes/No type selections.

Combo Box: Adds a combo box, which allows the user to select an option from a drop-down menu.

List Box: Adds a list box, which displays a list of items for selection.

TextBox: Enables the user to enter a single line of text.

Rich TextBox: Enables the user to enter multiple lines of text.

Creating objects and setting their properties
To create an object in the Form, first select the required object in the Form Object Palette and then click the location in the Form where you want to insert it. After the object has been inserted, you can resize it as well as drag it to another location in the Form.

When an object is selected in the design, you can specify its properties in the Properties and Events pane. In the toolbar of the Properties and Events pane, click the Properties icon to display a list of the object's properties.

For example, in the screenshot below, the Label object with the text Start Date has been selected in the design. In the Properties and Events pane, the name of the object (which is the name that is to be used to identify the object in code, Label1 in the screenshot below) is given in the Design category of properties; in this case, the name of the object is Label1.

The text of the label (which is what appears in the Form) must be entered as the value of the Text property in the Appearance category of properties.

To assign other object properties, enter values for them in the Properties and Events pane.

Testing a Form
You can test a form in the Scripting Editor by right-clicking it in the Project Overview pane and selecting the **Test Form** Command.
1.4.3 Form Events

When an object is selected in the design, clicking on the Events icon in the toolbar of the Properties and Events pane (fourth icon from left), displays all the events available for that object (see screenshot below). These can be displayed either by category (screenshot below) or alphabetically.

For each event, you can enter the name of an existing event handler or function. Alternatively:

- you can double click on an event to create: (i) an empty function script in the Source tab of the Main Window, and (ii) an association of the newly created function with the selected event.
- double click a button in the design tab, to directly generate the handler stub in the code window.

The screenshot below was taken after the Click event was double-clicked. Notice that an empty event handler function called FormExample_Label1_Click has been created in the Main Window and that, in the Properties and Events pane, this function has been associated with the Click event.

Enter the required scripting code and save the project.

Writing the required scripts
After the visual design of the form is complete, form objects will typically be associated with suitable scripts. The example below is a script that adds colors when a button is clicked. The script is inserted as an event handler for the `Click` event of the button `Button1` (the event is available in the Properties and Events pane when the button is selected in the design):

```javascript
function FormExample_Button1_Click( objSender, e_EventArgs )
{
    // Sets the ForeColor (red) of the button.
    // Sets the BackColor (blue) of the button.
    // Sets the form BackColor (green).
}
```
1.5 Events

The Events folder of the scripting project (see screenshot below) contains folders for the following type of events:

- Application Events
- Document Events
- Authentic View Events
- Grid View Events
- Text View Events

Note that these events are Authentic Desktop-specific, as opposed to Form-based events. Each of the folders listed above contains a set of events for which Event Handler scripts can be written.

Application Events, for example, are shown in the screenshot below.

To access the event handler script of any of these events, right-click the event and select **Open** from the context menu. The script will be displayed in the Main Window (see screenshot below) and can be edited there. After you have finished editing the script, save changes by clicking the **Save** command in the toolbar of the Scripting Editor.

```
//objApplication: the XMLSpy application object
function OnInitApplication( objApplication )
{

}
```

Note the following points:

- Event Handlers need function headers with the correct spelling of the event name. Otherwise the Event Handler will not be called.
- It is possible to define local variables and helper functions within Macros and Event Handlers. Example:

  ```
  //return value: true allows editing
  //return value: false disallows editing
  var txtLocal;
  function Helper()
  {
    txtMessage = txtLocal;
  }
  ```
In order for events to be processed, the Process Events options must be toggled on in the Scriptings options of Authentic Desktop. See Scripting Projects in Authentic Desktop for details.

Also see Programming Points.

Application Events

OnInitialize
The OnInitialize event is raised after the main window becomes visible but before any project is loaded. This event is not raised if the application can't be loaded at all.

OnRunning
If the application is completely loaded and after the OnInitialize event occurs, the OnRunning event is raised.

OnShutdown
The event is raised after any open project and all documents have been closed on shutdown of the application. The main window is no longer visible.

Example
The following script is an Event Handler for the On_BeforeOpenProject event. It allows you to add a script that will be executed each time before Authentic Desktop opens a project. The example script below sequentially opens all XML files located in the XML folder of the project and validates them. If the validation fails, the script shows the validation error and stops. If a file passes the validity test, it will be closed and the next file will be opened.

Enter the following script for the On_BeforeOpenProject() event, and then save the scripting project.

```javascript
function On_BeforeOpenProject() {
  var bOK;
  var nIndex, nCount;
  var objItems, objXMLFolder = null;

  objItems = Application.CurrentProject.RootItems;
  nCount = objItems.Count;

  // search for XML folder
  for (nIndex = 1; nIndex <= nCount; nIndex++) {
    var txtExtensions;
    txtExtensions = objItems.Item(nIndex).FileExtensions;
    if (txtExtensions.indexOf("xml") >= 0) {
      objXMLFolder = objItems.Item(nIndex);
      Application.ShowForm("MsgBox");
    }
  }
}
```
break;
}
}

// does XML folder exist?
if(objXMLFolder){
    var objChild, objDoc;

    nCount = objXMLFolder.ChildItems.Count;

    // step through associated xml files
    for(nIndex = 1;nIndex <= nCount;nIndex++) {
        objChild = objXMLFolder.ChildItems.Item(nIndex);

        try{
            objDoc = objChild.Open();

            // use JScript method to access out-parameters
            var strError = new Array(1);
            var nErrorPos = new Array(1);
            var objBadData = new Array(1);

            bOK = objDoc.IsValid(strError,nErrorPos,objBadData);

            if(!bOK) {
                // if the validation fails, we should display the
                // message from XMLSpy
                // of course we have to create the form "MsgBox" and
                // define the global txtMessage variable
                //
                // txtMessage = Position:" + nErrorPos[0] + "," + strError[0];
                // txtMessage +=\"\nXML:\" + objBadData[0].Name +", " +
                // objBadData[0].TextValue;
                //
                // Application.ShowForm("MsgBox");

                break;
            }

            objDoc.Close(true);
            objDoc = null;
        }
        catch(Err) {
            // displaying the error description here is a good idea

            // txtMessage = Err.Description;
            // Application.ShowForm("MsgBox");

            break;
        }
    }
}
Testing the Event Handler
Switch to Authentic Desktop, and open a project to see how the BeforeOpenProject event is handled.
1.6 Macros

Macros automate repetitive or complex tasks. In the Scripting Environment, you can create a script that calls application functions as well as custom functions that you have defined. This flexibility provides you with a powerful method of automating tasks within Authentic Desktop. This section about macros is organized as follows:

- **Creating and Editing a Macro** describes how to create a new macro and edit an existing one.
- **Running a Macro** explains how a macro can be run from the Scripting Editor and from the broader Authentic Desktop environment as well.
- **Debugging** describes how macros can be debugged.

**Key points about macros**

Given below is a summary of important points about macros.

- Any number of macros can be added to the active scripting project. These macros are saved in the Altova Scripting Project file (.asprj file).
- Functions that are used in a macro can be saved as a Global Declaration. All Global Declarations are also saved in the Altova scripting project file (.asprj file).
- The macro can be tested by running it from within the Scripting Editor, and it can be debugged from within the Scripting Editor.
- Authentic Desktop can have one global Scripting Project, and a second scripting project, assigned to the currently loaded project, active at any one time; the macros are available to both of them. See **Running a Macro** for details.
1.6.1 Creating and Editing a Macro

The following operations enable you to create a new macro and edit an existing macro.

Creating a new macro

Right-click the Macro folder in the Scripting Projects tree and select Add Macro from the context menu. (The Add Macro command can also be selected from the context menu of any item in the Scripting Projects tree.) Alternatively, click the New Macro icon in the toolbar of the Scripting Projects tree.

The newly created (and empty) macro document is displayed in the Main Window, and the name of the macro is displayed in the title bar of the Scripting Editor (screenshot below).

Naming or renaming a macro

To name or rename a macro, click the macro name in the Scripting Project tree and press the F2 function key, or right click the name and select Rename from the context menu.

Opening a macro

To open a macro, right-click the macro in the Macros folder of the Scripting Project tree (see screenshot above), and select the Open command. The macro is displayed in the Main Window and its name is displayed in the title bar of the Scripting Editor (screenshot below). Alternatively, double-clicking a macro in the Scripting Project tree opens it in the Main Window.

Editing the macro

To edit a macro, enter or edit its code in the Main Window. For example, the following code creates the Form named Form1 in memory and then shows it. Form1 must already have been created (using the Scripting Editor's Form creation) before this macro is run.

```javascript
objForm = CreateForm('Form1');
objForm.ShowDialog();
```

The following macro uses the RemoveAllNamespaces function to remove all namespaces in the active XML document.

```javascript
if(Application.ActiveDocument != null) {
    RemoveAllNamespaces(Application.ActiveDocument.RootElement);
}
```
Application.ActiveDocument.UpdateViews();
}

The `RemoveAllNamespaces` function itself will have to be defined in the Global Declarations script. After the `RemoveAllNamespaces` function has been defined, the macro is complete and can be run.

**Note:** Macros do not support parameters or return values.

### Setting a macro as an Auto-Macro
When a macro is set as an Auto-Macro it can be run automatically when: (i) Authentic Desktop is started, or (ii) an Altova Authentic Desktop project is loaded in Authentic Desktop. To specify whether Auto-Macros should be run in each of these two events, check the *Run Auto-Macros* option in the Automatic Script Processing pane of the relevant dialogs:

- **When Authentic Desktop is started:** the Scripting tab of the Authentic Desktop Options dialog ([Tools | Options](#) menu command).
- **When an Authentic Desktop project is loaded into Authentic Desktop:** the Scripting dialog ([screenshot below](#), **Project | Scripting Settings** menu command).

To set a macro as an Auto-Macro, right-click the macro in the Scripting Project tree and select the command **Set as Auto-Macro**. This is a toggle command; so to remove the Auto-Macro setting of a macro, select the command again.
1.6.2 Running a Macro

To run a macro in the Scripting Editor, right-click the macro in the Scripting Project tree and select the command Run Macro.

There are different ways to run a macro from Authentic Desktop:

- Via the Tools | Macros menu of Authentic Desktop.
- By creating and using a toolbar button for a macro.
- By creating and using a menu item for a macro.

Note that only one macro can be run at a time. After a macro (or event) is executed, the script is closed and global variables lose their values.

The Authentic Desktop command to run Macros

The Tools | Macros menu command (screenshot below) opens a submenu containing the macros defined in the Scripting Project that is currently active in Authentic Desktop. The active Scripting Projects are specified in the Scripting tab of the Options dialog, or in the Scripting tab of the project settings.

From the submenu of available macros, select the macro to run. The macro will be executed.

Toolbar icon

You can create an icon in the toolbar or a menu item that runs a selected macro. To do this, click Tools | Customize | Macros. This causes the Customize dialog to be displayed (screenshot below).
Now do the following:

1. In the Macros tab of the Customize dialog, select the required macro from the Macros pane. The macros in the Macros pane are those in the active Scripting Project (which is specified in the Scripting tab of the Options dialog).
2. In the Display Text input field enter the name of the icon. This name will appear when the cursor is placed over the icon when it is in the toolbar.
3. Click Add Command to add it to the list of commands.
4. Select the command and click Edit Icon to create a new icon.
5. Drag the finished icon from the Associated Commands pane and drop it on to the toolbar or menu when the cursor changes from an arrow to an I-beam or line.
6. Macros can even be assigned their own shortcuts in the Keyboard tab of the Customize dialog (see screenshot above).

To remove the toolbar icon, open the Macros tab of the Customize dialog and drag the icon out of the toolbar and into the Associated Commands pane. Select the command in the Associated Commands pane and click Remove to remove the command from the pane.

**Item in the Tools menu**

The XMLSpy API includes a function, AddMacroMenuItem(), to add macros as menu items to the Tools menu. This function can be used to add one or more macros to the Tools | Macros list of macros. Typically, you should do this as follows:

1. Add the macro menu item by calling the XMLSpy API function, AddMacroMenuItem().

   ```csharp
   Application.AddMacroMenuItem("DeleteElements","Delete Elements Dialog");
   ```
• The function’s first parameter (*DeleteElements* in the example listing above) is the name of the macro. If you run the macro and there is an open project having scripts associated with it, Authentic Desktop searches for the macro in the project scripts first.
   If there are no project scripts, or if Authentic Desktop cannot find the macro, then it looks for the macro in the global scripts.

• The second parameter (*Delete Elements Dialog*) is the display text for the menu item.

2. Reset the **Tools** menu by calling `ClearMacroMenu()`. This removes all previously added menu items.

The best way to call these two functions is with the *Autorun* macro of the global scripting project or the *On_OpenProject* event.
1.6.3 Debugging a Macro

You can debug a macro using an installed debugger. To do this, right-click the macro in the Scripting Project tree and select the command **Debug Macro**.

This pops up the Just-In-Time Debugging dialog (*screenshot below*), which lists the debuggers available on the machine. Select the debugger you wish to use and click **Yes**.

![Just-In-Time Debugging](image)

The selected debugger starts.
1.7 Programming Points

The following programming points should be noted:

- All namespaces and types of the following .NET assemblies can be accessed in the Microsoft .NET Framework per default:

```
System
System.Data
System.Design
System.Drawing
System.Windows.Forms
System.XML
```

Additional assemblies can be added to the scripting project via the project's context menu, or dynamically (at runtime) in the source code by using `CLR.LoadAssembly`.

- Out-parameters from methods of the XMLSpy API require special variables in JScript. Given below are some examples.

```
// use JScript method to access out-parameters
var strError = new Array(1);
var nErrorPos = new Array(1);
var objBadData = new Array(1);
bOK = objDoc.IsValid(strError,nErrorPos,objBadData);END
```

- Out-parameters from methods of the .NET Framework require special variables in JScript. For example:

```
var dictionary =
CLR.Create( "System.Collections.Generic.Dictionary< System.String,
System.String >" );
dictionary.Add("1", "A");
dictionary.Add("2", "B");

// use JScript method to access out-parameters
var strOut = new Array(1);
if ( dictionary.TryGetvalue("1", strOut) ) // TryGetvalue will set the out parameter
    alert( strOut[0] ); // use out parameter
```

- .NET Methods that require integer arguments should not be called directly with JScript Number Objects which are Floating Point Values.

For example, instead of:

```
var objCustomColor = CLR.Static( "System.Drawing.Color" ).FromArgb( 128, 128, 128 );
```
use:

```javascript
  FromArgb( Math.floor( 128 ), Math.floor( 128 ), Math.floor( 128 ) );
```

- To iterate .NET collections the JScript Enumerator as well as the .NET iterator technologies can be used:

  For example:

  ```javascript
  // iterate using the JScript iterator
  var itr = new Enumerator( coll );
  for ( ; !itr.atEnd(); itr.moveNext() )
    alert( itr.item() );

  // iterate using the .NET iterator
  var itrNET = coll.GetEnumerator();
  while( itrNET.MoveNext() )
    alert( itrNET.Current );
  ```

- .NET templates can be instantiated as shown below:

  ```javascript
  var coll =
  CLR.Create( "System.Collections.Generic.List<System.String>" );
  ```

  or

  ```javascript
  CLR.Import( "System" );
  CLR.Import( "System.Collections.Generic" );
  var dictionary = CLR.Create( "Dictionary< String, Dictionary<String, String> >" );
  ```

- .NET Enum values are accessed as shown below:

  ```javascript
  var enumValStretch =
  ```

- Enumeration literals, as defined in the Altova type libraries, can now be used instead of numerical values.

  ```javascript
  objExportXMIFileDlg.XMIType = eXMI21ForUML23;
  ```
1.7.1 Built-in Commands

This section lists:

- **Built-in commands**
  - alert
  - conform
  - doevents
  - CreateForm
  - lastform
  - prompt
  - ShowForm
  - watchdog

- **.NET interoperability** commands
  - CLR.Create
  - CLR.Import
  - CLR.LoadAssembly
  - CLR.ShowImports
  - CLR.ShowLoadedAssemblies
  - CLR.Static

**Built-in commands**
The following built-in commands are available.

*ShowForm(strFormName : String)*  
Instantiates a New Form object from the given form name and immediately shows it as Dialog.  
*Return Value:* A Number that represents the generated DialogResult ([System.Windows.Forms.DialogResult]).

Example:
```javascript
var dialogResult = ShowForm("FormName");
```

Shows Form "FormName" as Dialog:

The DialogResult can be evaluated e.g. by:
```javascript
if (dialogResult ==
    CLR.Static("System.Windows.Forms.DialogResult").OK)
    alert("ok");
else
    alert("cancel");
```
CreateForm(strFormName : String)
Instantiates a New Form object from the given Form name.
Return Value: The Form object (System.Windows.Forms.Form) of the given name, or null if no Form with such name exists.

Example:
var myForm = CreateForm( "FormName" );
if ( myForm != null )
{
    myForm.textboxFirstName.Text = "Daniel";
    myForm.textboxLastName.Text = "Smith";
    var dialogResult = myForm.ShowDialog();
}

Shows Form "FormName" as Dialog - TextBoxes are initialized:

The DialogResult can be evaluated e.g. by:

if ( dialogResult ==
    alert( "ok" );
else
    alert( "cancel" );

lastform
This global field can be used to conveniently access the last form object that was created. 
Return Value: Returns a reference to the last form object (System.Windows.Forms.Form) 
that was successfully instantiated via CreateForm() or ShowForm().

Example:
CreateForm( "FormName" );
if ( lastform != null )
{
    lastform.textboxFirstName.Text = "Daniel";
    lastform.textboxLastName.Text = "Smith";
    var dialogResult = lastform.ShowDialog();
}

Shows Form "FormName" as Dialog - TextBoxes are initialized (similar to the CreateForm example above):
doevents()
Processes all Windows messages currently in the message queue.
Return Value: None

Example:
for ( i=0; i < nLongLastingProcess; ++i )
{
    // do long lasting process
    doevents();  // process windows messages; give UI a chance to update
}

watchdog(bEnable : boolean)
Long running CPU-intensive scripts cause the watchdog to ask the user if the script should be terminated. The `watchdog()` method is used to disable or enable this behavior. Per default the watchdog is enabled.
Return Value: None

Example:
watchdog( false );  // disable watchdog - we know the next statement is CPU intensive but it will terminate for sure
doCPUIntensiveScript();
watchdog( true );  // re-enable watchdog

Usage tip:
Calling watchdog(true) can also be used to reset the watchdog. This can be useful before executing long running (CPU intensive) tasks to ensure they have the maximum allowed script processing quota.

alert(strMessage : String) or MsgBox(strMessage : String)
An alert box is used to show a given message. The user will have to click "OK" to proceed.
Return Value: None

Example:
alert( "Hello World" );
confirm(strMessage : String)
Opens a dialog that shows the given confirm message.
A confirm box is often used to verify or accept something. The user will have to click either "OK"
or "Cancel" to proceed.
Return Value: A Boolean that represents the users answer. If the user clicks "OK", the dialog returns true, if the user clicks "Cancel", the dialog returns false.

Example:
```
if ( confirm( "Continue processing?" ) == false )
    return;
```

prompt(strMessage : String, strDefault : String)
Opens a dialog that shows the given prompt message and a TextBox control with a default answer.
A prompt box is often used to input a simple string value.
Return Value: A String that contains the TextBox value or null if the user selected "Cancel".

Example:
```
var name = prompt( "Please enter your name", "Daniel Smith" );
if ( name != null )
    alert( "Hello " + name + "!" );
```
.NET interoperability commands

To allow further interoperability with the .NET Framework additional functions are provided under CLR.

CLR.Import(strNamespaceCLR : String)

This is the scripting equivalent to the C# using / VB.Net imports keyword. This allows to leave out the given namespaces in successive calls like CLR.Create() and CLR.Static().

Return Value: None

Example:

Instead of always having to use full qualified names:

```csharp
if ( ShowForm( "FormName" ) == 
{
    var sName = lastform.textboxFirstName.Text + " " +
lastform.textboxLastName.Text;
sName );
}
```

one can import namespaces and use the short form:

```csharp
CLR.Import( "System.Windows.Forms" );
if ( ShowForm( "FormName" ) == CLR.Static( "DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " +
lastform.textboxLastName.Text;
    CLR.Static( "MessageBox" ).Show( "Hello " + sName );
}
```

Please note:

Importing a namespace does not add or load the corresponding assembly to the scripting project! Assemblies can be added to the scripting project dynamically (at runtime) in the source code by using CLR.LoadAssembly.

CLR.ShowImports()

Opens a MessageBox dialog that shows the currently imported namespaces. The user will have to click "OK" to proceed.

Return Value: None
Example:

CLR.ShowImports();

![ShowImports Info]

CLR.LoadAssembly(strAssemblyNameCLR : String)

Loads the .NET assembly with the given long assembly name or file path.

_Return Value:_ A Boolean value. True if the assembly could be loaded, false otherwise.

Example:

// set clipboard text (if possible)
// System.Windows.Clipboard is part of the PresentationCore assembly, so
// load this assembly first:
if ( CLR.LoadAssembly( "PresentationCore, Version=3.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35", true ) )
{
    var clipboard = CLR.Static( "System.Windows.Clipboard" );
    if ( clipboard != null )
        clipboard.SetText( "HelloClipboard" );
}

CLR.ShowLoadedAssemblies()

Opens a MessageBox dialog that shows the currently loaded assemblies. The user will have to

_click "OK" to proceed.

_Return Value:_ None

Example:

CLR.ShowLoadedAssemblies();
CLR.Create(strTypeNameCLR : String, constructor arguments ...)  
Creates a new .NET object instance for the given typename. If more than one argument is passed the successive arguments are interpreted as the arguments for the constructor of the .NET object.  
*Return Value:* A reference to the created .NET object  

Examples:  
```
var objArray = CLR.Create("System.Collections.ArrayList");

var newItem = CLR.Create( "System.Windows.Forms.ListViewItem", "NewItemText" );

var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
```

CLR.Import( "System" );  
CLR.Import( "System.Collections.Generic" );  
var dictionary = CLR.Create( "Dictionary< String, Dictionary< String, String > >" );

CLR.Static(strTypeNameCLR : String)  
Gives access to .NET types that have no instances and contain only static members.  
*Return Value:* A reference to the static .NET object  

Examples:  
```

var clipboard = CLR.Static( "System.Windows.Clipboard" );
clipboard.SetText( "HelloClipboard" );
```
if ( ShowForm( "FormName" ) == 
    alert( "ok" );
else
    alert( "cancel" );

Form usage and commands

Form usage is as follows:

With Form objects, the Form Component Tree can be accessed naturally via field access:

For example, suppose there is a Form designed as follows:

```
MyForm
    ButtonPanel
        OkButton
        CancelButton
    TextEditor
    AxMediaPlayer1

TrayComponents:
    MyTimer
```

The Form can then be instantiated from script as:

```javascript
var objForm = CreateForm("MyForm");
```

To access one its components the field access can be used:

```javascript
```

or

```javascript
objForm.TextEditor.Text = "Hello World";
```

To access Tray Components use the following method on the Form object:

```javascript
var objTrayComponent = <A form object>.GetTrayComponent(strComponentName : String);
```

In our example to get a reference to the Timer Component to enable it use the following:

```javascript
var objTimer = objForm.GetTrayComponent("MyTimer");
objTimer.Enabled = true;
```

For ActiveX Controls the underlying COM object can be accessed via the OCX property:

```javascript
var ocx = lastform.AxMediaPlayer1.OCX; // get underlying COM object
ocx.enableContextMenu = true;
ocx.URL = "mms://apasf.apa.at/fm4_live_worldwide";
```
1.8 Migrating to Scripting Editor 2010 and Later

The Scripting Editor in Authentic Desktop from version 2010 onwards uses a different underlying technology than earlier versions used. Consequently, scripting projects that were created with versions of Authentic Desktop prior to version 2010 might need to be modified. The following points need to be noted.

- If a previous Scripting Projects (.prj file) is opened with the new Scripting Editor (version 2010 and later), the visual layout of Forms will be migrated as faithfully as possible and scripts will be copied as they are in the .prj file. You will then need to modify the scripts to be in accordance with the new technology used by the Scripting Editor, and which is described in this documentation.

- TheView object: The old Scripting Environment provided an artificial property named TheView that was only accessible from inside event handlers. It was used to access the Form that trigged the event (either directly or from one of its child controls). The new Scripting IDE does not provide this artificial property but instead provides the same functionality, and much more, with orthogonal built-in scripting helper functions combined with the power of the .NET framework.

- Since all event handlers in the new Scripting Environment get a sender object as a first parameter, the source that triggered the event is always available. By calling the .NET function FindForm() on the sender object one can access the Form object easily. Alternatively (if only one Form is involved) the built-in property lastform can be used. Note that the use of lastform is not constrained to event handlers (as was the case with TheView). It can be used everywhere in script code.

Given below is a list of methods and properties of the TheView object, each accompanied by an alternative mechanism offered by the new Scripting Environment.

**Methods**
The following methods were provided by the TheView object and must be migrated as explained:

- **Cancel()**
  In the new scripting environment the same can be achieved with: lastform.Close(); // Use .NET Form.Close()

- **IsFormOpen(Name as String) as Boolean**
  Since for .NET Forms there is a distinction between showing a Form and instantiating a Form, the previous concept does not directly translate. Instead the user can ask if a certain Form is currently shown. For example:

  ```javascript
  var objFormPencilSelector = CreateForm("PencilSelector");
  var objFormColorSelector = CreateForm("ColorSelector");
  ...
  // Anywhere in code ...
  if(objFormColorSelector.Visible)
  {
    ...
  }
  ```
FormFind(Name as String) as Object
The new Scripting Environment allows you to instantiate more Forms of the same kind. In the old Scripting Environment each Form could only exist once (as a Singleton). Thus there is no equivalent of FormFind(). In the new Scripting Environment.

OpenDoc(File as String)
The same can be achieved with: Application.OpenDocument( File as String )

PumpData()
This corresponds to the built-in function doevents() which processes all Windows messages currently in the message queue.

RunClick(), RunInitialize(), RunTerminate()
There is no direct replacement for these methods. Call the corresponding handlers directly instead.

Properties
The following properties were provided by the TheView object and must be migrated as explained:

ToolTipText as String
To use tooltips in the new scripting environment, the .NET infrastructure can be used. This allows fine-grained control of tooltip behaviour (adjusting delays, when to show, etc). For example, to provide tooltips for a Form with two controls, the following code could to be added to the Form's Load event handler:

```javascript
//Occurs whenever the user loads the form.
function MyForm_Load( objSender, e_EventArgs )
{
    // Create the ToolTip and associate with the Form container.
    var toolTip = CLR.Create("System.Windows.Forms.ToolTip");

    // Set up the delays for the ToolTip.
    toolTip.AutoPopDelay = 3000;
    toolTip.InitialDelay = 1000;
    toolTip.ReshowDelay = 500;

    // Force the ToolTip text to be displayed whether or not
    // the form is active.
    toolTip.ShowAlways = true;

    // Set up the ToolTip text for several Controls.
    toolTip.SetToolTip(objSender.ProgressBar1,
        "Shows the progress of the operation");
    toolTip.SetToolTip(objSender.Button1,
        "Click Button to start the processing");
}
```

Color as Long
Since all Form/controls in the new Scripting Environment are .NET controls from the System.Windows.Forms namespace, the possibilities to modify colors, background image, fonts, and all other visual aspects are numerous. For example, every Visual Component has the
properties `BackColor` and `ForeColor` to modify the visual appearance. The following handler could be used to change the color of a button at runtime:

```javascript
function TestForm_Button1_Click( objSender, e_EventArgs )
{
}
```

2 IDE Plugins

Authentic Desktop allows you to create your own IDE plug-ins and integrate them into Authentic Desktop.

Use plug-ins to:
- Configure your version of Authentic Desktop, add commands through menus, icons, buttons etc.
- React to events from Authentic Desktop.
- Run your specific code within Authentic Desktop with access to the complete Authentic Desktop API

Authentic Desktop expects your plug-in to implement the `IXMLSpyPlugIn` interface. VB.NET, C# and C++ examples are included with your installation package and are located in the `Authentic2019\AuthenticExamples\IDEPlugin` folder of your Authentic Desktop installation.

| Windows 7, 8, 10 | C:/Users/<username>/Documents |

See [ATL sample files](#) for an example using C++.
2.1 Registration of IDE PlugIns

Authentic Desktop maintains a specific key in the Registry where it stores all registered IDE plug-ins:

HKEY_CURRENT_USER\Software\Altova/XML Spy\PlugIns

All values of this key are treated as references to registered plug-ins and must conform to the following format:

Value name: ProgID of the plug-in
Value type: must be REG_SZ
Value data: CLSID of the component

Each time the application starts the values of the "PlugIns" key is scanned, and the registered plug-ins are loaded.

Register plug-in manually

To register a plug-in manually, use the "Customize" dialog box of the Authentic Desktop "Tools" menu. Use the "Add Plug-In..." button to specify the DLL that implements your plug-in. Authentic Desktop registers the DLL as a COM server and adds the corresponding entry in its "PlugIns" key.

If you experience problems with manual registration you can check if the CLSID of your plug-in is correctly registered in the "PlugIns" key. If this is not the case, the name of your plug-in DLL was probably not sufficiently unique. Use a different name or perform direct registration.

Register plug-in directly

A plug-in can be directly registered as an IDE plug-in by first registering the DLL and then adding the appropriate value to the "PlugIns" key of Authentic Desktop during plug-in setup for example. The new plug-in will be activated the next time Authentic Desktop is launched.

Creating plug-ins

Source code for sample plug-ins has been provided in the application's (My) Documents folder: Examples\IDEPlugin folder. To build a plug-in from such source code, do the following:

1. Open the solution you want to build as a plug-in in Visual Studio.
2. Build the plug-in with the command in the Build menu.
3. The plug-in's DLL file that will be created in the Bin or Debug folder. This DLL file is the file that must be added as a plug-in (see above).
2.2 ActiveX Controls

ActiveX controls are supported. Any IDE PlugIn which is also an ActiveX control will be displayed in a Dialog Control Bar. A sample Plugin that is also an ActiveX control is included in the XMLSpyPlugInActiveX folder in the Examples folder of your application folder.
2.3 Configuration XML

The IDE plug-in allows you to change the user interface (UI) of Authentic Desktop. This is done by describing each separate modification using an XML data stream. The XML configuration is passed to Authentic Desktop using the `GetUIModifications` method of the `IXMLSpyPlugIn` interface.

The XML file containing the UI modifications for the IDE Plugin, must have the following structure:

```xml
<ConfigurationData>
  <ImageFile>path To image file</ImageFile>
  <Modifications>
    <Modification>
      ...
    </Modification>
    ...
  </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for the new menu items which are added to the UI of Authentic Desktop by the plug-in. The path to the file containing the images is set using the `ImageFile` element. Each image must be 16 x 16 pixels using max. 256 colors. The image references must be arranged from left to right in a single (`<ImageFile>...`) line. The rightmost image index value, is zero.

The `Modifications` element can have any number of `Modification` child elements. Each `Modification` element defines a specific change to the standard UI of Authentic Desktop. Starting with version 4.3, it is also possible to remove UI elements from Authentic Desktop.

Structure of Modification elements
All Modification elements consist of the following two child elements:

```xml
<Modification>
  <Action>Type of action</Action>
  <UIElement Type="type of UI element">
    ...
  </UIElement>
</Modification>
```

Valid values for the `Action` element are:

- Add - to add the following UI element to Authentic Desktop
- Hide - to hide the following UI element in Authentic Desktop
- Remove - to remove the UI element from the "Commands" list box, in the customize dialog

You can combine values of the `Action` element e.g. "Hide Remove"

The `UIElement` element describes any new, or existing UI element for Authentic Desktop. Possible elements are currently: new toolbars, buttons, menus or menu items. The `type` attribute, defines which UI element is described by the XML element.

Common `UIElement` children
The ID and Name elements are valid for all different types of XML `UIElement` fragments. It is however possible, to ignore one of the values for a specific type of `UIElement` e.g. Name is ignored
for a separator.

```xml
<ID></ID>
<Name></Name>
```

If `UIElement` describes an existing element of the UI, the value of the ID element is predefined by Authentic Desktop. Normally these ID values are not known to the public. If the XML fragment describes a new part of the UI, then the ID is arbitrary and the value should be less than 1000. The `Name` element sets the textual value. Existing UI elements can be identified just by name, for e.g. menus and menu items with associated sub menus. For new UI elements, the `Name` element sets the caption e.g. the title of a toolbar, or text for a menu item.

**Toolbars and Menus**

To define a toolbar it's necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name, or by the ID if it is known. To create a new toolbar both values must be set. The `type` attribute must be equal to "ToolBar".

```xml
<UIElement Type="ToolBar">
  <ID>1</ID>
  <Name>TestPlugIn</Name>
</UIElement>
```

To specify an Authentic Desktop menu you need two parameters:

- The ID of the menu bar which contains the menu. If no XML documents are open in the main window, the menu bar ID is 128. If one or more XML documents are open, the menu bar ID is 129.
- The menu name. Menus do not have an associated ID value. The following example defines the "Edit" menu of the menu bar which is active, when at least one XML document is open:

```xml
<UIElement Type="Menu">
  <ID>129</ID>
  <Name>Edit</Name>
</UIElement>
```

An additional element is used if you want to create a new menu. The `Place` element defines the position of the new menu in the menu bar:

```xml
<UIElement Type="Menu">
  <ID>129</ID>
  <Name>PlugIn Menu</Name>
  <Place>12</Place>
</UIElement>
```

A value of -1 for the `Place` element sets the new button or menu item at the end of the menu or toolbar.

**Commands**

If you add a new command, through a toolbar button or a menu item, the `UIElement` fragment can contain any of these sub elements:

```xml
<MacroName></MacroName>
<Info></Info>
```
If **MacroName** is specified, Authentic Desktop searches for a macro with the same name in the scripting environment and executes it each time this command is processed. The **Info** element contains a short description string which is displayed in the status bar, when the mouse pointer is over the associated command (button or menu item). **ImageID** defines the index of the icon the external image file. Please note that all icons are stored in one image file.

To define a toolbar button create an **UIElement** with this structure:

```xml
<UIElement Type="ToolBarItem">
    <!--don't reuse local IDs even the commands do the same-->
    <ID>5</ID>
    <Name>Open file from repository...</Name>
    <!--Set Place To -1 If this is the first button To be inserted-->
    <Place>-1</Place>
    <ImageID>0</ImageID>
    <ToolBarID>1</ToolBarID>
    <!--instead of the toolbar ID the toolbar name could be used-->
    <ToolBarName>TestPlugIn</ToolBarName>
</UIElement>
```

Additional elements to declare a toolbar button are **Place**, **ToolBarID** and **ToolBarName**. **ToolBarID** and **ToolBarName** are used to identify the toolbar which contains the new or existing button. The textual value of **ToolBarName** is case sensitive. The (**UIElement**) **type** attribute must equal "ToolBarItem".

To define a menu item, the elements **MenuID**, **Place** and **Parent** are available in addition to the standard elements used to declare a command. **MenuID** can be either 128 or 129. Please see "Toolbars and Menus" for more information on these values.

The **Parent** element is used to identify the **menu** where the new menu entry should be inserted. As sub menu items have no unique Windows ID, we need some other way to identify the parent of the menu item.

The value of the **Parent** element is a path to the menu item. The text value of the Parent element, must equal the **parent menu name** of the submenu, where the submenu name is separated by a colon. If the menu has no parent, because its not a submenu, add a colon to the beginning of the name. The **type** attribute must be set to "MenuItem". Example for an **UIElement** defining a menu item:

```xml
<UIElement Type="MenuItem">
    <!--the following element is a Local command ID-->
    <ID>3</ID>
    <Name>Open file from repository...</Name>
    <!--Set Place To -1 If this is the first button To be inserted-->
    <Place>-1</Place>
    <MenuID>129</MenuID>
    <Parent>:PlugIn Menu</Parent>
    <ImageID>0</ImageID>
</UIElement>
```

Authentic Desktop makes it possible to add toolbar separators and menus if the value of the **ID** element is set to 0.
2.4 ATL sample files

The following pages show how to create a simple Authentic Desktop IDE plug-in DLL using ATL. To build the DLL it is necessary to know about ATL, the wizards that generate new ATL objects, as well as MS Visual Studio.

To access the API the implementation imports the Type Library of Authentic Desktop. The code reads various properties and calls methods using the smart pointers provided by the #import statement.

In addition, the sample code uses the MFC class CString and the ATL conversion macros such as W2T.

At a glance the steps to create an ATL DLL are as follows:

1. Open Visual Studio and select "New..." from the "File" menu.
2. Select the "Projects" tab.
3. Select "ATL COM AppWizard" and type in a project name.
4. Select "Support for MFC" if you want to use MFC classes, or if you want to create a project for the sample code.

Having created the project files you can add an ATL object to implement the IXMLSpyPlugin interface:

1. Select "New ATL Object..." from the "Insert" menu.
2. Select "Simple Object" from the wizard and click "Next".
3. Type in a name for the object.
4. On the "Attributes" tab, select "Custom" for the type of interface, and disable Aggregation.

These steps produce the skeleton code for the implementation of the IDE plug-in interface. Please see the following pages on how to modify the code and achieve some basic functionality.
2.4.1 Interface description (IDL)

The IDL of the newly created ATL object contains a declaration for one COM interface.

- This interface declaration must be replaced by the declaration of IXMLSpyPlugIn as shown below.
- The IDL must also contain the definition of the SPYUpdateAction enumeration.
- Replace the generated default interface name, (created by the wizard) with "IXMLSpyPlugIn" in the coclass declaration. The IDL should then look something like the example code below:

Having created the ATL object, you then need to implement the IDE plug-in interface of Authentic Desktop.

```c
import "oaidl.idl";
import "ocidl.idl";

// ----- please insert the following block into your IDL file ----- 

typedef enum {
    spyEnable = 1,
    spyDisable = 2,
    spyCheck = 4,
    spyUncheck = 8
} SPYUpdateAction;

// ----- end insert block -----

// ----- E.g. Interface entry automatically generated by the ATL wizard ----- 

// [ 
//   object,
//   uuid(AB7CD86A-8145-429A-A1F3-270692E08AFC),
//   helpstring("IXMLSpyPlugIn Interface")
//   pointer_default(unique)
// ] 
// interface IXMLSpyPlugIn : IUnknown
// {
// }

// ----- end automatically generated Interface Entry

// ----- replace the Interface Entry (shown above) generated for you by the ATL wizard, with the following block ----- 

[ 
    odl,
    uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85),
    helpstring("IXMLSpyPlugIn Interface")
]
interface IXMLSpyPlugIn : IUnknown
{
    HRESULT _stdcall OnCommand([in] long nID, [in] IDispatch* pXMLSpy);
}
```
HRESULT _stdcall OnUpdateCommand([in] long nID, [in] IDispatch* pXMLSpy, [out, retval] SPYUpdateAction* pAction);


HRESULT _stdcall GetUIModifications([out, retval] BSTR* pModificationsXML);

HRESULT _stdcall GetDescription([out, retval] BSTR* pDescription);
};

// ----- end replace block ----- 

// ----- The code below is automatically generated by the ATL wizard and will look slightly different in your case ----- 

[uuid(24FE0D1B-3FC0-494E-B36E-1D4CE412B014), version(1.0), helpstring("XMLSpyIDEPlugInDLL 1.0 Type Library") ]
library XMLSpyIDEPLUGINDLLLib
{
importlib("stdole32.tlb");
importlib("stdole2.tlb");

[uuid(3800E791-7F6B-4ACD-9E32-2AC184444501), helpstring("XMLSpyIDEPlugIn Class") ]
coclass XMLSpyIDEPlugIn
{
    [default] interface IXMLSpyPlugIn; // ----- define IXMLSpyPlugIn as the default interface ----- 
};
}
378

Programmers' Reference

2.4.2

Class definition

IDE Plugins

In the class definition of the ATL object, several changes must be made. The class has to derive
from IXMLSpyPlugIn, the "Interface Map" needs an entry for IXMLSpyPlugIn, and the methods of
the IDE plug-in interface must be declared:
#ifndef __XMLSPYIDEPLUGIN_H_
#define __XMLSPYIDEPLUGIN_H_
#include "resource.h"

// main symbols

/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn
class ATL_NO_VTABLE CXMLSpyIDEPlugIn :
public CComObjectRootEx<CComSingleThreadModel>,
public CComCoClass<CXMLSpyIDEPlugIn, &CLSID_XMLSpyIDEPlugIn>,
public IXMLSpyPlugIn
{
public:
CXMLSpyIDEPlugIn()
{
}
DECLARE_REGISTRY_RESOURCEID(IDR_XMLSPYIDEPLUGIN)
DECLARE_NOT_AGGREGATABLE(CXMLSpyIDEPlugIn)
DECLARE_PROTECT_FINAL_CONSTRUCT()
BEGIN_COM_MAP(CXMLSpyIDEPlugIn)
COM_INTERFACE_ENTRY(IXMLSpyPlugIn)
END_COM_MAP()
// IXMLSpyIDEPlugIn
public:
virtual HRESULT _stdcall OnCommand(long nID, IDispatch* pXMLSpy);
virtual HRESULT _stdcall OnUpdateCommand(long nID, IDispatch* pXMLSpy,
SPYUpdateAction* pAction);
virtual HRESULT _stdcall OnEvent(long nEventID, SAFEARRAY **arrayParameters,
IDispatch* pXMLSpy, VARIANT* pReturnValue);
virtual HRESULT _stdcall GetUIModifications(BSTR* pModificationsXML);
virtual HRESULT _stdcall GetDescription(BSTR* pDescription);
};
#endif //__XMLSPYIDEPLUGIN_H_

Altova Authentic 2019 Desktop Enterprise Edition

© 2019 Altova Gmb H


### 2.4.3 Implementation

The code below shows a simple implementation of an Authentic Desktop IDE plug-in. It adds a menu item and a separator (available with Authentic Desktop) to the Tools menu. Inside the OnUpdateCommand() method, the new command is only enabled when the active document is displayed using the Grid View. The command searches for the XML element which has the current focus, and opens any URL starting with "http://", from the textual value of the element.

```cpp
(itemView)
import "XMLSpy.tlb"
using namespace XMLSpyLib;

HRESULT CXMLSpyIDEPlugIn::OnCommand(long nID, IDispatch* pXMLSpy)
{
    USES_CONVERSION;

    if(nID == 1) {
        IApplicationPtr ipSpyApp;

        if(pXMLSpy) {
            if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication), (void**) &ipSpyApp))) {
                IDocumentPtr ipDocPtr = ipSpyApp->ActiveDocument;

                // we assume that grid view is active
                if(ipDocPtr) {
                    IGridViewPtr ipGridPtr = ipDocPtr->GridView;

                    if(ipGridPtr) {
                        IXMLDataPtr ipXMLData = ipGridPtr->CurrentFocus;

                        CString strValue = W2T(ipXMLData->TextValue);

                        if(!strValue.IsEmpty() && (strValue.Left(7) == _T("http://")))
                            ::ShellExecute(NULL, _T("open"), W2T(ipXMLData->TextValue), NULL, NULL, SW_SHOWNORMAL);
                    }
                }
            }
        }
    }

    return S_OK;
}

HRESULT CXMLSpyIDEPlugIn::OnUpdateCommand(long nID, IDispatch* pXMLSpy, SPYUpdateAction* pAction)
{
    *pAction = spyDisable;
}
if(nID == 1) {
    IApplicationPtr ipSpyApp;
}

if(pXMLSpy) {
    if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void **) &ipSpyApp))) {
        IDocumentPtr ipDocPtr = ipSpyApp->ActiveDocument;

        // only enable if grid view is active
        if((ipDocPtr != NULL) && (ipDocPtr->CurrentViewMode == spyViewGrid))
            *pAction = spyEnable;
        }
    }
}

return S_OK;

HRESULT CXMLSpyIDEPlugIn::OnEvent(long nEventID, SAFEARRAY **arrayParameters, IDispatch* pXMLSpy, VARIANT* pReturnValue)
{
    return S_OK;
}

HRESULT CXMLSpyIDEPlugIn::GetUIModifications(BSTR* pModificationsXML)
{
    CComBSTR bstrMods = _T("<ConfigurationData>
<Modifications>");
    // add "Open URL..." to Tools menu
    bstrMods.Append(_T("<Modification>
<Action>Add</Action> 
<UIElement type="MenuItem"> 
<ID>1</ID> 
{Name>Open URL...</Name> 
<Place>0</Place> 
<MenuID>129</MenuID> 
<Parent>:Tools</Parent> 
</UIElement> 
</Modification>
"));
    // add Separator to Tools menu
    bstrMods.Append(_T("<Modification> 
>Action>Add</Action> 
<UIElement type="MenuItem"> 
<ID>0</ID> 
<Place>1</Place> 
<MenuID>129</MenuID> 
<Parent>:Tools</Parent> 
</UIElement> 
</Modification>"));
// finish modification description
bstrMods.Append(_T(" // finish modification description 
    </Modifications> 
    </ConfigurationData>");

return bstrMods.CopyTo(pModificationsXML);
}

HRESULT CXMLSpyIDEPlugIn::GetDescription(BSTR* pDescription)
{
    CComBSTR bstrDescr = _T("ATL C++ XMLSpy IDE PlugIn;This PlugIn demonstrates the implementation of a simple ATL DLL as a IDE PlugIn for XMLSpy.");
    return bstrDescr.CopyTo(pDescription);
}
2.5 IXMLSpyPlugIn

See also

Methods
OnCommand
OnUpdateCommand
OnEvent
GetUIModifications
GetDescription

Description
If a DLL is added to Authentic Desktop as an IDE plug-in, it is necessary that it registers a COM component that answers to an IXMLSpyPlugIn interface with the reserved uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85), for it to be recognized as a plug-in.
2.5.1 OnCommand

See also

Declaration: OnCommand(nID as long, pXMLSpy as IDispatch)

Description
The OnCommand() method of the interface implementation, is called each time a command added by the IDE plug-in (menu item or toolbar button) is processed. nID stores the command ID defined by the ID element of the respective UIElement.

pXMLSpy holds a reference to the dispatch interface of the Application object of Authentic Desktop.

Example

Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
    If (Not (pXMLSpy Is Nothing)) Then
        Dim objDlg
        Dim objDoc As XMLSpyLib.Document
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If nID = 3 Or nID = 5 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "XML Files (*.xml)|*.xml|All Files (*.*)|*.*||"
            objDlg.FilterIndex = 1
            objDlg.ShowOpen
            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.Documents.OpenFile(objDlg.FileName, False)
                Set objDoc = Nothing
            End If
        End If

        If nID = 4 Or nID = 6 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "All Files (*.*)|*.*||"
            objDlg.Flags = cdlOFNPathMustExist
            objDlg.ShowSave
            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.ActiveDocument
                If Not (objDoc Is Nothing) Then
                    objDoc.SetPathName objDlg.FileName
                    objDoc.Save
                    Set objDoc = Nothing
                End If
            End If
        End If
        Set objSpy = Nothing
    End If
End Sub
2.5.2 OnUpdateCommand

See also

Declaration: OnUpdateCommand(nID as long, pXMLSpy as IDispatch) as SPYUpdateAction

Description
The OnUpdateCommand() method is called each time the visible state of a button or menu item needs to be set. nID stores the command ID defined by the ID element of the respective UIElement.

pXMLSpy holds a reference to the dispatch interface of the Application object.

Possible return values to set the update state are:

spyEnable = 1
spyDisable = 2
spyCheck = 4
spyUncheck = 8

Example
Public Function IXMLSpyPlugIn_OnUpdateCommand(ByVal nID As Long, ByVal pXMLSpy As Object) As SPYUpdateAction
    IXMLSpyPlugIn_OnUpdateCommand = spyDisable
    If (Not (pXMLSpy Is Nothing)) Then
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy
        If nID = 3 Or nID = 5 Then
            IXMLSpyPlugIn_OnUpdateCommand = spyEnable
        End If
        If nID = 4 Or nID = 6 Then
            If objSpy.Documents.Count > 0 Then
                IXMLSpyPlugIn_OnUpdateCommand = spyEnable
            Else
                IXMLSpyPlugIn_OnUpdateCommand = spyDisable
            End If
        End If
    End If
End Function
2.5.3 **OnEvent**

**See also**

*Declaration:* `OnEvent(nEventID as long, arrayParameters as SAFEARRAY(VARIANT), pXMLSpy as IDispatch) as VARIANT`

**Description**

`OnEvent()` is called each time an event is raised from Authentic Desktop.

Possible values for `nEventID` are:

- `On_BeforeStartEditing = 1`
- `On_EditingFinished = 2`
- `On_FocusChanged = 3`
- `On_Beforedrag = 4`
- `On_BeforeDrop = 5`
- `On_OpenProject = 6`
- `On_OpenDocument = 7`
- `On_CloseDocument = 8`
- `On_SaveDocument = 9`

Events available since Authentic Desktop 4r4:

- `On_DocEditDragOver = 10`
- `On_DocEditDrop = 11`
- `On_DocEditKeyDown = 12`
- `On_DocEditKeyUp = 13`
- `On_DocEditKeyPressed = 14`
- `On_DocEditMouseMove = 15`
- `On_DocEditButtonUp = 16`
- `On_DocEditButtonDown = 17`
- `On_DocEditContextMenu = 18`
- `On_DocEditPaste = 19`
- `On_DocEditCut = 20`
- `On_DocEditCopy = 21`
- `On_DocEditClear = 22`
- `On_DocEditSelectionChanged = 23`

Events available since Authentic Desktop 2004:

- `On_DocEditDragOver = 10`
Events available since Authentic Desktop 2004r4 (type library version 1.4):

- On_BeforeOpenProject = 25
- On_BeforeOpenDocument = 26
- On_BeforeSaveDocument = 27
- On_BeforeCloseDocument = 28
- On_ViewActivation = 29
- On_DocEditKeyboardEvent = 30
- On_DocEditMouseEvent = 31

Events available since Authentic Desktop 2006 SP1 (type library version 1.5):

- On_BeforeValidate = 32

Events available since Authentic Desktop 2007 (type library version 1.6):

- On_BeforeShowSuggestions = 33
- On_ProjectOpened = 34
- On_Char = 35

Events available since Authentic Desktop 2009 (type library version 2.2):

- On_Initialize = 36
- On_Running = 37
- On_Shutdown = 38

Events available since Authentic Desktop 2012 (type library version 2.8):

- On_AuthenticBeforeSave = 39
- On_AuthenticContextMenuActivated = 40
- On_AuthenticLoad = 41
- On_AuthenticToolbarButtonClicked = 42
- On_AuthenticToolbarButtonExecuted = 43
- On_AuthenticUserAddedXMLNode = 44

The names of the events are the same as they appear in the Scripting Environment of Authentic Desktop. For IDE plug-ins the names used are immaterial. The events are identified using the ID value.

arrayParameters is an array which is filled with the parameters of the currently raised event. Order, type and meaning of the single parameters are available through the scripting environment of Authentic Desktop. The events module of a scripting project, contains predefined functions for all events prior to version 4.4. The parameters passed to the predefined functions are identical to the array elements of the arrayParameters parameter.

Events raised from the Authentic View of Authentic Desktop do not pass any parameters directly. An "event" object is used instead. The event object can be accessed through the Document object of the active document.

pXMLSpy holds a reference to the dispatch interface of the Application object of Authentic Desktop.
If the return value of `OnEvent()` is set, then neither the IDE plug-in, nor an event handler inside of the scripting environment will get this event afterwards. Please note that all IDE plug-ins get/process the event before the Scripting Environment does.
2.5.4 GetUIModifications

See also

Declaration: GetUIModifications() as String

Description

The GetUIModifications() method is called during initialization of the plug-in, to get the configuration XML data that defines the changes to the UI of Authentic Desktop. The method is called when the plug-in is loaded for the first time, and at every start of Authentic Desktop.

See also Configuration XML for a detailed description how to change the UI.

Example

Public Function IXMLSpyPlugIn_GetUIModifications() As String
' GetUIModifications() gets the XML file with the specified modifications of
' the UI from the config.xml file in the plug-in folder
Dim strPath As String
strPath = App.Path

If Len(strPath) > 0 Then
    Dim fso As New FileSystemObject
    Dim file As file
    Set file = fso.GetFile(strPath & "\config.xml")

    If (Not (file Is Nothing)) Then
        Dim stream As TextStream
        Set stream = file.OpenAsTextStream(ForReading)

        ' this replaces the token '**path**' from the XML file with
        ' the actual installation path of the plug-in to get the image file
        Dim strMods As String
        strMods = stream.ReadAll
        strMods = Replace(strMods, "**path**", strPath)

        IXMLSpyPlugIn_GetUIModifications = strMods
    Else
        IXMLSpyPlugIn_GetUIModifications = ""
    End If
End If
End Function
2.5.5 GetDescription

See also

Declaration: GetDescription() as String

Description
GetDescription() is used to define the description string for the plug-in entries visible in the Customize dialog box.

Example
Public Function IXMLSpyPlugIn_GetDescription() As String
    IXMLSpyPlugIn_GetDescription = "Sample Plug-in for XMLSpy;This Plug-in demonstrates the implementation of a simple VisualBasic DLL as a Plug-in for XMLSpy."
End Function
3 Application API

The COM-based API of Authentic Desktop (also called the Application API from now on) enables other applications to use the functionality of Authentic Desktop. As a result, it is possible to automate a wide range of tasks, from validating an XML file to modifying complex XML content (with the XMLData interface).

Authentic Desktop and its Application API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the Application API from common development environments, such as those using C, C++, VisualBasic, and Delphi, and with scripting languages like JScript and VBScript.

Execution environments for the Application API

The Application API can be accessed from the following execution environments:

- External programs (described below and in the Overview part of this section)
- From within the built-in Scripting Editor of Authentic Desktop. For a description of the scripting environment, see the section, Scripting Editor.
- Authentic Desktop allows you to create and integrate your own plug-ins into the application using a special interface for plug-ins. A description of how to create plug-ins is given in the section IDE Plug-ins.
- Via an ActiveX Control, which is available if the integration package is installed. For more information, see the section ActiveX Integration.

External programs

In the Overview part of this section, we describe how the functionality of Authentic Desktop can be accessed and automated from external programs.

Using the Application API from outside Authentic Desktop requires an instance of Authentic Desktop to be started first. How this is done depends on the programming language used. See the section, Programming Languages, for information about individual languages.

Essentially, Authentic Desktop will be started via its COM registration. Then the Application object associated with the Authentic Desktop instance is returned. Depending on the COM settings, an object associated with an already running Authentic Desktop can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript and VBScript script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C# is a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#.
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for
Applications, Perl, and Python.

**Programming points**
The following limitations must be considered in your client code:

- Be aware that if your client code crashes, instances of Authentic Desktop may still remain in the system.
- Don't hold references to objects in memory longer than you need them, especially those from the `XMLData` interface. If the user interacts between two calls of your client, then there is no guarantee that these references are still valid.
- Don't forget to disable dialogs if the user interface is not visible.
- See [Error handling in JScript](#) (and in [C#](#) and [Java](#)) for details of how to avoid annoying error messages.
- Free references explicitly if you are using C or C++.

**This documentation**
This documentation section about the Application API is broadly divided into two parts.

- The first part consists of an Overview, which describes the object model for the API and explains how the API is accessed via various programming languages.
- The second part is a reference section (Interfaces and Enumerations) that contains descriptions of the interface objects of the Application API.
3.1 Overview

This overview of the Application API is organized as follows:

- **The Object Model** describes the relationships between the objects of the Application API.
- **Programming Languages** explains how the most commonly used programming languages (JScript, VBScript, C#, and Java) can be used to access the functionality of the Application API. Code listings from the example files supplied with your application package are used to describe basic mechanisms.
3.1.1 Object Model

The starting point for every application which uses the Application API is the Application object. This object contains general methods like import/export support and references to the open documents and any open project.

The Application object is created differently in various programming languages. In scripting languages such as JScript or VBScript, this involves calling a function which initializes the application's COM object. For examples, see the Programming Languages section.

XMLSpy.Application or AuthenticDesktop.Application

Authentic Desktop installs a TypeLibrary containing the XMLSpyLib. If this TypeLibrary has been added to the development environment (VB development environment, for example) then an object of the Application type can be created with:

    Set objSpy = New XMLSpyLib.Application

If only Authentic Desktop is installed (and not XMLSpy), then

    Set objSpy = GetObject("", "XMLSpy.Application")

does not work, because there won't be any object registered in the Registry with a ProgID of XMLSpy.Application. In this case, the registered object is AuthenticDesktop.Application.

The code listings in this documentation assume that both Authentic Desktop and XMLSpy have been installed. If, however, only Authentic Desktop has been installed, then please modify code fragments to take account of this difference.

The application object consists of the following parts:

2. Reference to current project and methods for creating and opening projects.
3. Methods to support the export to and import from databases, text files, and Word documents.
4. URL management.
5. Methods for macro menu items.

Once you have created an Application object you can start using the functionality of Authentic Desktop. In most cases, you either open a project and access the documents from there or you directly open a document via the Documents interface.
3.1.2 Programming Languages

Programming languages differ in the way they support COM access. A few examples for the most frequently used languages (links below) will help you get started. The code listings in this section show how basic functionality can be accessed. This basic functionality is included in the files in the API Examples folder and can be tested straight away. The path to the API Examples folder is given below:

<table>
<thead>
<tr>
<th>Windows 7, Windows 8, Windows 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\Users&lt;username&gt;\Documents\Altova\Authentic2019\Authentic</td>
</tr>
</tbody>
</table>

JScript

The JScript listings demonstrate the following basic functionality:

- **Start application or attach to a running instance**
- **Simple document access**
- **Iteration**
- **Error handling**
- **Events**

VBScript

VBScript is different than JScript only syntactically; otherwise it works in the same way. The listings below describe is an example of how VBScript can be used. For more information, refer to the [JScript examples](#).

- **Events**: Shows how events are handled using VBScript.

C#

C# can be used to access the Application API functionality. The code listings show how to access the API for certain basic functionality.

- **Start Authentic Desktop**: Starts Authentic Desktop, which is registered as an automation server, or activates the application if it is already running.
- **Open OrgChart.pxf**: Locates one of the example documents installed with Authentic Desktop and opens it. If this document is already open it becomes the active document.
- **OnDocumentOpened Event On/Off**: Shows how to listen to Authentic Desktop events. When turned on, a message box will pop up after a document has been opened.
- **Open ExpReport.xml**: Opens another example document.
- **Toggle View Mode**: Changes the view of all open documents between Browser View and Authentic View. The code shows how to iterate through open documents.
- **Validate**: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- **Shutdown Authentic Desktop**: Stops Authentic Desktop.

Java

The Authentic Desktop API can be accessed from Java code. The Java sub-section of this section explains how some basic Authentic Desktop functionality can be accessed from Java code. It is
organized into the following sub-sections:

- Mapping Rules for the Java Wrapper
- Example Java Project
- Application Startup and Shutdown
- Simple Document Access
- Iterations
- Use of Out-Parameters
- Event Handlers

**JScript**

This section contains listings of JScript code that demonstrate the following basic functionality:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling
- Events

**Example files**

The code listings in this section are available in example files that you can test as is or modify to suit your needs. The JScript example files are located in the JScript folder of the API Examples folder:

| Windows 7, Windows 8, Windows 10 | C: \Users\<username>\Documents\Altova\Authentic2019\Authentic |

The example files can be run in one of two ways:

- From the command line: Open a command prompt window, change the directory to the path above, and type the name of one of the example scripts (for example, Start.js).
- From Windows Explorer: In Windows Explorer, browse for the JScript file and double-click it.

The script is executed by Windows Script Host that is packaged with Windows operating system. For more information about Windows Script Host, refer to MSDN documentation (https://msdn.microsoft.com).

**Start Application**

The JScript below starts the application and shuts it down. If the COM object of the 32-bit Authentic Desktop cannot be found, the code attempts to get the COM object of the 64-bit application; otherwise, an error is thrown. If an instance of the application is already running, the running instance will be returned.

**Note:** For 32-bit Authentic Desktop, the registered name, or programmatic identifier (ProgId) of the COM object is AuthenticDesktop.Application. For 64-bit Authentic Desktop, the name is AuthenticDesktop_x64.Application.
// Initialize application's COM object. This will start a new instance of the application and
// return its main COM object. Depending on COM settings, the main COM object of an already
// running application might be returned.

try {
} catch (err) {}

if( typeof( objAuthentic ) == "undefined" ) {
    try {
        objAuthentic = WScript.GetObject("", "AuthenticDesktop_x64.Application")
    } catch (err)
    {
        WScript.Echo( "Can't access or create AuthenticDesktop.Application" );
        WScript.Quit();
    }
}

// if newly started, the application will start without its UI visible. Set it to visible.
objAuthentic.Visible = true;
WScript.Echo(objAuthentic.Edition + " has successfully started. ");

objAuthentic.Visible = false; // will shutdown application if it has no more COM connections
//objAuthentic.Visible = true; // will keep application running with UI visible

The JScript code listed above is available in the sample file Start.js (see Example Files).

Simple Document Access

After you have started the application as shown in Start Application, you will most likely want to
programmatically open a document in order to work with it. The JScript code listing below illustrates how to open two documents from the Authentic Desktop Examples folder and set one of them as the active document.

// Locate examples via USERPROFILE shell variable. The path needs to be adapted to major release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objAuthentic.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\My Documents\Altova\Authentic" + majorVersionYear + "\AuthenticExamples\";

// Tell Authentic to open two documents. No dialogs
objDoc1 = objAuthentic.Documents.OpenFile(strExampleFolder + "OrgChart.pxf", false);
objAuthentic.Documents.OpenFile(strExampleFolder + "ExpReport.xml", false);

// The document currently active can be easily located.
objDoc2 = objAuthentic.ActiveDocument;
Programmers' Reference

Application API

397

// Let us make sure that the document is shown in Authentic view.
objDoc2.SwitchViewMode(5);
// SPYViewModes.spyViewAuthentic = 5
// Now switch back to the document opened first
objDoc1.SetActiveDocument();

The JScript code listed above is available in the sample file DocumentAccess.js (see Example
Files).

Iteration
The JScript listing below shows how to iterate through the open documents. It is assumed that
you have already started the application and opened some documents as shown in the previous
sections.
// go through all open documents using a JScript Enumerator
bRequiresSaving = false;
for (var iterDocs = new Enumerator(objAuthentic.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
if (iterDocs.item().IsModified)
bRequiresSaving = true;
var strErrorText = new Array(1);
var nErrorNumber = new Array(1);
var errorData = new Array(1);
if (!iterDocs.item().IsValid(strErrorText, nErrorNumber, errorData))
{
var text = strErrorText;
// access that XMLData object only if filled in
if (errorData[0] != null)
text += "(" + errorData[0].Name + "/" + errorData[0].TextValue + ")";
WScript.Echo("Document \"" + iterDocs.item().Name +"\" validation
error[" + nErrorNumber + "]: " + text);
}
else
{
// The COM call succeeded and the document is valid.
WScript.Echo("Document \"" + iterDocs.item().Name + "\" is valid.");
}
}
// go through all open documents using index-based access to the document
collection
for (i = objAuthentic.Documents.Count; i > 0; i--)
objAuthentic.Documents.Item(i).Close(false);

The JScript code listed above is available in the sample file DocumentAccess.js (see Example
Files).

Error Handling
The Application API returns errors in two different ways:

© 2019 Altova Gmb H

Altova Authentic 2019 Desktop Enterprise Edition


• The HRESULT returned by every API method
• The IErrorInfo interface of the Application API

Every API method returns an HRESULT. This return value gives the caller information about errors during execution of the method. If the call was successful, the return value is S_OK. The HRESULT option is commonly used in C/C++ programs.

However, programming languages such as VisualBasic and scripting languages (and other high-level development environments) don't give the programmer access to the HRESULT return of a COM call. Such languages use the IErrorInfo interface, which is also supported by the Application API. If an error occurs, the Application API creates a new object that implements the IErrorInfo interface. The information provided by the IErrorInfo interface is imported by the development environment into its own error-handling mechanism.

For example, the JScript code listing below causes an error to be thrown by incorrectly declaring an array. Additional information about the error object is provided by its properties number and description.

```javascript
try {
    var arr = new Array(-1);
} catch (err) {
    WScript.Echo("Error : (" + (err.number & 0xffff) + ")" + err.description);
}
```

**Events**

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using JScript.

The method WScript.ConnectObject is used to receive events.

```javascript
// The event-handler function
function DocEvent_OnBeforeCloseDocument(objDocument) {
    WScript.Echo("Received event - before closing document");
}

// Create or connect to XMLSpy (or Authentic Desktop)
try {
    // Create the environment and XMLSpy (or Authentic Desktop)
    objWshShell = WScript.CreateObject("WScript.Shell");
    objFSO = WScript.CreateObject("Scripting.FileSystemObject");
    objSpy = WScript.GetObject("", "XMLSpy.Application");

    // If only Authentic Desktop is installed (and XMLSpy is not installed) use:
    // objSpy = WScript.GetObject("", "AuthenticDesktop.Application")
}
catch(err) {
    WScript.Echo("Can't create WScript.Shell object or XMLSpy");
}
```
// Create document object and connect to its events
objSpy.Visible = true;
majorVersionYear = objSpy.MajorVersion + 1998
docPath = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\Documents \Altova\XMLSpy" + majorVersionYear + "\Examples\ExpReport.xml";
objDoc = objSpy.Documents.OpenFile (docPath, false);
WScript.ConnectObject(objDoc, "DocEvent_");

// Keep running while waiting for the event
// In the meanwhile close this document in XMLSpy (or Authentic Desktop) manually
WScript.Echo("Sleeping for 10 seconds ...");
WScript.Sleep(10000);
objDoc = null;
WScript.Echo("Stopped listening for event");
objSpy.Quit();

VBScript

VBScript is syntactically different than JScript but works in the same way. This section contains a listing showing how events are used with VBScript and an example.

For information about other functionality, refer to the JScript examples listed below:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling

Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using VBScript.

The method WScript.ConnectObject is used to receive events.

To run this code, paste it into a file with .vbs extension, and either double-click in Windows Explorer, or run it from a command prompt.

' the event handler function
    Call WScript.Echo("received event - before closing document")
End Function

' create or connect to XmlSpy
Set objWshShell = WScript.CreateObject("WScript.Shell")
Set objFSO = WScript.CreateObject("Scripting.FileSystemObject")
Set objSpy = WScript.GetObject("", "XMLSpy.Application")
' If only Authentic is installed (and XMLSpy is not installed) use:
' Set objSpy = WScript.GetObject("", "AuthenticDesktop.Application")
' If only XMLSpy 64-bit is installed, use:
' Set objSpy = WScript.GetObject("", "XMLSpy_x64.Application")

' create document object and connect to its events
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed examples.
personalFolder = objWshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & 
majorVersionYear & "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' open a document
Set objDoc = objSpy.Documents.OpenFile (docPath, False)
Call WScript.ConnectObject(objDoc, "DocEvent_")

' keep running while waiting on the event
' in the meantime close the document in XMLSpy manually
Call WScript.Echo ("sleeping for 10 seconds ...")
Call WScript.Sleep (10000)

Set objDoc = Nothing
Call WScript.Echo ("stopped listening for event")
Call objSpy.Quit

Note: For 32-bit Authentic Desktop, the registered name, or programmatic identifier (ProgId) of the COM object is AuthenticDesktop.Application. For 64-bit Authentic Desktop, the name is AuthenticDesktop_x64.Application.

Example: Using Events

Authentic View supports event connection on a per-object basis. Implementation of this feature is based on COM connection points and is available in environments that support this mechanism.

The following example is a VBScript code example that shows how to use events from within a VBScript project.

' ==========================================================================
' VBScript example that demonstrates how to use events.
' ==========================================================================

' Event handler for OnSelectionChanged event of AuthenticView
Function AuthenticViewEvent_OnSelectionChanged(objAuthenticRange)
If objAuthenticRange.FirstTextPosition <> objAuthenticRange.LastTextPosition Then
Call WScript.Echo("Selection: " & objAuthenticRange.Text & vbNewLine & 
vbNewLine & "Close this dialog.")
Else
Call WScript.Echo("Cursor position: " & 
objAuthenticRange.FirstTextPosition & vbNewLine & 
vbNewLine & "Close this dialog.")
End If
End Function

' Start/access XMLSpy and connect to its automation interface.
Set WshShell = WScript.CreateObject("WScript.Shell")
Set objSpy = GetObject("", "XMLSpy.Application")
' Make the UI of XMLSpy visible.
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed XMLSpy examples.
personalFolder = WshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & majorVersionYear & "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' Create object to access windows file system and test if the our document exists.
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(docPath) Then
    ' open the document
    Call objSpy.Documents.OpenFile(docPath, False)
    Set objDoc = objSpy.ActiveDocument

    ' switch active document to authentic view
    objDoc.SwitchViewMode 4 ' spyViewAuthentic

    ' Register for connection point events on the authentic view of the active document.
    ' Any function with a valid event name prefixed with "AuthenticViewEvent_" will
    ' be called when the corresponding event gets triggered on the specified object.
    Set objView = objDoc.AuthenticView
    Call WScript.ConnectObject(objView, "AuthenticViewEvent_")
    Call WScript.Echo("Events are connected.
        Now set or move the cursor in XMLSpy." & vbNewLine & vbNewLine & "Close this
dialog to shut down XMLSpy.")

    ' To disconnect from the events delete the reference to the object.
    Set objView = Nothing
Else
    Call WScript.Echo("The file " & docPath & " does not exist.")
End If

' shut down XMLSpy when this script ends
objSpy.Visible = False

C#

The C# programming language can be used to access the Application API functionality. You could use Visual Studio 2010/2012/2013/2015/2017 to create the C# code, saving it in a Visual Studio project. Create the project as follows:

1. In Microsoft Visual Studio, add a new project using File | New | Project.
2. Add a reference to the Authentic Desktop Type Library by clicking Project | Add Reference. The Add Reference dialog appears. Browse for the Authentic Desktop Type Library component, which is located in the Authentic Desktop application folder, and add it.
3. Enter the code you want.
4. Compile the code and run it.
Example C# project
Your Authentic Desktop package contains an example C# project, which is located in the C# folder of the API Examples folder:

| Windows 7, Windows 8, Windows 10 | C:\Users\<username>\Documents\Altova\Authentic2019\Authentic |

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017.

The code listing below shows how basic application functionality can be used. This code is similar to the example C# project in the API Examples folder of your application package, but might differ slightly.

Platform configuration
If you have a 64-bit operating system and are using a 32-bit installation of Authentic Desktop, you must add the x86 platform in the solution’s Configuration Manager and build the sample using this configuration.

A new x86 platform (for the active solution in Visual Studio) can be created in the New Solution Platform dialog (Build | Configuration Manager | Active solution platform | <New…>).

What the code listing below does
The example code listing below creates a simple user interface (screenshot below) with buttons that invoke basic Authentic Desktop operations:

- **Start Authentic Desktop**: Starts Authentic Desktop, which is registered as an automation server, or activates the application if it is already running.
- **Open OrgChart.pxf**: Locates one of the example documents installed with Authentic Desktop and opens it. If this document is already open it becomes the active document.
- **Open ExpReport.xml**: Opens another example document.
- **Toggle View Mode**: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- **Validate**: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- **Shut down Authentic Desktop**: Stops Authentic Desktop.

You can modify the code (of the code listing below or of the example C# project in the API Examples folder) in any way you like and run it.

### Compiling and running the example
In the API Examples folder, double-click the file AutomateAuthenticDesktop_VS2008.sln or the file AutomateAuthenticDesktop_VS2010.sln (to open in Visual Studio 2010/2012/2013/2015/2017). Alternatively, the file can be opened from within Visual Studio (with File | Open | Project/Solution). To compile and run the example, select Debug | Start Debugging or Debug | Start Without Debugging.

### Code listing of the example
Given below is the C# code listing of the basic functionality of the form (Form1.cs) created in the AutomateAuthenticDesktop example. Note that the code listed below might differ slightly from the code in the API Examples form. The listing below is commented for ease of understanding. Parts of the code are also presented separately in the sub-sections of this section, according to the Application API functionality they access.

The code essentially consists of a series of handlers for the buttons in the user interface shown in the screenshot above.

```csharp
namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // An instance of AuthenticDesktop is accessed via its automation interface.
        XMLSpyLib.Application AuthenticDesktop;

        // Location of examples installed with AuthenticDesktop
        String strExamplesFolder;

        private void Form1_Load(object sender, EventArgs e)
        {
            // Locate examples installed with AuthenticDesktop.
            // REMARK: You might need to adapt this if you have a different major version of the product.
            strExamplesFolder = Environment.GetEnvironmentVariable("USERPROFILE") + "\My Documents\Altova\Authentic2012\AuthenticExamples\";
        }
```
// Handler for the "Start AuthenticDesktop" button
private void StartAuthenticDesktop_Click(object sender, EventArgs e)
{
    if (AuthenticDesktop == null)
    {
        Cursor.Current = Cursors.WaitCursor;
        // If there is no AuthenticDesktop instance, create one and make
        it visible.
        AuthenticDesktop = new XMLSpyLib.Application();
        AuthenticDesktop.Visible = true;
        Cursor.Current = Cursors.Default;
    }
    else
    {
        // If an AuthenticDesktop instance is already running, make sure
        it's visible.
        if (!AuthenticDesktop.Visible)
            AuthenticDesktop.Visible = true;
    }
}

// Handler for the "Open OrgChart.pxf" button
private void openOrgChart_Click(object sender, EventArgs e)
{
    // Make sure there's a running Authentic Desktop instance, and that
    it's visible
    StartAuthenticDesktop_Click(null, null);
    // Open a sample file installed with the product.
    AuthenticDesktop.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
}

// Handler for the "Open ExpReport.xml" button
private void openExpReport_Click(object sender, EventArgs e)
{
    // Make sure there's a running Authentic Desktop instance, and that
    it's visible
    StartAuthenticDesktop_Click(null, null);
    // Open a sample file installed with the product.
    AuthenticDesktop.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
}

// Handler for the "Toggle View Mode" button
private void toggleView_Click(object sender, EventArgs e)
{
    // Make sure there's a running Authentic Desktop instance, and that
    it's visible
    StartAuthenticDesktop_Click(null, null);
    // Iterate through all open documents and toggle the current view
    between Text View and Authentic View.
    foreach (XMLSpyLib.Document doc in AuthenticDesktop.Documents)
if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewAuthentic)
    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewBrowser);
else
    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);

// Handler for the "Shut down AuthenticDesktop" button
// Shut down application instance by explicitly releasing the COM object.
private void shutdownAuthenticDesktop_Click(object sender, EventArgs e)
{
    if (AuthenticDesktop != null)
    {
        // Allow shut down of AuthenticDesktop by releasing the UI
        AuthenticDesktop.Visible = false;

        // Explicitly release the COM object
        try
        {
            while (System.Runtime.InteropServices.Marshal.ReleaseComObject(AuthenticDesktop) > 0);
        }
        finally
        {
            // Avoid subsequent access to this object.
            AuthenticDesktop = null;
        }
    }
}

// Handler for the "Validate" button
private void validate_Click(object sender, EventArgs e)
{
    // COM errors get returned to C# as exceptions. Use a try/catch block to handle them.
    try
    {
        // Method 'IsValid' is one of the few functions that use output parameters.
        // Use 'object' type for these parameters.
        object strErrorText = "";
        object nErrorNumber = 0;
        object errorData = null;

        if (!AuthenticDesktop.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData))
        {
            // The COM call succeeds but the document is not valid.
            // A detailed description of the problem is returned in strErrorText, nErrorNumber and errorData.
            listBoxMessages.Items.Add("Document "+
                       AuthenticDesktop.ActiveDocument.Name + " is not valid.");
            listBoxMessages.Items.Add("\tErrorText : " + strErrorText);
            listBoxMessages.Items.Add("\tErrorNumber: " + nErrorNumber);
            listBoxMessages.Items.Add("\tElement    : " + (errorData !=
                       null ? (XMLSpyLib.XMLData)errorData).TextValue : "null");
        }
    }
else

{ // The COM call succeeds and the document is valid.
  listBoxMessages.Items.Add("Document " +
  AuthenticDesktop.ActiveDocument.Name + " is valid.");
} }

catch (Exception ex)
{
  // The COM call was not successful.
  // Probably no application instance has been started or no
  // document is open.
  listBoxMessages.Items.Add("Error validating active document: " +
  ex.Message);
}

delegate void addListBoxItem_delegate(string sText);
// Called from the UI thread
private void addListBoxItem(string sText)
{
  listBoxMessages.Items.Add(sText);
}
// Wrapper method to call UI control methods from a worker thread
void syncWithUIthread(Control ctrl, addListBoxItem_delegate
  methodToInvoke, String sText)
{
  // Control.Invoke: Executes on the UI thread, but calling thread
  // waits for completion before continuing.
  // Control.BeginInvoke: Executes on the UI thread, and calling
  // thread doesn't wait for completion.
  if (ctrl.InvokeRequired)
    ctrl.BeginInvoke(methodToInvoke, new Object[] { sText });
}

// Event handler for OnDocumentOpened event
private void handleOnDocumentOpened(XMLSpyLib.Document i_ipDocument)
{
  String sText = "";

  if (i_ipDocument.Name.Length > 0)
    sText = "Document " + i_ipDocument.Name + " was opened!";
  else
    sText = "An empty document was created.";

  // Synchronize the calling thread with the UI thread because
  // COM events are triggered from a working thread
  addListBoxItem_delegate methodToInvoke = new
  addListBoxItem_delegate(addListBoxItem);
  // Call syncWithUIthread with the following arguments:
  // 1 - listBoxMessages - list box control to display messages from
  // COM events
  // 2 - methodToInvoke - a C# delegate which points to the method
  // which will be called from the UI thread
  // 3 - sText - the text to be displayed in the list box
  syncWithUIthread(listBoxMessages, methodToInvoke, sText);
}

private void checkBoxEventOnOff_CheckedChanged(object sender, EventArgs e)
© 2019 Altova GmbH

{ 
  if (AuthenticDesktop != null) 
  { 
    if (checkBoxEventOnOff.Checked) 
      AuthenticDesktop.OnDocumentOpened += new 
      XMLSpyLib._IAplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpend);
    else
      AuthenticDesktop.OnDocumentOpened -= new 
      XMLSpyLib._IAplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpend);
  }
}

Add Reference to Authentic Desktop API

Add the application’s type library as a reference in a .NET project as follows: With the .NET project open, click Project | Add Reference. Then browse for the type library, which is called Authentic Desktop.tlb, and is located in the Authentic Desktop application folder.

Then declare a variable to access the Authentic Desktop API:

```csharp
  // An instance of Authentic Desktop is accessed via its automation
  interface.
  XMLSpyLib.Application Authentic Desktop;
```

Application Startup and Shutdown

In the code snippets below, the methods StartAuthenticDesktop_Click and ShutdownAuthenticDesktop_Click are those assigned to buttons in the AutomateAuthenticDesktop example that, respectively, start up and shut down the application. This example is located in the C# folder of the API Examples folder (see the file Form1.cs):

| Windows 7, Windows 8, Windows 10 | C:\Users\<username>\Documents\Altova\Authentic2019\Authentic |

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017.

Starting Authentic Desktop

The following code snippet from the AutomateAuthenticDesktop example shows how to start up the application.

```csharp
  // Handler for the "Start AuthenticDesktop" button
  private void StartAuthenticDesktop_Click(object sender, EventArgs e) 
  { 
    if (AuthenticDesktop == null) 
    { 
      Cursor.Current = Cursors.WaitCursor;
```
// If there is no AuthenticDesktop instance, create one and make it visible.
AuthenticDesktop = new XMLSpyLib.Application();
AuthenticDesktop.Visible = true;

// If there is no AuthenticDesktop instance, create one and make it visible.
AuthenticDesktop = new XMLSpyLib.Application();
AuthenticDesktop.Visible = true;

else
{
// If an instance of Authentic Desktop is already running, make sure it's visible
if (!AuthenticDesktop.Visible)
AuthenticDesktop.Visible = true;
}

Shutting down Authentic Desktop
The following code snippet from the AutomateAuthenticDesktop example shows how to shut down the application.

// Handler for the "Shut down AuthenticDesktop" button
// Shut down application instance by explicitly releasing the COM object.
private void shutdownAuthenticDesktop_Click(object sender, EventArgs e)
{
    if (AuthenticDesktop != null)
    {
        // Allow shut down of AuthenticDesktop by releasing the UI
        AuthenticDesktop.Visible = false;

        // Explicitly release the COM object
        try
        {
            while (System.Runtime.InteropServices.Marshal.ReleaseComObject(AuthenticDesktop) > 0) ;
        }
        finally
        {
            // Avoid subsequent access to this object.
            AuthenticDesktop = null;
        }
    }
}

Opening Documents
The code snippets below (from the AutomateAuthenticDesktop example) show how two files are opened via two separate methods assigned to two buttons in the user interface. Both methods use the same Application API access mechanism: Documents.OpenFile(string, boolean).

The AutomateAuthenticDesktop example (see the file Form1.cs) is located in the C# folder of the API Examples folder.
You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017.

**Code snippet**

```csharp
// Handler for the "Open OrgChart.pxf" button
private void openOrgChart_Click(object sender, EventArgs e)
{
    // Make sure there's a running Authentic Desktop instance, and that the Sample's View is visible
    StartAuthenticDesktop_Click(null, null);

    // Open a sample file installed with the product.
    AuthenticDesktop.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
}

// Handler for the "Open ExpReport.xml" button
private void openExpReport_Click(object sender, EventArgs e)
{
    // Make sure there's a running Authentic Desktop instance, and that the Sample's View is visible
    StartAuthenticDesktop_Click(null, null);

    // Open a sample file installed with the product.
    AuthenticDesktop.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
}
```

The file opened last will be the active file.

**Iterating through Open Documents**

The code snippet below (from the AutomateAuthenticDesktop example; see the file Form1.cs) shows how to iterate through open documents. A condition is then tested within the iteration loop, and the document view is switched between Text View and Authentic View.

```csharp
// Handler for the "Toggle View Mode" button
private void toggleView_Click(object sender, EventArgs e)
{
    // Make sure there's a running Authentic Desktop instance, and that it's visible
    StartAuthenticDesktop_Click(null, null);

    // Iterate through all open documents and toggle the current view between Text View and Authentic View.
    foreach (XMLSpyLib.Document doc in AuthenticDesktop.Documents)
        if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewAuthentic)
            doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewBrowser);
        else
            doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
```
The **AutomateAuthenticDesktop example** example is located in the C# folder of the API Examples folder:

| Windows 7, Windows 8, Windows 10 | C:\Users\<username>\Documents\Altova\Authentic2019\Authentic |

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017.

**Errors and COM Output Parameters**

The code snippet below (from the **AutomateAuthenticDesktop example**) shows how to handle errors and COM output parameters. The method `AuthenticDesktop.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData)` uses output parameters that are used, in the code snippet below, to generate an error-message text.

The **AutomateAuthenticDesktop example** (see the file Form1.cs) is located in the C# folder of the API Examples folder:

| Windows 7, Windows 8, Windows 10 | C:\Users\<username>\Documents\Altova\Authentic2019\Authentic |

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017.

**Code snippet**

```csharp
// Handler for the "Validate" button
private void validate_Click(object sender, EventArgs e)
{
    // COM errors get returned to C# as exceptions. Use a try/catch block to handle them.
    try
    {
        // Method 'IsValid' is one of the few functions that use output parameters.
        // Use 'object' type for these parameters.
        object strErrorText = "";
        object nErrorNumber = 0;
        object errorData = null;

        if (!AuthenticDesktop.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData))
        {
            // The COM call succeeds but the document is not valid.
            // A detailed description of the problem is returned in
            strErrorText, nErrorNumber and errorData.
            listBoxMessages.Items.Add("Document " +
            AuthenticDesktop.ActiveDocument.Name + " is not valid.");
            listBoxMessages.Items.Add("\nErrorText : " + strErrorText);
```
listBoxMessages.Items.Add("\tErrorNumber: " + nErrorNumber);
listBoxMessages.Items.Add("\tElement : " + (errorData !=
null ? ((XMLSpyLib.XMLData)errorData).TextValue : "null"));
}
else
{
    // The COM call succeeds and the document is valid.
    listBoxMessages.Items.Add("Document " +
AuthenticDesktop.ActiveDocument.Name + " is valid.");
}
}
catch (Exception ex)
{
    // The COM call was not successful.
    // Probably no application instance has been started or no
document is open.
    listBoxMessages.Items.Add("Error validating active document: " +
ex.Message);
}
}

Events

The code snippet below (from the AutomateAuthenticDesktop example) lists the code for two event handlers. The AutomateAuthenticDesktop example (see the file Form1.cs) is located in the C# folder of the API Examples folder:

```
| Windows 7, Windows 8, Windows 10 | C:\Users\<username>\Documents\Altova\Authentic2019\Authentic |
```

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017.

delegate void addListBoxItem_delegate(string sText);
// Called from the UI thread
private void addListBoxItem(string sText)
{
    listBoxMessages.Items.Add(sText);
}
// Wrapper method to call UI control methods from a worker thread
void syncWithUIthread(Control ctrl, addListBoxItem_delegate
methodToInvoke, String sText)
{
    // Control.Invoke: Executes on the UI thread, but calling thread
    // waits for completion before continuing.
    // Control.BeginInvoke: Executes on the UI thread, and calling
    // thread doesn't wait for completion.
    if (ctrl.InvokeRequired)
        ctrl.BeginInvoke(methodToInvoke, new Object[] { sText });
}

// Event handler for OnDocumentOpened event
private void handleOnDocumentOpened(XMLSpyLib.Document i_ipDocument)
{
    String sText = ""
    if (i_ipDocument.Name.Length > 0)
sText = "Document " + i_ipDocument.Name + " was opened!";
else
sText = "An empty document was created.";

// Synchronize the calling thread with the UI thread because
// COM events are triggered from a working thread
addListBoxItem_delegate methodToInvoke = new
addListBoxItem_delegate(addListBoxItem);
// Call syncWithUIthread with the following arguments:
// 1 - listBoxMessages - list box control to display messages from
COM events
// 2 - methodToInvoke - a C# delegate which points to the method
which will be called from the UI thread
// 3 - sText           - the text to be displayed in the list box
syncWithUIthread(listBoxMessages, methodToInvoke, sText);
}

private void checkBoxEventOnOff_CheckedChanged(object sender, EventArgs e)
{
    if (AuthenticDesktop != null)
    {
        if (checkBoxEventOnOff.Checked)
            AuthenticDesktop.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
        else
            AuthenticDesktop.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
    }
}

Java

The Application API can be accessed from Java code. To allow accessing the Authentic Desktop automation server directly from Java code, the libraries listed below must reside in the classpath. They are installed in the folder: JavaAPI in the Authentic Desktop application folder.

- AltovaAutomation.dll: a JNI wrapper for Altova automation servers
  (AltovaAutomation_x64.dll in the case of 64-bit versions)
- AltovaAutomation.jar: Java classes to access Altova automation servers
- AuthenticAPI.jar: Java classes that wrap the Authentic Desktop automation interface
- AuthenticAPI_JavaDoc.zip: a Javadoc file containing help documentation for the Java API

Note: In order to use the Java API, the DLL and Jar files must be on the Java Classpath.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details of the example Java project, see the section, Example Java Project.
Rules for mapping the Application API names to Java

The rules for mapping between the Application API and the Java wrapper are as follows:

- **Classes and class names**
  For every interface of the Authentic Desktop automation interface a Java class exists with the name of the interface.

- **Method names**
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with `get` and `set` can be used. If a property does not support write-access, no setter method is available. Example: For the `Name` property of the `Document` interface, the Java methods `getName` and `setName` are available.

- **Enumerations**
  For every enumeration defined in the automation interface, a Java enumeration is defined with the same name and values.

- **Events and event handlers**
  For every interface in the automation interface that supports events, a Java interface with the same name plus 'Event' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus 'DefaultHandler'. For example:

    ```java
    Application: Java class to access the application
    ApplicationEvents: Events interface for the Application
    ApplicationEventsDefaultHandler: Default handler for ApplicationEvents
    ```

Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Java name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document, method SetEncoding</td>
<td>setFileEncoding</td>
</tr>
<tr>
<td>AuthenticView, method Goto</td>
<td>gotoElement</td>
</tr>
<tr>
<td>AuthenticRange, method Goto</td>
<td>gotoElement</td>
</tr>
<tr>
<td>AuthenticRange, method Clone</td>
<td>cloneRange</td>
</tr>
</tbody>
</table>

This section

This section explains how some basic Authentic Desktop functionality can be accessed from Java code. It is organized into the following sub-sections:

- **Example Java Project**
- **Application Startup and Shutdown**
- **Simple Document Access**
- **Iterations**
Example Java Project

The Authentic Desktop installation package contains an example Java project, located in the Java folder of the API Examples folder:

| Windows 7, Windows 8, Windows 10 | C:\Users\<username>\Documents\Altova\Authentic2019\Authentic |

This folder contains Java examples for the Authentic Desktop API. You can test it directly from the command line using the batch file BuildAndRun.bat, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

File list

The Java examples folder contains all the files required to run the example project. These files are listed below. If you are using a 64-bit version of the application, some filenames contain _x64 in the name. These filenames are indicated with (_x64).

- AltovaAutomation(_x64).dll: Java-COM bridge: DLL part
- AltovaAutomation.jar: Java-COM bridge: Java library part
- AuthenticAPI.jar: Java classes of the Authentic Desktop API
- RunAuthenticDesktop.java: Java example source code
- BuildAndRun.bat: Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
- .classpath: Eclipse project helper file
- .project: Eclipse project file
- Authentic_JavaDoc.zip: Javadoc file containing help documentation for the Java API

What the example does

The example starts up Authentic Desktop and performs a few operations, including opening and closing documents. When done, Authentic Desktop stays open. You must close it manually.

- **Start Authentic Desktop:** Starts Authentic Desktop, which is registered as an automation server, or activates Authentic Desktop if it is already running.
- **Open example files:** Locates example documents installed with Authentic Desktop and opens them.
- **Iteration and Changing the View Mode:** Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- **Iteration, validation, output parameters:** Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- **Event Handling:** Shows how to handle Authentic Desktop events.
- **Shut down Authentic Desktop:** Shuts down Authentic Desktop.

You can modify the example in any way you like and run it.

**Running the example from the command line**
To run the example from the command line, open a command prompt window, go to the Java folder of the API Examples folder (see above for location), and then type:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

The Java binary folder must be that of a JDK 1.5 or later installation on your computer.

Press the Return key. The Java source in `RunAuthenticDesktop.java` will be compiled and then executed.

**Loading the example in Eclipse**
Open Eclipse and use the `Import | Existing Projects into Workspace` command to add the Eclipse project file (.project) located in the Java folder of the API Examples folder (see above for location). The project `RunAuthenticDesktop` will then appear in your Package Explorer or Navigator.

Select the project and then the command `Run as | Java Application` to execute the example.

**Note:** You can select a class name or method of the Java API and press F1 to get help for that class or method.

**Java source code listing**
The Java source code in the example file `RunAuthenticDesktop.java` is listed below with comments.

```java
01 // Access general JAVA-COM bridge classes
02 import com.altova.automation.libs.*;
03
04 // Access AuthenticDesktop Java-COM bridge
05 import com.altova.automation.AuthenticDesktop.*;
06 import com.altova.automation.AuthenticDesktop.Enums.SPYViewModes;
07
08 /**
09  * A simple example that starts AuthenticDesktop COM server and performs a
10  * view operations on it.
11  * Feel free to extend.
12  */
13 public class RunAuthenticDesktop
14 {
15    public static void main(String[] args)
16    {
17      // An instance of the application.
18      Application authenticDesktop = null;
```
try {
    // Start AuthenticDesktop as COM server.
    authenticDesktop = new Application();
    // COM servers start up invisible so we make it visible
    authenticDesktop.setVisible(true);

    // Locate samples installed with the product.
    String strExamplesFolder = System.getenv("USERPROFILE") + "\My Documents\Altova\Authentic2012\AuthenticExamples\"
    authenticDesktop.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
    authenticDesktop.getDocuments().openFile(strExamplesFolder + "ExpReport.xml", false);

    // Iterate through all open documents and set the View Mode to 'Text'.
    for (Document doc:authenticDesktop.getDocuments())
        if (doc.getCurrentViewMode() != SPYViewModes.spyViewText)
            doc.switchViewMode(SPYViewModes.spyViewText);

    // An alternative iteration mode is index-based. COM indices are typically zero-based.
    Documents documents = authenticDesktop.getDocuments();
    for (int i = 1; i <= documents.getCount(); i++)
        Document doc = documents.getItem(i);

    // Validation is one of the few methods that have output parameters.
    // The class JVariant is the correct type for parameters in these cases.
    JVariant validationErrorText = new JVariant.JStringVariant(""); validationErrorText.setByRefFlag();
    JVariant validationErrorCount = new JVariant.JIntVariant(0); validationErrorCount.setByRefFlag();
    JVariant validationErrorXMLData = new JVariant.JIDispatchVariant(0); validationErrorXMLData.setByRefFlag();
    if (!doc.isValid(validationErrorText, validationErrorCount, validationErrorXMLData))
        System.out.println("Document " + doc.getName() + " is not wellformed - " + validationErrorText.getStringValue());
    else
        System.out.println("Document " + doc.getName() + " is wellformed.");

    // The following lines attach to the document events using a default implementation
    // for the events and override one of its methods.
    if you want to override all document events it is better to derive your listener class
    from DocumentEvents and implement all methods of this interface.
    Document doc = authenticDesktop.getActiveDocument();
    doc.addListener(new DocumentEventsDefaultHandler() {
```java
@Override
public boolean onBeforeCloseDocument(Document i_ipDoc) throws AutomationException {
    System.out.println("Document " + i_ipDoc.getName() + " requested closing.");
    // Allow closing of document
    return true;
}
```

```java
doc.close(true);
doc = null;
System.out.println("Watch AuthenticDesktop!");
```

```
catch (AutomationException e) {
    // e.printStackTrace();
}
```

```
finally {
    // Make sure that AuthenticDesktop can shut down properly.
    if (authenticDesktop != null)
        authenticDesktop.dispose();

    // Since the COM server was made visible and still is visible, it will keep running
    // and needs to be closed manually.
    System.out.println("Now close AuthenticDesktop!");
}
```

**Application Startup and Shutdown**

The code listings below show how the application can be started up and shut down.

**Application startup**

Before starting up the application, the appropriate classes must be imported (see below).

```
// Access general JAVA-COM bridge classes
import com.altova.automation.libs.*;

// Access AuthenticDesktop Java-COM bridge
import com.altova.automation.AuthenticDesktop.*;
import com.altova.automation.AuthenticDesktop.Enums.SPYViewModes;

/**
 * A simple example that starts AuthenticDesktop COM server and performs a view operations on it.
 * Feel free to extend.
 */
public class RunAuthenticDesktop {
    public static void main(String[] args)
```
15 {  
16     // An instance of the application.
17     Application authenticDesktop = null;
18     // Instead of COM error-handling, use Java exception mechanism.
19     try  
20         // Start AuthenticDesktop as COM server.
21         authenticDesktop = new Application();
22         // COM servers start up invisible so we make it visible
23         authenticDesktop.setVisible(true);
24         ...
25     }  
26     
27     Application shutdown
28     The application can be shut down as shown below.
29     
30     {  
31         // Make sure that AuthenticDesktop can shut down properly.
32         if (authenticDesktop != null)  
33             authenticDesktop.dispose();  
34         // Since the COM server was made visible and still is visible, it will keep running
35         // and needs to be closed manually.
36         System.out.println("Now close AuthenticDesktop!");  
37     }
38
39     Simple Document Access
40     The code listing below shows how to open a document.
41     
42     // Locate samples installed with the product.
43     String strExamplesFolder = System.getenv("USERPROFILE") + "\My Documents\"Altova\Authentic2012\AuthenticExamples\";
44     // Open two files from the product samples.
45     authenticDesktop.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
46     authenticDesktop.getDocuments().openFile(strExamplesFolder + "ExpReport.xml", false);
47
48     Iterations
49     The listing below shows how to iterate through open documents.
50     
51     // Iterate through all open documents and set the View mode to ‘Text’.
52     for (Document doc:authenticDesktop.getDocuments())
if (doc.getCurrentViewMode() != SPYViewModes.spyViewText)
    doc.switchViewMode(SPYViewModes.spyViewText);
// An alternative iteration mode is index-based. COM indices are typically zero-based.
Documents documents = authenticDesktop.getDocuments();
for (int i = 1; i <= documents.getCount(); i++)
{
    Document doc = documents.getItem(i);
    ...
}

Use of Out-Parameters

The code listing below iterates through open documents and validates each of them. For each validation, a message is generated using the output parameters of the Validation method.

01 // Iterate through all open documents and set the View mode to 'Text'.
02 for (Document doc:authenticDesktop.getDocuments())
03     if (doc.getCurrentViewMode() != SPYViewModes.spyViewText)
04         doc.switchViewMode(SPYViewModes.spyViewText);
05 // An alternative iteration mode is index-based. COM indices are typically zero-based.
07 Documents documents = authenticDesktop.getDocuments();
08 for (int i = 1; i <= documents.getCount(); i++)
09 {
10     Document doc = documents.getItem(i);
11     ...
12 }

Event Handlers

The listing below shows how to listen for and use events.

01 // The following lines attach to the document events using a default implementation
02 // for the events and override one of its methods.
03 // If you want to override all document events, it is better to derive your
listener class
04 // from DocumentEvents and implement all methods of this interface.
05 Document doc = authenticDesktop.getActiveDocument();
06 doc.addListener(new DocumentEventsDefaultHandler()
07 { 
08   @Override
09   public boolean onBeforeCloseDocument(Document i_ipDoc) throws 
10     AutomationException
11       { System.out.println("Document "+ i_ipDoc.getName() + " requested 
12         closing.");
13       // allow closing of document
14       return true;
15     }
16 });
17 doc.close(true);
18 doc = null;
3.2 Interfaces

Object Hierarchy

Application
   SpyProject
      SpyProjectItems
      SpyProjectItem
   Documents
      Document
         GridView
         AuthenticView
            AuthenticRange
            AuthenticDataTransfer (previously DocEditDataTransfer)
            AuthenticDataTransfer (previously DocEditDataTransfer)
      TextView
      XMLData
   Dialogs
      CodeGeneratorDlg
      FileSelectionDlg
      SchemaDocumentationDlg
      GenerateSampleXMLDlg
      DTDSchemaGeneratorDlg
      FindInFilesDlg
   DatabaseConnection
   ExportSettings
   TextImportExportSettings
   ElementList
      ElementListItem

Enumerations

Description
This chapter contains the reference of the Authentic Desktop 1.5 Type Library.

Most of the given examples are written in VisualBasic. These code snippets assume that there is a variable defined and set, called **objSpy of type Application**. There are also some code samples written in JavaScript.
3.2.1 Application

See also

Methods
- GetDatabaseImportElementList
- GetDatabaseSettings
- GetDatabaseTables
- ImportFromDatabase
- CreateXMLSchemaFromDBStructure
- GetTextImportElementList
- GetTextImportExportSettings
- ImportFromText
- ImportFromWord
- ImportFromSchema
- GetExportSettings
- NewProject
- OpenProject
- AddMacroMenuItem
- ClearMacroMenu
- ShowForm
- ShowApplication
- URLDelete
- URLMakeDirectory
- AddXSLT_XQParameter
- GetXSLT_XQParameterCount
- GetXSLT_XQParameterName
- GetXSLT_XQParameterXPath
- RemoveXSLT_XQParameter
- FindInFiles
- Quit

Properties
- Application
- Parent
- ActiveDocument
- Documents
- CurrentProject
Application is the root for all other objects. It is the only object you can create by CreateObject (VisualBasic) or other similar COM related functions.

Example

```
Dim objSpy As Application
Set objSpy = CreateObject("XMLSpy.Application")
```

Events

**OnBeforeOpenDocument**

See also

**Event:** `OnBeforeOpenDocument(objDialog as FileSelectionDlg)`

Description

This event gets fired whenever a document gets opened via the OpenFile or OpenURL menu command. It is sent after a document file has been selected but before the document gets opened. The file selection dialog object is initialized with the name of the selected document file. You can modify this selection. To continue the opening of the document leave the `FileSelectionDlg.DialogAction` property of `io_objDialog` at its default value `spyDialogOK`. To abort the opening of the document set this property to `spyDialogCancel`.

Examples

Given below are examples of how this event can be scripted.

**XMLSpy scripting environment - VBScript:**

```
Function On_BeforeOpenDocument(objDialog)
End Function
```

**XMLSpy scripting environment - JScript:**

```
function On_BeforeOpenDocument(objDialog)
{
}
```

**XMLSpy IDE Plugin:**

```
IXMLSpyPlugIn.OnEvent(26, ...) // nEventId = 26
```
OnBeforeOpenProject

See also

Event: OnBeforeOpenProject(objDialog as FileSelectionDlg)

Description
This event gets fired after a project file has been selected but before the project gets opened. The file selection dialog object is initialized with the name of the selected project file. You can modify this selection. To continue the opening of the project leave the FileSelectionDlg.DialogAction property of io_objDialog at its default value spyDialogOK. To abort the opening of the project set this property to spyDialogCancel.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
Function On_BeforeOpenProject(objDialog)
End Function

XMLSpy scripting environment - JScript:
function On_BeforeOpenProject(objDialog)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (25, ...) // nEventId = 25

OnDocumentOpened

See also


Description
This event gets fired whenever a document opens in Authentic Desktop. This can happen due to opening a file with the OpenFile or OpenURL dialog, creating a new file or dropping a file onto Authentic Desktop. The new document gets passed as parameter. The operation cannot be canceled.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
Function On_OpenDocument(objDocument)
End Function

XMLSpy scripting environment - JScript:
function On_OpenDocument(objDocument)
OnProjectOpened

See also

**Event:** OnProjectOpened(*objProject* as SpyProject)

**Description**

This event gets fired whenever a project gets opened in Authentic Desktop. The new project gets passed as parameter.

**Examples**

Given below are examples of how this event can be scripted.

**XMLSpy scripting environment - VBScript:**

```vbnet
Function On_OpenProject(objProject)
End Function
```

**XMLSpy scripting environment - JScript:**

```jscript
function On_OpenProject(objProject)
{
}
```

**XMLSpy IDE Plugin:**

```csharp
IXMLSpyPlugIn.OnEvent(7, ...)  // nEventId = 7
```

ActiveDocument

See also

**Property:** ActiveDocument as Document

**Description**

Reference to the active document. If no document is open, ActiveDocument is null (nothing).

**Errors**

- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.

AddMacroMenuItem

See also
Method: `AddMacroMenuItem(strMacro as String, strDisplayText as String)`

Description
Adds a menu item to the Tools menu. This new menu item invokes the macro defined by `strMacro`. See also "Calling macros from Authentic Desktop".

Errors
- 1111 The application object is no longer valid.
- 1100 Invalid parameter or invalid address for the return parameter was specified.
- 1108 Number of macro items is limited to 16 items.

AddXSLT_XQParameter

Method: `AddXSLT_XQParameter(name as String, XPath as String)`

Description
Adds an XSLT or XQuery parameter. The parameter's name and value are the two arguments of the method.

Errors
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.
- 1124 The XPath expression is not set.
- 1125 Not a QName.
- 1126 The specified XPath is not valid. Reason for invalidity appended.
- 1127 A parameter with the submitted name already exists.

Application

See also

Property: `Application as Application` (read-only)

Description
Accesses the Authentic Desktop application object.

Errors
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.

ClearMacroMenu

See also

Method: `ClearMacroMenu()`

Return Value
None

Description
Removes all menu items from the **Tools** menu. See also [Running macros](#).

### Errors

- **1111** The application object is no longer valid.

### CreateXMLSchemaFromDBStructure

**See also**

**Method:** CreateXMLSchemaFromDBStructure(\(p\text{ImportSettings as DatabaseConnection}, \ p\text{Tables as ElementList}\))

**Description**

CreateXMLSchemaFromDBStructure creates from a database specified in \(p\text{ImportSettings}\) for the defined tables in \(p\text{Tables}\) new XML Schema document(s) describing the database tables structure.

The parameter \(p\text{Tables}\) specifies which table structures the XML Schema document should contain. This parameter can be NULL, specifying that all table structures will be exported.

See also [GetDatabaseTables](#).

**Errors**

- **1112** Invalid database specified.
- **1120** Database import failed.

### CurrentProject

**See also**

**Property:** CurrentProject as SpyProject

**Description**

Reference to the active document. If no project is open, CurrentProject is null (nothing).

**Errors**

- **1111** The application object is no longer valid.
- **1100** Invalid address for the return parameter was specified.

### Dialogs

**See also**

**Property:** Dialogs as Dialogs (read-only)

**Description**

Access the built-in dialogs of Authentic Desktop.

**Errors**

- **1111** The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Documents
See also

Property: Documents as Documents

Description
Collection of all open documents.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Edition
See also

Property: Edition as String

Description
Returns the edition of the application, for example Altova Authentic Desktop Enterprise Edition for the Enterprise edition.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

FindInFiles
See also

Method: FindInFiles(pSettings as FindInFilesDlg) as FindInFilesResults

Description
Returns a FindInFilesResults object containing information about the files that matched the specified settings.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetDatabaseImportElementList
See also

Method: GetDatabaseImportElementList(pImportSettings as DatabaseConnection) as ElementList

Description
The function returns a collection of ElementListItems where the properties
**ElementListItem.Name** contain the names of the fields that can be selected for import and the properties **ElementListItem.ElementKind** are initialized either to **spyXMLDataAttr** or **spyXMLDataElement**, depending on the value passed in **DatabaseConnection.AsAttributes**. This list serves as a filter to what finally gets imported by a future call to **ImportFromDatabase**. Use **ElementList.RemoveElement** to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of **DatabaseConnection.File**, **DatabaseConnection.ADOConnection** and **DatabaseConnection.ODBCConnection**, as well as **DatabaseConnection.SQLSelect**. Use the property **DatabaseConnection.AsAttributes** to initialize **ElementListItem.ElementKind** of the resulting element list to either **spyXMLDataAttr** or **spyXMLDataElement**, respectively.

**Example**
See example at **ImportFromDatabase**.

**Errors**

- 1111  The application object is no longer valid.
- 1100  Invalid parameter or invalid address for the return parameter was specified.
- 1107  Import from database failed.
- 1112  Invalid database specified.
- 1114  Select statement is missing.
- 1119  database element list import failed.

**GetDatabaseSettings**

**See also**

**Method**: **GetDatabaseSettings() as DatabaseConnection**

**Description**

**GetDatabaseSettings** creates a new object of database settings. The object is used to specify database connection parameters for the methods **GetDatabaseTables**, **GetDatabaseImportElementList**, **ImportFromDatabase**, **ImportFromSchema** and **ExportToDatabase**.

**Example**
See example of **ImportFromDatabase**.

**Errors**

- 1111  The application object is no longer valid.
- 1100  Invalid address for the return parameter was specified.

**GetDatabaseTables**

**See also**

**Method**: **GetDatabaseTables(pImportSettings as DatabaseConnection) as ElementList**

**Description**
GetDatabaseTables reads the table names from the database specified in `pImportSettings`. Properties mandatory to be filled out for the database connection are one of `DatabaseConnection.File`, `DatabaseConnection.ADOConnection` and `DatabaseConnection.ODBCConnection`. All other properties are ignored. The function returns a collection of `ElementListItems` where the properties `ElementListItem.Name` contain the names of tables stored in the specified database. The remaining properties of `ElementListItem` are unused.

Errors

1111  The application object is no longer valid.
1100  Invalid parameter or invalid address for the return parameter was specified.
1112  Invalid database specified.
1113  Error while reading database table information.
1118  Database table query failed.

Example

```vbnet
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
objImpSettings.ADOConnection = TxtADO.Text

' store table names in list box
ListTables.Clear

Dim objList As ElementList
Dim objItem As ElementListItem
On Error GoTo ErrorHandler
Set objList = objSpy.GetDatabaseTables(objImpSettings)

For Each objItem In objList
    ListTables.AddItem objItem.Name
Next
```

GetExportSettings

See also

`Method`: `GetExportSettings() as ExportSettings` (read-only)

Description

`GetExportSettings` creates a new object of common export settings. This object is used to pass the parameters to the export functions and defines the behaviour of the export calls. See also the export functions from `Document`.

Errors

1111  The application object is no longer valid.
1100  Invalid address for the return parameter was specified.

GetTextImportElementList

See also

`Method`: `GetTextImportElementList(pImportSettings as`
**TextImportExportSettings** as **ElementList**

**Description**

GetTextImportElementList retrieves importing information about the text-file as specified in pImportSettings. The function returns a collection of ElementListItems where the properties **ElementListItem.Name** contain the names of the fields found in the file. The values of remaining properties are undefined.

If the text-file does not contain a column header, set **pImportSettings.HeaderRow** to **False**. The resulting element list will contain general column names like 'Field1' and so on.

**Errors**

- **1111** The application object is no longer valid.
- **1100** Invalid parameter or invalid address for the return parameter was specified.
- **1107** Import from database failed.
- **1115** Error during text element list import. Cannot create parser for import file.
- **1116** Error during text element list import.

**Example**

```vba
' ---------------------------------------------------------
' VBA client code fragment - import selected fields from text file
' ---------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = "C:\ImportMe.txt"
objImpSettings.HeaderRow = False

Dim objList As ElementList
Set objList = objSpy.GetTextImportElementList(objImpSettings)

'exclude first column
objList.RemoveItem 1

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings, objList)
CheckForError
```

**GetTextImportExportSettings**

**See also**

**Method**: GetTextImportExportSettings() as **TextImportExportSettings** (read-only)

**Description**

GetTextImportExportSettings creates a new object of common import and export settings for text files. See also the example for Application.GetTextImportElementList.

**Errors**

- **1111** The application object is no longer valid.
- **1100** Invalid address for the return parameter was specified.
GetXSLT_XQParameterCount

Method: GetXSLT_XQParameterCount() as Long

Description
Returns the number of XSLT and XQuery parameters.

Errors
- 1111  The application object is no longer valid.
- 1100  Invalid address for the return parameter was specified.

GetXSLT_XQParameterName

Method: GetXSLT_XQParameterName(index as Long) as String

Description
Returns the name of the XSLT or XQuery parameter identified by the supplied index.

Errors
- 1111  The application object is no longer valid.
- 1100  Invalid address for the return parameter was specified.

GetXSLT_XQParameterXPath

Method: GetXSLT_XQParameterXPath(index as Long) as String

Description
Returns the XPath expression of the XSLT or XQuery parameter identified by the supplied index.

Errors
- 1111  The application object is no longer valid.
- 1100  Invalid address for the return parameter was specified.

ImportFromDatabase

See also

Method: ImportFromDatabase(pImportSettings as DatabaseConnection, pElementList as ElementList) as Document

Return Value
Creates a new document containing the data imported from the database.

Description
ImportFromDatabase imports data from a database as specified in pImportSettings and creates a new document containing the data imported from the database. Properties mandatory to
be filled out are one of `DatabaseConnection.File`, `DatabaseConnection.ADOConnection` or `DatabaseConnection.ODBCConnection` and `DatabaseConnection.SQLSelect`. Additionally, you can use `DatabaseConnection.AsAttributes`, `DatabaseConnection.ExcludeKeys`, `DatabaseConnection.IncludeEmptyElements` and `NumberDateTimeFormat` to further parameterize import.

The parameter `pElementList` specifies which fields of the selected data gets written into the newly created document, and which are created as elements and which as attributes. This parameter can be NULL, specifying that all selected fields will be imported as XML elements.

See `GetDatabaseSettings` and `GetDatabaseImportElementList` for necessary steps preceding any import of data from a database.

### Errors

- **1111** The application object is no longer valid.
- **1100** Invalid parameter or invalid address for the return parameter was specified.
- **1107** Import from database failed.
- **1112** Invalid database specified.
- **1114** Select statement is missing.
- **1117** Transformation to XML failed.
- **1120** Database import failed.

### Example

```vba
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings

objImpSettings.ADOConnection = strADOConnection
objImpSettings.SQLSelect = "SELECT * FROM MyTable"

Dim objDoc As Document
On Error Resume Next
Set objDoc = objSpy.ImportFromDatabase(objImpSettings, objSpy.GetDatabaseImportElementList(objImpSettings))
    ' CheckForError here
```

**ImportFromSchema**

See also

**Method:** `ImportFromSchema(pImportSettings as DatabaseConnection, strTable as String, pSchemaDoc as Document) as Document`

**Return Value**

Creates a new document filled with data from the specified database as specified by the schema definition in `pSchemaDoc`.

**Description**

`ImportFromSchema` imports data from a database specified in `pImportSettings`. Properties mandatory to be filled out are one of `DatabaseConnection.File`, `DatabaseConnection.ADOConnection` or `DatabaseConnection.ODBCConnection`. 
Additionally, you can use `DatabaseConnection.AsAttributes`, `DatabaseConnection.ExcludeKeys` and `NumberDateTimeFormat` to further parameterize import. All other properties get ignored.

`ImportFromSchema` does not use and explicit SQL statement to select the data. Instead, it expects a structure definition of the document to create in form of an XML schema document in `pSchemaDoc`. From this definition the database select statement is automatically deduced. Specify in `strTable` the table name of the import root that will become the root node in the new document.

See `GetDatabaseSettings` and `GetDatabaseTables` for necessary steps preceding an import from a database based on a schema definition. To create the schema definition file use command 'create database schema' from the 'convert' menu of Authentic Desktop.

**Errors**

- 1111  The application object is no longer valid.
- 1100  Invalid parameter or invalid address for the return parameter was specified.
- 1107  Import from database failed.
- 1112  Invalid database specified.
- 1120  Database import failed.
- 1121  Could not create validator for the specified schema.
- 1122  Failed parsing schema for database import.

### ImportFromText

**See also**

**Method:** `ImportFromText(pImportSettings as TextImportExportSettings, pElementList as ElementList) as Document`

**Description**

`ImportFromText` imports the text file as specified in `pImportSettings`. The parameter `pElementList` can be used as import filter. Either pass the list returned by a previous call to `GetTextImportElementList` or `null` to import all columns. To avoid import of unnecessary columns use `ElementList.RemoveElement` to remove the corresponding field names from `pElementList` before calling `ImportFromText`.

The method returns the newly created document containing the imported data. This document is the same as the active document of Authentic Desktop.

**Errors**

- 1111  The application object is no longer valid.
- 1100  Invalid parameter or invalid address for the return parameter was specified.
- 1107  Import from text file failed.
- 1117  Transformation to XML failed.

**Example**

```
' VBA client code fragment - import from text file
'---------------------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = strFileName
```
objImpSettings.HeaderRow = False

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings, 
   objSpy.GetTextImportElementList(objImpSettings))

CheckForError

ImportFromWord
See also

*Method*: ImportFromWord(*strFile* as String) as Document

*Description*
ImportFromWord imports the MS-Word Document *strFile* into a new XML document.

*Errors*
- 1111  The application object is no longer valid.
- 1100  Invalid parameter or invalid address for the return parameter was specified.
  Import from document failed.

IsAPISupported
See also

*Property*: IsAPISupported as Boolean

*Description*
Returns whether the API is supported in this version or not.

*Errors*
- 1111  The application object is no longer valid.
- 1100  Invalid address for the return parameter was specified.

MajorVersion
See also

*Property*: MajorVersion as Integer

*Description*
Returns the application version's major number, for example 15 for 2013 versions, and 16 for 2014 versions.

*Errors*
- 1111  The application object is no longer valid.
- 1100  Invalid address for the return parameter was specified.
MinorVersion

See also

Property: MinorVersion as Integer

Description
Returns the application version's minor number.

Errors
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.

NewProject

See also

Method: NewProject (strPath as String, bDiscardCurrent as Boolean)

Description
NewProject creates a new project.

If there is already a project open that has been modified and bDiscardCurrent is false, then NewProject() fails.

Errors
- 1111 The application object is no longer valid.
- 1102 A project is already open but bDiscardCurrent is true.
- 1103 Creation of new project failed.

OpenProject

See also

Method: OpenProject (strPath as String, bDiscardCurrent as Boolean, bDialog as Boolean)

Parameters
strPath
Path and file name of the project to open. Can be empty if bDialog is true.

bDiscardCurrent
Discard currently open project and possible lose changes.

bDialog
Show dialogs for user input.

Return Value
None

Description
OpenProject opens an existing project. If there is already a project open that has been modified and bDiscardCurrent is false, then OpenProject() fails.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1101 Cannot open specified project.
1102 A project is already open but bDiscardCurrent is true.

Parent
See also

Property: Parent as Application (read-only)
Description
Accesses the Authentic Desktop application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Quit
See also

Method: Quit()
Return Value
None
Description
This method terminates Authentic Desktop. All modified documents will be closed without saving the changes. This is also true for an open project.

If Authentic Desktop was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any document is still open. Use the Quit method to ensure automatic shut-down.

Errors
1111 The application object is no longer valid.

ReloadSettings
See also

Method: ReloadSettings
Return Value
Description
The application settings are reloaded from the registry.
RemoveXSLT_XQParameter

*Method:* RemoveXSLT_XQParameter(index as Long)

*Description*
Removes the XSLT or XQuery parameter identified by the supplied index.

*Errors*
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.

RunMacro

*See also*

*Method:* RunMacro(strMacro as String)

*Return Value*

*Description*
Calls the specified macro either from the project scripts (if present) or from the global scripts.

Available with TypeLibrary version 1.5

*Errors*
- 1111 The application object is no longer valid.

ScriptingEnvironment

*See also*

*Property:* ScriptingEnvironment as IUnknown (read-only)

*Description*
Reference to any active scripting environment. This property makes it possible to access the TypeLibrary of the XMLSpyFormEditor.exe application which is used as the current scripting environment.

Available with TypeLibrary version 1.5

*Errors*
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.
ServicePackVersion

See also

Property: ServicePackVersion as Long

Description
Returns the Service Pack version number of the application. Eg: 1 for 2010 R2 SP1

Errors
1111  The application object is no longer valid.
1100  Invalid address for the return parameter was specified.

ShowApplication

See also

Method: ShowApplication(bShow as Boolean)

Return Value
None

Description
The method shows (bShow = True) or hides (bShow = False) Authentic Desktop.

Errors
1110  The application object is no longer valid.

ShowFindInFiles

See also

Method: ShowFindInFiles(pSettings as FindInFilesDlg) as Boolean

Return Value
Returns false if the user pressed the Cancel button, true otherwise.

Description
Displays the FindInFiles dialog preset with the given settings. The user modifications of the settings are stored in the passed dialog object.

Errors
1111  The application object is no longer valid.
1100  Invalid parameter or invalid address for the return parameter was specified.

ShowForm

See also

Method: ShowForm(strFormName as String) as Long

Return Value
Returns zero if the user pressed a Cancel button or the form calls `TheView.Cancel()`.

**Description**
Displays the form `strFormName`.

Forms, event handlers and macros can be created with the Scripting Environment. Select "Switch to scripting environment" from the **Tools** menu to invoke the Scripting Environment.

**Errors**
- 1111 The application object is no longer valid.
- 1100 Invalid parameter or invalid address for the return parameter was specified.

### Status

**See also**

**Property:** `Status` as `ENUMApplicationStatus`

**Description**
Returns the current status of the running application.

**Errors**
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.

### URLDelete

**See also**

**Method:** `URLDelete(strURL as String, strUser as String, strPassword as String)`

**Return Value**
None

**Description**
The method deletes the file at the URL `strURL`.

**Errors**
- 1111 The application object is no longer valid.
- 1109 Error deleting file at specified URL.

### URLMakeDirectory

**See also**

**Method:** `URLMakeDirectory(strURL as String, strUser as String, strPassword as String)`

**Return Value**
None

**Description**
The method creates a new directory at the URL `strURL`.

**Errors**
- 1111 The application object is no longer valid.
- 1100 Invalid parameter specified.

**Visible**

**See also**

*Property:* `Visible` as `VARIANT_BOOL`

*Description*
Sets or gets the visibility attribute of Authentic Desktop. This standard automation property makes usage of `ShowApplication` obsolete.

**Errors**
- 1110 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.

**WarningNumber**

**See also**

*Property:* `WarningNumber` as integer

*Description*
Some methods fill the property `WarningNumber` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

**Errors**
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.

**WarningText**

**See also**

*Property:* `WarningText` as String

*Description*
Some methods fill the property `WarningText` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

**Errors**
- 1111 The application object is no longer valid.
- 1100 Invalid address for the return parameter was specified.
3.2.2 AuthenticContextMenu

The context menu interface provides the mean for the user to customize the context menus shown in Authentic. The interface has the methods listed in this section.

CountItems

*Method:* `CountItems()`  *Return Value:* Returns the number of menu items.

*Errors*

- 2501 Invalid object.

DeleteItem

*Method:* `DeleteItem(IndexPosition as long)`  *Return Value:* Deletes the menu item that has the index position submitted in the first parameter.

*Errors*

- 2501 Invalid object
- 2502 Invalid index

GetItemText

*Method:* `GetItemText(IndexPosition as long) MenuItemName as string`  *Return Value:* Gets the name of the menu item located at the index position submitted in the first parameter.

*Errors*

- 2501 Invalid object
- 2502 Invalid index

InsertItem

*Method:* `InsertItem(IndexPosition as long, MenuItemName as string, MacroName as string)`  *Return Value:* Inserts a user-defined menu item at the position in the menu specified in the first parameter and having the name submitted in the second parameter. The menu item will start a macro, so a valid macro name must be submitted.

*Errors*

- 2501 Invalid object
- 2502 Invalid index
2503  No such macro
2504  Internal error

**SetItemText**

*Method:* `SetItemText(IndexPosition as long, MenuItemName as string)`

**Return Value**
Sets the name of the menu item located at the index position submitted in the first parameter.

**Errors**
- 2501  Invalid object
- 2502  Invalid index
3.2.3 AuthenticDataTransfer

**Renamed from DocEditDataTransfer to AuthenticDataTransfer**

The DocEditView object is renamed to OldAuthenticView. DocEditSelection is renamed to AuthenticSelection. DocEditEvent is renamed to AuthenticEvent. DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage—except for AuthenticDataTransfer—is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

**See also**

**Methods**

getData

**Properties**

dropEffect

ownDrag

type

**Description**

The events OnDragOver and OnBeforeDrop provide information about the object being dragged with an instance of type AuthenticDataTransfer. It contains a description of the dragged object and its content. The latter is available either as string or a pointer to a COM object supporting the IUnknown interface.

**dropEffect**

**See also**

*Property: dropEffect as long*

**Description**

The property stores the drop effect from the default event handler. You can set the drop effect if you change this value and return TRUE for the event handler.

**Errors**

2101 Invalid address for the return parameter was specified.

**getData**

**See also**
**Method:** `getData()` as Variant

**Description**
Retrieve the data associated with the dragged object. Depending on `AuthenticDataTransfer.type`, that data is either a string or a COM interface pointer of type `IUnknown`.

**Errors**
2101 Invalid address for the return parameter was specified.

**ownDrag**

**See also**

**Property:** `ownDrag` as Boolean (read-only)

**Description**
The property is `TRUE` if the current dragging source comes from inside Authentic View.

**Errors**
2101 Invalid address for the return parameter was specified.

**type**

**See also**

**Property:** `type` as String (read-only)

**Description**
Holds the type of data you get with the `DocEditDataTransfer.getData` method.

Currently supported data types are:

- OWN data from Authentic View itself
- TEXT plain text
- UNICODETEXT plain text as UNICODE

**Errors**
2101 Invalid address for the return parameter was specified.
3.2.4 AuthenticEventContext

The EventContext interface gives access to many properties of the context in which a macro is executed.

EvaluateXPath

*Method:* EvaluateXPath (strExpression as string) as strValue as string

*Return Value*
The method evaluates the XPath expression in the context of the node within which the event was triggered and returns a string.

*Description*
EvaluateXPath() executes an XPath expression with the given event context. The result is returned as string, in the case of a sequence it is a space-separated string.

*Errors*
- 2201 Invalid object.
- 2202 No context.
- 2209 Invalid parameter.
- 2210 Internal error.
- 2211 XPath error.

GetEventContextType

*Method:* GetEventContextType () Type as AuthenticEventContextType enumeration

*Return Value*
Returns the context node type.

*Description*
GetEventContextType allows the user to determine whether the macro is in an XML node or in an XPath atomic item context. The enumeration AuthenticEventContextType is defined as follows:

authenticEventContextXML,
authenticEventContextAtomicItem,
authenticEventContextOther

If the context is a normal XML node, the GetXmlNode() function gives access to it (returns NULL if not).

*Errors*
- 2201 Invalid object.
- 2202 No context.
- 2209 Invalid parameter.

GetNormalizedTextValue

*Method:* GetNormalizedTextValue() strValue as string

*Return Value*
Returns the value of the current node as string

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

GetVariableValue

Method: GetVariableValue(strName as string) strValue as string

Return Value
Gets the value of the variable submitted as the parameter.

Description
GetVariableValue gets the variable's value in the scope of the context.

nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

GetXMLNode

Method: GetXMLNode() Node as XMLData object

Return Value
Returns the context XML node or NULL

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

IsAvailable

Method: IsAvailable() as Boolean

Return Value
Returns true if EventContext is set, false otherwise.
Errors
   2201  Invalid object.

SetVariableValue

Method: SetVariableValue(strName as string, strValue as string)

Return Value
Sets the value (second parameter) of the variable submitted in the first parameter.

Description
SetVariableValue sets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors
   2201  Invalid object.
   2202  No context.
   2204  No such variable in scope
   2205  Variable cannot be evaluated
   2206  Variable returns sequence
   2207  Variable read-only
   2208  No modification allowed
3.2.5 **AuthenticRange**

**See also**

The first table lists the properties and methods of *AuthenticRange* that can be used to navigate through the document and select specific portions.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Clone</td>
</tr>
<tr>
<td>FirstTextPosition</td>
<td>MoveBegin</td>
</tr>
<tr>
<td>FirstXMLData</td>
<td>MoveEnd</td>
</tr>
<tr>
<td>FirstXMLDataOffset</td>
<td>NextCursorPosition</td>
</tr>
<tr>
<td>LastTextPosition</td>
<td>PreviousCursorPosition</td>
</tr>
<tr>
<td>LastXMLData</td>
<td>Goto</td>
</tr>
<tr>
<td>LastXMLDataOffset</td>
<td>GotoNext</td>
</tr>
<tr>
<td>Parent</td>
<td>GotoPrevious</td>
</tr>
<tr>
<td></td>
<td>IsEmpty</td>
</tr>
<tr>
<td></td>
<td>ExpandTo</td>
</tr>
<tr>
<td></td>
<td>Select</td>
</tr>
<tr>
<td></td>
<td>isEmpty</td>
</tr>
<tr>
<td></td>
<td>GotoPrevious</td>
</tr>
<tr>
<td></td>
<td>SelectPrevious</td>
</tr>
<tr>
<td></td>
<td>SetFromRange</td>
</tr>
<tr>
<td></td>
<td>IsEqual</td>
</tr>
</tbody>
</table>

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Edit operations</th>
<th>Dynamic table operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Copy</td>
<td>AppendRow</td>
</tr>
<tr>
<td></td>
<td>Cut</td>
<td>DeleteRow</td>
</tr>
<tr>
<td></td>
<td>Delete</td>
<td>DuplicateRow</td>
</tr>
<tr>
<td></td>
<td>IsCopyEnabled</td>
<td>InsertRow</td>
</tr>
<tr>
<td></td>
<td>IsCutEnabled</td>
<td>IsFirstChild</td>
</tr>
<tr>
<td></td>
<td>IsDeleteEnabled</td>
<td>IsInDynamicTable</td>
</tr>
<tr>
<td></td>
<td>IsPasteEnabled</td>
<td>IsLastRow</td>
</tr>
<tr>
<td></td>
<td>Paste</td>
<td>MoveRowDown</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoveRowUp</td>
</tr>
</tbody>
</table>

The following methods provide the functionality of the Authentic entry helper windows for range objects.

**Operations of the entry helper windows**

<table>
<thead>
<tr>
<th>Elements</th>
<th>Attributes</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanPerformActionWith</td>
<td>GetElementAttributeValue</td>
<td>GetEntityNames</td>
</tr>
<tr>
<td>CanPerformAction</td>
<td>GetElementAttributeNames</td>
<td>InsertEntity</td>
</tr>
<tr>
<td>PerformAction</td>
<td>GetElementHierarchy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HasElementAttribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IsTextStateApplied</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetElementAttributeValue</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

*AuthenticRange* objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for *AuthenticRange* objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of *AuthenticRange*...
objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use `AuthenticView.Selection`, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial ranges are accessible via the read-only properties `AuthenticView.DocumentBegin`, `AuthenticView.DocumentEnd`, and `AuthenticView.WholeDocument`. The most flexible method is `AuthenticView.Goto`, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above, with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see `FirstTextPosition` and `LastTextPosition`. This method requires complex internal calculations and should be used with care.
- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see `FirstXMLData`, `FirstXMLDataOffset`, `LastXMLData`, and `LastXMLDataOffset`. This method is very efficient but requires knowledge on the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The **Text** property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions `Cut`, `Copy`, `Paste` and `Delete`.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the **XMLData** objects of the underlying document to modify them directly.

**AppendRow**

**See also**

**Method:** `AppendRow()` as Boolean

**Description**

If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns `true` if the append operation was successful, otherwise `false`.

**Errors**

- 2001 The authentic range object or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**Examples**

```
' ---------------------------------------------------------
' Scripting environment - VBScript
' Append row at end of current dynamically growable table
```
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
    objRange.AppendRow
    ' objRange points to beginning of new row
    objRange.Select
End If

Application

See also

Property: Application as Application (read-only)

Description
Accesses the Authentic Desktop application object.

Errors
   2001   The authentic range object or its related view object is no longer valid.
   2005   Invalid address for the return parameter was specified.

CanPerformAction

See also

Method: CanPerformAction (eAction as SPYAuthenticActions, strElementName as String) as Boolean

Description
CanPerformAction and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns True, otherwise it returns False.

HINT: To find out all valid element names for a given action, use CanPerformActionWith.

Errors
   2001   The authentic range object or its related view object is no longer valid.
   2005   Invalid address for the return parameter was specified.
   2007   Invalid action was specified.

Examples
See PerformAction.

CanPerformActionWith

See also

Method: CanPerformActionWith (eAction as SPYAuthenticActions,
**out_arrElementNames as Variant**)

**Description**
`PerformActionWith` and its related methods, enable access to the entry-helper functions of Authentic. These function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use `CanPerformActionWith`.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.
- 2007 Invalid action was specified.

**Examples**
See `PerformAction`.

**Clone**

See also

**Method:** `Clone()` as `AuthenticRange`

**Description**
Returns a copy of the range object.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**CollapsToBegin**

See also

**Method:** `CollapsToBegin()` as `AuthenticRange`

**Description**
Sets the end of the range object to its begin. The method returns the modified range object.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**CollapsToEnd**

See also

**Method:** `CollapsToEnd()` as `AuthenticRange`
Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Copy
See also

Method: Copy() as Boolean

Description
Returns False if the range contains no portions of the document that may be copied.
Returns True if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/paste buffer.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Cut
See also

Method: Cut() as Boolean

Description
Returns False if the range contains portions of the document that may not be deleted.
Returns True after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/paste buffer.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Delete
See also

Method: Delete() as Boolean

Description
Returns False if the range contains portions of the document that may not be deleted.
Returns True after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
DeleteRow

See also

Method: DeleteRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns true, if the delete operation was successful, otherwise false.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

' ---------------------------------------------------------
' Scripting environment - VBScrip
' Delete selected row from dynamically growing table
' ---------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we are in a table
If objRange.IsInDynamicTable Then
   objRange.DeleteRow
End If

DuplicateRow

See also

Method: DuplicateRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns true if the duplicate operation was successful, otherwise false.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

' ---------------------------------------------------------
' Scripting environment - VBScrip
' duplicate row in current dynamically growable table
' ---------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection
' check if we can insert something
If objRange.IsInDynamicTable Then
    objRange.DuplicateRow
    ' objRange points to beginning of new row
    objRange.Select
End If

EvaluateXPath

*Method:* `EvaluateXPath` (strExpression as string) strValue as string

**Return Value**
The method returns a string

**Description**
`EvaluateXPath()` executes an XPath expressions with the context node being the beginning of the range selection. The result is returned as string, in the case of a sequence it is a space-separated string. If XML context node is irrelevant, the user may provide any node, like `AuthenticView/XMLDataRoot`.

**Errors**
- 2001 Invalid object
- 2005 Invalid parameter
- 2008 Internal error
- 2202 Missing context node
- 2211 XPath error

ExpandTo

*See also*

*Method:* `ExpandTo` *(eKind as SPYAuthenticElementKind), as AuthenticRange*

**Description**
Selects the whole element of type `eKind`, that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2003 Range expansion would be beyond end of document.
- 2005 Invalid address for the return parameter was specified.

FirstTextPosition

*See also*

*Property:* `FirstTextPosition` as Long

**Description**
Set or get the left-most text position index of the range object. This index is always less or equal to `LastTextPosition`. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect.
as the cursor-left key.

If you set FirstTextPosition to a value greater than the current LastTextPosition, LastTextPosition gets set to the new FirstTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
    MsgBox "Test using direct text cursor positioning was ok"
Else
    MsgBox "Ooops!"
End If

FirstXMLData

See also

Property: FirstXMLData as XMLData

Description

Set or get the first XMLData element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the FirstXMLDataOffset attribute.

Whenever you set FirstXMLData to a new data object, FirstXMLDataOffset gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set FirstXMLData/FirstXMLDataOffset selects a position greater then the current LastXMLData/LastXMLDataOffset, the latter gets moved to the new start position.
HINT: You can use the **FirstXMLData** and **LastXMLData** properties, to directly access and manipulate the underlying XML document in those cases where the methods available with the **AuthenticRange** object are not sufficient.

**Errors**

- **2001** The authentic range object, or its related view object is not valid.
- **2005** Invalid address for the return parameter was specified.
- **2008** Internal error
- **2009** The XMLData object cannot be accessed.

**Examples**

```
' Scripting environment - VBScript
' show name of currently selected XMLData element
' -----------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objXmlData
Set objXmlData = objAuthenticView.Selection.FirstXMLData
' authentic view adds a 'text' child element to elements
' of the document which have content. So we have to go one
' element up.
Set objXmlData = objXmlData.Parent
MsgBox "Current selection selects element " & objXmlData.Name
```

**FirstXMLDataOffset**

**See also**

**Property:** **FirstXMLDataOffset** as Long

**Description**

Set or get the cursor position offset inside **FirstXMLData** element for the beginning of the range. Offset positions are based on the characters returned by the **Text** property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element.

The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If **FirstXMLData/FirstXMLDataOffset** selects a position after the current **LastXMLData/LastXMLDataOffset**, the latter gets moved to the new start position.

**Errors**

- **2001** The authentic range object, or its related view object is not valid.
- **2005** Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

', '---------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
', '---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0 ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1 ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
    objRange.Select()
Else
    MsgBox "Oops"
End If

GetElementAttributeNames

See also

Method: GetElementAttributeNames (strElementName as String,
out_arrAttributeNames as Variant)

Description
Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods
GetElementAttributeValue and SetElementAttributeValue.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
    Invalid address for the return parameter was specified.

Examples
See GetElementAttributeValue.
GetElementAttributeValue

See also

Method: GetElementAttributeValue (strElementName as String, strAttributeName as String) as String

Description
Retrieve the value of the attribute specified in strAttributeName, for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetElementHierarchy

See also

Method: GetElementHierarchy (out_arrElementNames as Variant)

Description
Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single XMLData element.

The names of the element hierarchy, together with the range object uniquely identify XMLData elements in the document. The attributes of these elements can be directly accessed by GetElementAttributeNames, and related methods.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

GetEntityNames

See also

Method: GetEntityNames (out_arrEntityNames as Variant)

Description
Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the InsertEntity function.
Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
See: GetElementHierarchy and InsertEntity.

GetVariableValue
Method: GetVariableValue(strName as string) strVal as string

Return Value
Gets the value of the variable named as the method's parameter.

Errors
2001 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

Goto
See also

Method: Goto(eKind as SPYAuthenticElementKind, nCount as Long, eFrom as SPYAuthenticDocumentPosition) as AuthenticRange

Description
Sets the range to point to the beginning of the nCount element of type eKind. The start position is defined by the parameter eFrom.

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
2006 Invalid start position specified.
2009 Invalid address for the return parameter was specified.

GotoNext
See also

Method: GotoNext(eKind as SPYAuthenticElementKind) as AuthenticRange
Description
Sets the range to the beginning of the next element of type eKind. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples
' --------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' --------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False
On Error Resume Next
While Not bEndOfDocument
    objRange.GotoNext(spyAuthenticWord).Select
    If ((Err.number - vbObjecterror) = 2003) Then
        bEndOfDocument = True
        Err.Clear
    ElseIf (Err.number <> 0) Then
        Err.Raise ' forward error
    End If
End If
Wend

GotoNextCursorPosition

See also

Method: GotoNextCursorPosition() as AuthenticRange

Description
Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid address for the return parameter was specified.
GotoPrevious

See also

*Method: GotoPrevious (eKind as **SPYAuthenticElementKind**) as **AuthenticRange***

**Description**
Sets the range to the beginning of the element of type eKind which is before the beginning of the current range. The method returns the modified range object.

**Errors**
- 2001: The authentic range object, or its related view object is no longer valid.
- 2005: Invalid element kind specified.

**Examples**
```
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bBeginOfDocument
bBeginOfDocument = False
On Error Resume Next
While Not bBeginOfDocument
    objRange.GotoPrevious(spyAuthenticTag).Select
    If ((Err.number - vbObjecterror) = 2004) Then
        bBeginOfDocument = True
        Err.Clear
    ElseIf (Err.number <> 0) Then
        Err.Raise ' forward error
    End If
End If
Wend
```

GotoPreviousCursorPosition

See also

*Method: GotoPreviousCursorPosition() as **AuthenticRange***

**Description**
Set the range to the cursor position immediately before the current position. Returns the modified object.

**Errors**
- 2001: The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid address for the return parameter was specified.

**HasElementAttribute**

See also

*Method:* HasElementAttribute *(strElementName as String, strAttributeName as String) as Boolean*

**Description**
Tests if the enclosing element with name *strElementName*, supports the attribute specified in *strAttributeName*.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid element name was specified.
  Invalid address for the return parameter was specified.

**InsertEntity**

See also

*Method:* InsertEntity *(strEntityName as String)*

**Description**
Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by *GetEntityNames*.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Unknown entry name was specified.

**Examples**

```
' ---------------------------------------------------------
' Scripting environment - VBScript
' Insert the first entity in the list of available entities
' ---------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we get the names of all available entities as they
' are shown in the entry helper of XMLSpy
Dim arrEntities
objRange.GetEntityNames arrEntities

' we insert the first one of the list
If UBound(arrEntities) >= 0 Then
    objRange.InsertEntity arrEntities(0)
Else
    MsgBox "Sorry, no entities are available for this document"
End If
```
**InsertRow**

See also

*Method:* `InsertRow()` as Boolean

**Description**

If the beginning of the range is inside a dynamic table, this method inserts a new row before the current one. The selection of the range, gets modified to point to the beginning of the newly inserted row. The function returns `true` if the insert operation was successful, otherwise `false`.

**Errors**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>The authentic range object, or its related view object is no longer valid.</td>
</tr>
<tr>
<td>2005</td>
<td>Invalid address for the return parameter was specified.</td>
</tr>
</tbody>
</table>

**Examples**

```
' Scripting environment - VBScript
' Insert row at beginning of current dynamically growing table
' ---------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
    objRange.InsertRow
    ' objRange points to beginning of new row
    objRange.Select
End If
```

**IsCopyEnabled**

See also

*Property:* `IsCopyEnabled` as Boolean (read-only)

**Description**

Checks if the copy operation is supported for this range.

**Errors**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>The authentic range object, or its related view object is no longer valid.</td>
</tr>
<tr>
<td>2005</td>
<td>Invalid address for the return parameter was specified.</td>
</tr>
</tbody>
</table>

**IsCutEnabled**

See also

*Property:* `IsCutEnabled` as Boolean (read-only)

**Description**

Checks if the cut operation is supported for this range.
Errors
  2001  The authentic range object, or its related view object is no longer valid.
  2005  Invalid address for the return parameter was specified.

IsDeleteEnabled

See also

Property: IsDeleteEnabled as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors
  2001  The authentic range object, or its related view object is no longer valid.
  2005  Invalid address for the return parameter was specified.

IsEmpty

See also

Method: IsEmpty() as Boolean

Description
Tests if the first and last position of the range are equal.

Errors
  2001  The authentic range object, or its related view object is no longer valid.
  2005  Invalid address for the return parameter was specified.

IsEqual

See also

Method: IsEqual(objCmpRange as AuthenticRange) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors
  2001  One of the two range objects being compared, is invalid.
  2005  Invalid address for a return parameter was specified.

IsFirstRow

See also

Property: IsFirstRow as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more
information.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**IsInDynamicTable**

See also

**Method:** `IsInDynamicTable()` as Boolean

**Description**
Test if the whole range is inside a table that supports the different row operations like 'insert', 'append', duplicate, etc.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**IsLastRow**

See also

**Property:** `IsLastRow` as Boolean (read-only)

**Description**
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**IsPasteEnabled**

See also

**Property:** `IsPasteEnabled` as Boolean (read-only)

**Description**
Checks if the paste operation is supported for this range.

**Errors**
- 2001 The authentic range object, or its related view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.
IsSelected

Property: IsSelected as Boolean

Description
Returns true() if selection is present. The selection range still can be empty: that happens when e.g. only the cursor is set.

IsTextStateApplied

See also

Method: IsTextStateApplied (i_strElementName as String) as Boolean

Description
Checks if all the selected text is embedded into an XML Element with name i_strElementName. Common examples for the parameter i_strElementName are "strong", "bold" or "italic".

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

LastTextPosition

See also

Property: LastTextPosition as Long

Description
Set or get the rightmost text position index of the range object. This index is always greater or equal to FirstTextPosition. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

If you set LastTextPosition to a value less then the current FirstTextPosition, FirstTextPosition gets set to the new LastTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples
' ---------------------------------------
' Scripting environment - VBScript
' -------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
    MsgBox "Test using direct text cursor positioning was ok"
Else
    MsgBox "Oops!"
End If

LastXMLData
See also

Property: LastXMLData as XMLData

Description
Set or get the last XMLData element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the LastXMLDataOffset attribute.

Whenever you set LastXMLData to a new data object, LastXMLDataOffset gets set to the last cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set LastXMLData/LastXMLDataOffset, select a position less then the current FirstXMLData/FirstXMLDataOffset, the latter gets moved to the new end position.

HINT: You can use the FirstXMLData and LastXMLData properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the AuthenticRange object are not sufficient.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.
LastXMLDataOffset

See also

Property: LastXMLDataOffset as Long

Description
Set or get the cursor position inside LastXMLData element for the end of the range.

Offset positions are based on the characters returned by the Text property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If LastXMLData / LastXMLDataOffset selects a position before FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples
' ----------------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ----------------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
MoveBegin

See also

Method: MoveBegin (eKind as SYPAuthenticElementKind, nCount as Long) as AuthenticRange

Description
Move the beginning of the range to the beginning of the nCount element of type eKind. Counting starts at the current beginning of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

MoveEnd

See also

Method: MoveEnd (eKind as SYPAuthenticElementKind, nCount as Long) as AuthenticRange

Description
Move the end of the range to the begin of the nCount element of type eKind. Counting starts at the current end of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.
MoveRowDown

See also

Method: MoveRowDown() as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns true if the move operation was successful, otherwise false.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MoveRowUp

See also

Method: MoveRowUp() as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns true if the move operation was successful, otherwise false.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as AuthenticView (read-only)

Description
Access the view that owns this range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Paste

See also

Method: Paste() as Boolean

Description
Returns *False* if the copy/paste buffer is empty, or its content cannot replace the current selection.

Otherwise, deletes the current selection, inserts the content of the copy/paste buffer, and returns *True*.

**Errors**

- **2001**: The authentic range object, or its related view object is no longer valid.
- **2005**: Invalid address for the return parameter was specified.

**PerformAction**

**See also**

**Method**: PerformAction *(eAction as SPYAuthenticActions, strElementName as String) as Boolean*

**Description**

PerformAction and its related methods, give access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without a need to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If no such location can be found, the method returns *False*. Otherwise, the document gets modified and the range points to the beginning of the modification.

HINT: To find out element names that can be passed as the second parameter use **CanPerformActionWith**.

**Errors**

- **2001**: The authentic range object, or its related view object is no longer valid.
- **2005**: Invalid address for the return parameter was specified.
- **2007**: Invalid action was specified.

**Examples**

```
' --------------------------------------------
' Scripting environment - VBScript
' Insert the innermost element
' --------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' we determine the elements that can be inserted at the current position
Dim arrElements() = objRange.CanPerformActionWith spyAuthenticInsertBefore, arrElements

' we insert the first (innermost) element
If UBound(arrElements) >= 0 Then
    objRange.PerformAction spyAuthenticInsertBefore, arrElements(0)
    ' objRange now points to the beginning of the inserted element
    ' we set a default value and position at its end
    objRange.Text = "Hello"
    objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
```
MsgBox "Can't insert any elements at current position"

End If

Select

See also

Method: Select()

Description
Makes this range the current user interface selection. You can achieve the same result using:
' objRange.Parent.Selection = objRange

Errors
2001   The authentic range object or its related view object is no longer valid.

Examples
' -----------------------------
' Scripting environment - VBScript
' -----------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' set current selection to end of document
objAuthenticView.DocumentEnd.Select()

SelectNext

See also

Method: SelectNext (eKind as SPYAuthenticElementKind) as AuthenticRange

Description
Selects the element of type eKind after the current end of the range. The method returns the
modified range object.

Errors
2001   The authentic range object, or its related view object is no longer valid.
2003   Target lies after end of document.
2005   Invalid element kind specified.
       Invalid address for the return parameter was specified.

Examples
' -----------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' -----------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
    objRange.SelectNext(spyAuthenticWord).Select
    If ((Err.number - vbObjectError) = 2003) Then
        bEndOfDocument = True
        Err.Clear
    ElseIf (Err.number <> 0) Then
        Err.Raise ' forward error
    End If
Wend

SelectPrevious

See also

Method: GotoPrevious (eKind as SPYAuthenticElementKind) as AuthenticRange

Description
Selects the element of type eKind before the current beginning of the range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples
' --------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document tag-by-tag
' --------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bBeginOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
    objRange.SelectPrevious(spyAuthenticTag).Select
    If ((Err.number - vbObjectError) = 2004) Then
        bBeginOfDocument = True
        Err.Clear
    ElseIf (Err.number <> 0) Then
        Err.Raise ' forward error
    End If
SetElementAttributeValue

See also

**Method:** `SetElementAttributeValue` *(strElementName as String, strAttributeName as String, strAttributeValue as String)*

**Description**
Set the value of the attribute specified in `strAttributeName` for the element identified with `strElementName`. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use `GetElementAttributeNames`, or `HasElementAttribute`.

**Errors**
- 2001 The authentic range object or its related view object is no longer valid.
- 2005 Invalid element name was specified.
- Invalid attribute name was specified.
- Invalid attribute value was specified.

**Examples**
```
' ---------------------------------------------------------------
' Scripting environment - VBScript
' Get and set element attributes
' ---------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
  If UBound(arrElements) >= 0 Then
    ' we use the top level element and find out its valid attributes
    Dim arrAttrs()
    objRange.GetElementAttributeNames arrElements(0), arrAttrs

    If UBound(arrAttrs) >= 0 Then
      ' we retrieve the current value of the first valid attribute
      Dim strAttrVal
      strAttrVal = objRange.GetElementAttributeValue(arrElements(0), arrAttrs(0))
      msgbox "current value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " & strAttrVal

      ' we change this value and read it again
      strAttrVal = "Hello"
      objRange.SetElementAttributeValue arrElements(0), arrAttrs(0), strAttrVal
      strAttrVal = objRange.GetElementAttributeValue
```
(arrElements(0), arrAttrs(0))
    msgbox "new value of " & arrElements(0) & "//" & arrAttrs(0)
& " is: " & strAttrVal
  End If
End If
End If

SetFromRange

See also

**Method:** SetFromRange (objSrcRange as AuthenticRange)

**Description**
Sets the range object to the same beginning and end positions as objSrcRange.

**Errors**
- 2001 One of the two range objects, is invalid.
- 2005 Null object was specified as source object.

SetVariableValue

**Method:** SetVariableValue (strName as string, strValue as string)

**Return Value**
Sets the value (second parameter) of the variable named in the first parameter.

**Errors**
- 2201 Invalid object.
- 2202 No context.
- 2204 No such variable in scope
- 2205 Variable cannot be evaluated
- 2206 Variable returns sequence
- 2207 Variable read-only
- 2208 No modification allowed

Text

See also

**Property:** Text as String

**Description**
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio
buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more than one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&amp;'.

Setting the text to the empty string, does not delete any XML elements. Use **Cut**, **Delete** or **PerformAction** instead.

**Errors**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>The authentic range object or its related view object is no longer valid.</td>
</tr>
<tr>
<td>2005</td>
<td>Invalid address for a return parameter was specified.</td>
</tr>
</tbody>
</table>
3.2.6 **AuthenticView**

**See also**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Goto</td>
<td>OnBeforeCopy</td>
</tr>
<tr>
<td>AsXMLString</td>
<td>IsRedoEnabled</td>
<td>OnBeforeCut</td>
</tr>
<tr>
<td>DocumentBegin</td>
<td>IsUndoEnabled</td>
<td>OnBeforeDelete</td>
</tr>
<tr>
<td>DocumentEnd</td>
<td>Print</td>
<td>OnBeforeDrop</td>
</tr>
<tr>
<td>Event</td>
<td>Redo</td>
<td>OnBeforePaste</td>
</tr>
<tr>
<td>MarkupVisibility</td>
<td>Undo</td>
<td>OnDragOver</td>
</tr>
<tr>
<td>Parent</td>
<td>UpdateXMLInstanceEntities</td>
<td>OnKeyBoardEvent</td>
</tr>
<tr>
<td>Selection</td>
<td></td>
<td>OnMouseEvent</td>
</tr>
<tr>
<td>XMLDataRoot</td>
<td></td>
<td>OnMouseMove</td>
</tr>
<tr>
<td>WholeDocument</td>
<td></td>
<td>OnSelectionChanged</td>
</tr>
</tbody>
</table>

**Description**

AuthenticView and its child objects [AuthenticRange](#) and [AuthenticDataTransfer](#) provide you with an interface for Authentic View, which allow easy and consistent modification of document contents. These interfaces replace the following interfaces which are marked now as **obsolete**:

- OldAuthenticView *(old name was DocEditView)*
- AuthenticSelection *(old name was DocEditSelection, superseded by AuthenticRange)*
- AuthenticEvent *(old name was DocEditEvent)*

**AuthenticView** gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

**AuthenticView** uses objects of type [AuthenticRange](#) to make navigation inside the document straightforward, and to allow for the flexible selection of logical text elements. Use the properties `DocumentBegin`, `DocumentEnd`, or `WholeDocument` for simple selections, while using the `Goto` method for more complex selections. To navigate relative to a given document range, see the methods and properties of the [AuthenticRange](#) object.

**Events**

**OnBeforeCopy**

**See also**

**Event:** `OnBeforeCopy()` as Boolean

**Scripting environment - VBS**

```vbs
Function On_AuthenticBeforeCopy()
    ' On_AuthenticBeforeCopy = False ' to disable operation
End Function
```

**Scripting environment - JScript**

```javascript
function On_AuthenticBeforeCopy()
{
```
IDE Plugin:
IXMLSpyPlugIn.OnEvent(21, ...) // nEventId = 21

Description
This event gets triggered before a copy operation gets performed on the document. Return True (or nothing) to allow copy operation. Return False to disable copying.

OnBeforeCut
See also

Event: OnBeforeCut() as Boolean

Scripting environment - VBS:
Function On_AuthenticBeforeCut()
  ' On_AuthenticBeforeCut = False ' to disable operation
End Function

Scripting environment - JScript:
function On_AuthenticBeforeCut()
{
  // return false; /* to disable operation */
}

IDE Plugin:
IXMLSpyPlugIn.OnEvent(20, ...) // nEventId = 20

Description
This event gets triggered before a cut operation gets performed on the document. Return True (or nothing) to allow cut operation. Return False to disable operation.

OnBeforeDelete
See also

Event: OnBeforeDelete() as Boolean

Scripting environment - VBS:
Function On_AuthenticBeforeDelete()
  ' On_AuthenticBeforeDelete = False ' to disable operation
End Function

Scripting environment - JScript:
function On_AuthenticBeforeDelete()
{
  // return false; /* to disable operation */
}

IDE Plugin:
Description
This event gets triggered before a delete operation gets performed on the document. Return True (or nothing) to allow delete operation. Return False to disable operation.

OnBeforeDrop
See also

Event: OnBeforeDrop (i_nXPos as Long, i_nYPos as Long, i_ipRange as AuthenticRange, i_ipData as cancelBoolean)

Scripting environment - VBScript:
Function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
    ' On_AuthenticBeforeDrop = False  ' to disable operation
End Function

Scripting environment - JScript:
function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
{
    // return false;  /* to disable operation */
}

IDE Plugin:
IXMLSpyPlugIn.OnEvent (11, ...) // nEvent Id = 11

Description
This event gets triggered whenever a previously dragged object gets dropped inside the application window. All event related information gets passed as parameters.

The first two parameters specify the mouse position at the time when the event occurred. The parameter objRange passes a range object that selects the XML element below the mouse position. The value of this parameter might be NULL. Be sure to check before you access the range object. The parameter objData allows to access information about the object being dragged.

Return False to cancel the drop operation. Return True (or nothing) to continue normal operation.

OnBeforePaste
See also

Event: OnBeforePaste (objData as Variant, strType as String) as Boolean

Scripting environment - VBScript:
Function On_AuthenticBeforePaste(objData, strType)
    ' On_AuthenticBeforePaste = False  ' to disable operation
End Function

Scripting environment - JScript:
function On_AuthenticBeforePaste(objData, strType)
IDE Plugin:
IXMLSpyPlugIn.OnEvent(19, ...) // nEventId = 19

Description
This event gets triggered before a paste operation gets performed on the document. The parameter strType is one of "TEXT", "UNICODETEXT" or "IUNKNOWN". In the first two cases objData contains a string representation of the object that will be pasted. In the later case, objData contains a pointer to an IUnknown COM interface.

Return True (or nothing) to allow paste operation. Return False to disable operation.

OnBeforeSave
Event: OnBeforeSave (SaveAs flag) as Boolean

Description: OnBeforeSave gives the opportunity to e.g. warn the user about overwriting the existing XML document, or to make the document read-only when specific circumstances are not met. The event will be fired before the file dialog is shown.

OnDragOver
See also

Event: OnDragOver (nXPos as Long, nYPos as Long, eMouseEvent as SPYMouseEvent, objRange as AuthenticRange, objData as AuthenticDataTransfer) as Boolean

Scripting environment - VBScript:
Function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
    ' On_AuthenticDragOver = False ' to disable operation
End Function

Scripting environment - JScript:
function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
{
    // return false; /* to disable operation */
}

IDE Plugin:
IXMLSpyPlugIn.OnEvent(10, ...) // nEventId = 10

Description
This event gets triggered whenever an object from within or outside of Authentic View gets dragged with the mouse over the application window. All event related information gets passed as parameters.
The first three parameters specify the mouse position, the mouse button status and the status of the virtual keys at the time when the event occurred. The parameter \textit{objRange} passes a range object that selects the XML element below the mouse position. The value of this parameter might be \texttt{NULL}. Be sure to check before you access the range object. The parameter \textit{objData} allows to access information about the object being dragged.

Return \texttt{False} to cancel the drag operation. Return \texttt{True} (or nothing) to continue normal operation.

\textbf{OnKeyboardEvent}

\textbf{See also}

\textbf{Event:} \texttt{OnKeyboardEvent (eKeyEvent as SPYKeyEvent, nKeyCode as Long, nVirtualKeyStatus as Long) as Boolean}

\textbf{Scripting environment - VBScript:}

Function \texttt{On\_AuthenticKeyboardEvent (eKeyEvent, nKeyCode, nVirtualKeyStatus)}

\hspace{1cm} ' On\_AuthenticKeyboardEvent = True ' to cancel bubbling of event

End Function

\textbf{Scripting environment - JScript:}

function \texttt{On\_AuthenticKeyboardEvent (eKeyEvent, nKeyCode, nVirtualKeyStatus)}

\hspace{1cm} \{ // return true; /* to cancel bubbling of event */ \}

\textbf{IDE Plugin:}

\texttt{IXMLSpyPlugIn.OnEvent (30, ...)} // \texttt{nEventId = 30}

\textbf{Description}

This event gets triggered for \texttt{WM\_KEYDOWN}, \texttt{WM\_KEYUP} and \texttt{WM\_CHAR} Windows messages.

The actual message type is available in the \texttt{eKeyEvent} parameter. The status of virtual keys is combined in the parameter \texttt{nVirtualKeyStatus}. Use the bit-masks defined in the enumeration datatype \texttt{SPYVirtualKeyMask}, to test for the different keys or their combinations.

\textbf{OnLoad}

\textbf{Event:} \texttt{OnLoad ()}

\textbf{Description:} \texttt{OnLoad} can be used e.g. to restrict some AuthenticView functionality, as shown in the example below:

\texttt{function On\_AuthenticLoad( )}

\hspace{1cm} \{ // We are disabling all entry helpers in order to prevent user from manipulating XML tree

\hspace{2cm} AuthenticView.DisableElementEntryHelper();

\hspace{2cm} AuthenticView.DisableAttributeEntryHelper();

\hspace{1cm} \}


// We are also disabling the markup buttons for the same purpose
AuthenticationView.SetToolBarButtonState( 'AuthenticMarkupSmall',
authenticToolBarButtonDisabled );
AuthenticationView.SetToolBarButtonState( 'AuthenticMarkupLarge',
authenticToolBarButtonDisabled );
AuthenticationView.SetToolBarButtonState( 'AuthenticMarkupMixed',
authenticToolBarButtonDisabled );
}

In the example the status of the Markup Small, Markup Large, Markup Mixed toolbar buttons are manipulated with the help of button identifiers. See complete list.

**OnMouseEvent**

See also

**Event:** OnMouseEvent (nXPos as Long, nYPos as Long, eMouseEvent as SPYMouseEvent,
objRange as AuthenticRange) as Boolean

**Scripting environment - VBScript:**

Function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
    ' On AuthenticMouseEvent = True ' to cancel bubbling of event
End Function

**Scripting environment - JScript:**

function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
{
    // return true; /* to cancel bubbling of event */
}

**IDE Plugin:**

IXMLSpyPlugIn.OnEvent (31, ... ) // nEventId = 31

**Description**

This event gets triggered for every mouse movement and mouse button Windows message.

The actual message type and the mouse buttons status, is available in the eMouseEvent parameter. Use the bit-masks defined in the enumeration datatype SPYMouseEvent to test for the different messages, button status, and their combinations.

The parameter objRange identifies the part of the document found at the current mouse cursor position. The range objects always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no selectable part of the document is found at the current position, the range object is null.

**OnSelectionChanged**

See also

**Event:** OnSelectionChanged (objNewSelection as AuthenticRange)

**Scripting environment - VBScript:**

Function On_AuthenticSelectionChanged (objNewSelection)
End Function

**Scripting environment - JScript:**

```javascript
function On_AuthenticSelectionChanged (objNewSelection)
{
}
```

**IDE Plugin:**

```javascript
IXMLSpyPlugIn.OnEvent (23, ...) // nEventId = 23
```

**Description**

This event gets triggered whenever the selection in the user interface changes.

**OnToolbarButtonClicked**

**Event:** `OnToolbarButtonClicked` (Button identifier)

**Description:** `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter `button identifier` helps to determine which button was clicked. The list of predefined button identifiers is below:

- AuthenticPrint
- AuthenticPrintPreview
- AuthenticUndo
- AuthenticRedo
- AuthenticCut
- AuthenticCopy
- AuthenticPaste
- AuthenticClear
- AuthenticMarkupHide
- AuthenticMarkupLarge
- AuthenticMarkupMixed
- AuthenticMarkupSmall
- AuthenticValidate
- AuthenticChangeWorkingDBXMLCell
- AuthenticSave
- AuthenticSaveAs
- AuthenticReload
- AuthenticTableInsertRow
- AuthenticTableAppendRow
- AuthenticTableDeleteRow
- AuthenticTableInsertCol
- AuthenticTableAppendCol
- AuthenticTableDeleteCol
- AuthenticTableJoinCellRight
- AuthenticTableJoinCellLeft
- AuthenticTableJoinCellAbove
- AuthenticTableJoinCellBelow
- AuthenticTableSplitCellHorizontally
- AuthenticTableSplitCellVertically
- AuthenticTableAlignCellContentTop
For custom buttons the user might add his own identifiers. Please, note that the user must take care, as the identifiers are not checked for uniqueness. The same identifiers can be used to identify buttons in the Set/GetToolbarState() COM API calls. By adding code for different buttons, the user is in the position to completely redefine the AuthenticView toolbar behavior, adding own methods for table manipulation, etc.

**OnToolbarButtonExecuted**

**Event:** OnToolbarButtonExecuted (Button identifier)

**Description:** OnToolbarButtonClicked is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. See the list of predefined button identifiers.

OnToolbarButtonExecuted is fired after the toolbar action was executed. It is useful e.g. to add update code, as shown in the example below:

```javascript
//event fired when a toolbar button action was executed
function On_AuthenticToolbarButtonExecuted( varBtnIdentifier )
{
    // After whatever command user has executed - make sure to update toolbar button states
    UpdateOwnToolbarButtonStates();
}
```

In this case UpdateOwnToolbarButtonStates is a user function defined in the Global Declarations.

**OnUserAddedXMLNode**

**Event:** OnUserAddedXMLNode (XML node)

**Description:** OnUserAddedXMLNode will be fired when the user adds an XML node as a primary action. This happens in the situations, where the user clicks on
auto-add hyperlinks (see example OnUserAddedXMLNode.sps)

- the Insert..., Insert After..., Insert Before... context menu items
- Append row, Insert row toolbar buttons
- Insert After..., Insert Before... actions in element entry helper (outside StyleVision)

The event doesn’t get fired on Duplicate row, or when the node was added externally (e.g. via COM API), or on Apply (e.g. Text State Icons), or when in XML table operations or in DB operations.

The event parameter is the XML node object, which was added giving the user an opportunity to manipulate the XML node added. An elaborate example for an event handler can be found in the OnUserAddedXMLNode.sps file.

**Application**

See also

**Property:** Application as Application (read-only)

**Description**

Accesses the Authentic Desktop application object.

**Errors**

- 2000  The authentic view object is no longer valid.
- 2005  Invalid address for the return parameter was specified.

**AsXMLString**

See also

**Property:** AsXMLString as String

**Description**

Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new XMLString does not match the actual schema file error 2011 gets returned.

**Errors**

- 2000  The authentic view object is no longer valid.
- 2011  AsXMLString was set to a value which is no valid XML for the current schema file.

**ContextMenu**

**Property:** ContextMenu() as ContextMenu

**Description**

The property ContextMenu gives access to customize the context menu. The best place to do it is in the event handler OnContextMenuActivated.

**Errors**
CreateXMLNode

Method: CreateXMLNode(\textit{nKind} as \texttt{SPYXMLDataKind}) as \texttt{XMLData}

Return Value
The method returns the new \texttt{XMLData} object.

Description
To create a new \texttt{XMLData} object use the \texttt{CreateXMLNode()} method.

Errors
\begin{tabular}{l|l}
2000 & Invalid object. \\
2005 & Invalid parameter. \\
\end{tabular}

DisableAttributeEntryHelper

Method: DisableAttributeEntryHelper()

Description
\texttt{DisableAttributeEntryHelper()} disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
\begin{tabular}{l|l}
2000 & Invalid object. \\
\end{tabular}

DisableElementEntryHelper

Method: DisableElementEntryHelper()

Description
\texttt{DisableElementEntryHelper()} disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
\begin{tabular}{l|l}
2000 & Invalid object. \\
\end{tabular}

DisableEntityEntryHelper

Method: DisableEntityEntryHelper()

Description
\texttt{DisableEntityEntryHelper()} disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
\begin{tabular}{l|l}
2000 & Invalid object. \\
\end{tabular}
**DocumentBegin**

**See also**

**Property:** DocumentBegin as **AuthenticRange** (read-only)

**Description**
Retrieve a range object that points to the beginning of the document.

**Errors**
- 2000 The authentic view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**DocumentEnd**

**See also**

**Property:** DocumentEnd as **AuthenticRange** (read-only)

**Description**
Retrieve a range object that points to the end of the document.

**Errors**
- 2000 The authentic view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**DoNotPerformStandardAction**

**Method:** DoNotPerformStandardAction ()

**Description**
DoNotPerformStandardAction() serves as cancel bubble for macros, and stops further execution after macro has finished.

**Errors**
- 2000 Invalid object.

**EvaluateXPath**

**Method:** EvaluateXPath (XMLData as **XMLData**, strExpression as string) strValue as string

**Return Value**
The method returns a string

**Description**
EvaluateXPath() executes an XPath expressions with the given XML context node. The result is returned as string, in the case of a sequence it is a space-separated string.

**Errors**
- 2000 Invalid object.
- 2005 Invalid parameter.
- 2008 Internal error.
XPath error.

**Event**

See also

*Property: Event as AuthenticEvent (read-only)*)

**Description**

This property gives access to parameters of the last event in the same way as `OldAuthenticView.event` does. Since all events for the scripting environment and external clients are now available with parameters this `Event` property should only be used from within IDE-Plugins.

**Errors**

- 2000 The authentic view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**EventContext**

*Property: EventContext() as EventContext*)

**Description**

`EventContext` property gives access to the running macros context. See the `EventContext` interface description for more details.

**Errors**

- 2000 Invalid object.

**GetToolbarButtonState**

*Method: GetToolbarButtonState(ButtonIdentifier as string) as AuthenticToolbarButtonState*)

**Return Value**

The method returns `AuthenticToolbarButtonState`

**Description**

`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by `AuthenticView`. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

**Errors**
Goto

See also

**Method:** Goto(eKind as SPYAuthenticElementKind, nCount as Long, eFrom as SPYAuthenticDocumentPosition) as AuthenticRange

**Description**
Retrieve a range object that points to the beginning of the nCount element of type eKind. The start position is defined by the parameter eFrom. Use positive values for nCount to navigate to the document end. Use negative values to navigate towards the beginning of the document.

**Errors**
- **2000** The authentic view object is no longer valid.
- **2003** Target lies after end of document.
- **2004** Target lies before beginning of document.
- **2005** Invalid element kind specified. The document position to start from is not one of spyAuthenticDocumentBegin or spyAuthenticDocumentEnd.
- **2006** Invalid address for the return parameter was specified.

**Examples**

```vbscript
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView
On Error Resume Next
Dim objRange
' goto beginning of first table in document
Set objRange = objAuthenticView.Goto(spyAuthenticTable, 1, spyAuthenticDocumentBegin)
If (Err.number = 0) Then
    objRange.Select()
Else
    MsgBox "No table found in document"
End If
```

IsRedoEnabled

See also

**Property:** IsRedoEnabled as Boolean (read-only)

**Description**
True if redo steps are available and **Redo** is possible.
Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsUndoEnabled
See also

Property: IsUndoEnabled as Boolean (read-only)

Description
True if undo steps are available and Undo is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MarkupVisibility
See also

Property: MarkupVisibility as SPYAuthenticMarkupVisibility

Description
Set or get current visibility of markup.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

Parent
See also

Property: Parent as Document (read-only)

Description
Access the document shown in this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Print
See also
**Method:** Print (bWithPreview as Boolean, bPromptUser as Boolean)

**Description**
Print the document shown in this view. If bWithPreview is set to True, the print preview dialog pops up. If bPromptUser is set to True, the print dialog pops up. If both parameters are set to False, the document gets printed without further user interaction.

**Errors**
- 2000 The authentic view object is no longer valid.

**Redo**

**See also**

**Method:** Redo() as Boolean

**Description**
Redo the modification undone by the last undo command.

**Errors**
- 2000 The authentic view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

**Selection**

**See also**

**Property:** Selection as AuthenticRange

**Description**
Set or get current text selection in user interface.

**Errors**
- 2000 The authentic view object is no longer valid.
- 2002 No cursor selection is active.
- 2005 Invalid address for the return parameter was specified.

**Examples**
```vbscript
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' if we are the end of the document, re-start at the beginning
If (objAuthenticView.Selection.EqualTo(objAuthenticView.DocumentEnd)) Then
    objAuthenticView.Selection = objAuthenticView.DocumentBegin
Else
    objAuthenticView.Selection = objAuthenticView.Selection.GotoNextCursorPosition()
    ' or shorter:
    objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```
SetToolbarButtonState

**Method:** SetToolbarButtonState(ButtonIdentifier as string, AuthenticToolbarButtonState state)

**Description**
Get/SetToolbarButtonState queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put SetToolbarButtonState in the OnSelectionChanged event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

**Errors**
- 2000 Invalid object.
- 2008 Internal error.
- 2014 Invalid button identifier.

Undo

See also

**Method:** Undo() as Boolean

**Description**
Undo the last modification of the document from within this view.

**Errors**
- 2000 The authentic view object is no longer valid.
- 2005 Invalid address for the return parameter was specified.

UpdateXMLInstanceEntities

See also

**Method:** UpdateXMLInstanceEntities()

**Description**
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

**Errors**
The method never returns an error.

**Example**
```javascript
// -----------------------------
// Scripting environment - JavaScript
// -----------------------------
{
    var objDocType;
    objDocType = Application.ActiveDocument.DocEditView.XMLRoot.GetFirstChild(10);

    if(objDocType)
    {
        var objEntity = Application.ActiveDocument.CreateChild(14);
        objEntity.Name = "child";
        objEntity.TextValue = "SYSTEM \"child.xml\"";
        objDocType.AppendChild(objEntity);
    }
}

Application.ActiveDocument.AuthenticView.UpdateXMLInstanceEntities();
```

### WholeDocument

**See also**

**Property:** WholeDocument as [AuthenticRange](#) (read-only)

**Description**

Retrieve a range object that selects the whole document.

**Errors**

- **2000**  The authentic view object is no longer valid.
- **2005**  Invalid address for the return parameter was specified.

### XMLDataRoot

**See also**

**Property:** XMLDataRoot as [XMLData](#) (read-only)

**Description**

Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

**Errors**

- **2000**  The authentic view object is no longer valid.
- **2005**  Invalid address for the return parameter was specified.
3.2.7 CodeGeneratorDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Properties and Methods

Standard automation properties
Application
Parent

Programming language selection properties
ProgrammingLanguage
TemplateFileName

Settings for C++ code
CPPSettings_DOMType
CPPSettings_LibraryType
CPPSettings_useMFC
CPPSettings_GenerateVC6ProjectFile
CPPSettings_GenerateVSProjectFile

Settings for C# code
CSharpSettings_ProjectType

Dialog handling for above code generation properties
PropertySheetDialogAction

Output path selection properties
OutputPath
OutputPathDialogAction

Presentation of result
OutputResultDialogAction

Description
Use this object to configure the generation of program code for schema files. The method GenerateProgramCode expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

Application

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

CPPSettings_DOMType

Property: CPPSettings_DOMType as SPYDOMType
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateVC6ProjectFile

Property: CPPSettings_GenerateVC6ProjectFile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateGCCMakefile

Property: CPPSettings_GenerateGCCMakefile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Creates makefiles to compile the generated code under Linux with GCC.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.
### CPPSettings_GenerateVSProjectFile

**Property:** `CPPSettings_GenerateVSProjectFile` as `SPYProjectType`  
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**  
Defines one of the settings that configure generation of C++ code. Only `spyVisualStudio2005Project (=4)` and `spyVisualStudio2008Project (=5)` and `spyVisualStudio2010Project (=6)` are valid project types.

**Errors**  
- **2200** The object is no longer valid.  
- **2201** Invalid action passed as parameter or an invalid address was specified for the return parameter.

### CPPSettings_LibraryType

**Property:** `CPPSettings_LibraryType` as `SPYLibType`  
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**  
Defines one of the settings that configure generation of C++ code.

**Errors**  
- **2200** The object is no longer valid.  
- **2201** Invalid action passed as parameter or an invalid address was specified for the return parameter.

### CPPSettings_UseMFC

**Property:** `CPPSettings_UseMFC` as Boolean  
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**  
Defines one of the settings that configure generation of C++ code.

**Errors**  
- **2200** The object is no longer valid.  
- **2201** Invalid action passed as parameter or an invalid address was specified for the return parameter.

### CSharpSettings_ProjectType

**Property:** `CSharpSettings_ProjectType` as `SPYProjectType`  
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**
Defines the only setting to configure generation of C# code.

**Errors**
- 2200 The object is no longer valid.
- 2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

**OutputPath**

**Property:** OutputPath as String

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**
Selects the base directory for all generated code.

**Errors**
- 2200 The object is no longer valid.
- 2201 Invalid address for the return parameter was specified.

**OutputPathDialogAction**

**Property:** OutputPathDialogAction as SPYDialogAction

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**
Defines how the sub-dialog for selecting the code generation output path gets handled. Set this value to spyDialogUserInput(2) to show the dialog with the current value of the OutputPath property as default. Use spyDialogOK(0) to hide the dialog from the user.

**Errors**
- 2200 The object is no longer valid.
- 2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

**OutputResultDialogAction**

**Property:** OutputResultDialogAction as SPYDialogAction

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**
Defines how the sub-dialog that asks to show the result of the code generation process gets handled. Set this value to spyDialogUserInput(2) to show the dialog. Use spyDialogOK(0) to hide the dialog from the user.

**Errors**
- 2200 The object is no longer valid.
- 2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.
Parent

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
- 2200  The object is no longer valid.
- 2201  Invalid address for the return parameter was specified.

ProgrammingLanguage

Property: ProgrammingLanguage as ProgrammingLanguage
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the output language for the code to be generated.

CAUTION: Setting this property to one of C++, C# or Java, changes the property TemplateFileName to the appropriate template file delivered with Authentic Desktop as well. If you want to generate C++, C# or Java code based on your own templates, set first the programming language and then select your template file.

Errors
- 2200  The object is no longer valid.
- 2201  Invalid address for the return parameter was specified.

PropertySheetDialogAction

Property: PropertySheetDialogAction as SPYDialogAction
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that configures the code generation process gets handled. Set this value to spyDialogUserInput(2) to show the dialog with the current values as defaults. Use spyDialogOK(0) to hide the dialog from the user.

Errors
- 2200  The object is no longer valid.
- 2201  Invalid action passed as parameter or an invalid address was specified for the return parameter.
**TemplateName**

*Property:* `TemplateName` as `String`  
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Description**  
Selects the code generation template file. Authentic Desktop comes with template files for C++, C# or Java in the SPL folder of your installation directory.

Setting this property to one of the code generation template files of your Authentic Desktop installation automatically sets the `ProgrammingLanguage` property to its appropriate value.

**Errors**
- 2200 The object is no longer valid.
- 2201 Invalid address for the return parameter was specified.
### 3.2.8 DatabaseConnection

**See also**

**Properties for import and export**
- File
- ADOConnection
- ODBCConnection

**Properties for import only**
- DatabaseKind
- SQLSelect
- AsAttributes
- ExcludeKeys
- IncludeEmptyElements
- NumberDateTimeFormat
- NullReplacement
- CommentIncluded

**Properties for export only**
- CreateMissingTables
- CreateNew
- TextFieldLen
- DatabaseSchema

**Properties for XML Schema from DB Structure generation**
- PrimaryKeys
- ForeignKeys
- UniqueKeys
- SchemaExtensionType
- SchemaFormat
- ImportColumnsType

**Description**
DatabaseConnection specifies the parameters for the database connection.

Please note that the properties of the DatabaseConnection interface are referring to the settings of the import and export dialogs of Authentic Desktop.

### ADOConnection

**See also**

**Property:** ADOConnection as String

**Description**
The property ADOConnection contains a connection string. Either use this property or ODBCConnection or File to refer to a database.

**Errors**
No error codes are returned.

**Example**
Dim objSpyConn As DatabaseConnection
Set objSpyConn = objSpy.GetDatabaseSettings

Dim objADO As DataLinks
Set objADO = CreateObject("DataLinks")

If Not (objADO Is Nothing) Then
    Dim objConn As Connection
    Set objConn = objADO.PromptNew
    objSpyConn.ADOConnection = objConn.ConnectionString
End If

AsAttributes

See also

Property: AsAttributes as Boolean

Description
Set AsAttributes to true if you want to initialize all import fields to be imported as attributes. Default is false and will initialize all fields to be imported as elements. This property is used only in calls to Application.GetDatabaseImportElementList.

Errors
No error codes are returned.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated XML. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

CreateMissingTables

See also

Property: CreateMissingTables as Boolean

Description
If CreateMissingTables is true, tables which are not already defined in the export database will be created during export. Default is true. This property is used only when exporting to databases.

Errors
No error codes are returned.
CreateNew
See also

Property: CreateNew as Boolean

Description
Set CreateNew true if you want to create a new database on export. Any existing database will be overwritten. See also DatabaseConnection.File. Default is false. This property is used only when exporting to databases.

Errors
No error codes are returned.

DatabaseKind
See also

Property: DatabaseKind as SPYDatabaseKind

Description
Select the kind of database that gets access. The default value is spyDB_Unspecified(7) and is sufficient in most cases. This property is used only when importing from databases.

Errors
No error codes are returned.

DatabaseSchema
See also

Property: DatabaseSchema as String

Description
This property specifies the Schema used for export in Schema aware databases. Default is "". This property is used only when exporting to databases.

Errors
No error codes are returned.

ExcludeKeys
See also

Property: ExcludeKeys as Boolean

Description
Set ExcludeKeys to true if you want to exclude all key columns from the import data. Default is false. This property is used only when importing from databases.

Errors
No error codes are returned.

File
See also

Property: File as String

Description
The property `File` sets the path for the database during export or import. This property can only be used in conjunction with a Microsoft Access database. Either use this property or `ODBCConnection` or `ADOConnection` to refer to the database.

Errors
No error codes are returned.

ForeignKeys
See also

Property: ForeignKeys as Boolean

Description
Specifies whether the Foreign Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

ImportColumnsType
See also

Property: ImportColumnsType as SPYImportColumnsType

Description
Defines if column information from the DB is saved as element or attribute in the XML Schema. Default is as element. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

IncludeEmptyElements
See also

Property: IncludeEmptyElements as Boolean

Description
Set `IncludeEmptyElements` to false if you want to exclude all empty elements. Default is true. This property is used only when importing from databases.
Errors
No error codes are returned.

NullReplacement
See also

Property: NullReplacement as String

Description
This property contains the text value that is used during import for empty elements (null values). Default is "". This property is used only when importing from databases.

Errors
No error codes are returned.

NumberDateTimeFormat
See also

Property: NumberDateTimeFormat as SPYNumberDateTimeFormat

Description
The property NumberDateTimeFormat sets the format of numbers and date- and time-values. Default is spySystemLocale. This property is used only when importing from databases.

Errors
No error codes are returned.

ODBCConnection
See also

Property: ODBCConnection as String

Description
The property ODBCConnection contains a ODBC connection string. Either use this property or ADOConnection or File to refer to a database.

Errors
No error codes are returned.

PrimaryKeys
See also

Property: PrimaryKeys as Boolean

Description
Specifies whether the Primary Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.
Errors
No error codes are returned.

SchemaExtensionType
See also

Property: SchemaExtensionType as SPYSchemaExtensionType

Description
Defines the Schema extension type used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

SchemaFormat
See also

Property: SchemaFormat as SPYSchemaFormat

Description
Defines the Schema format used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

SQLSelect
See also

Property: SQLSelect as String

Description
The SQL query for the import is stored in the property SQLSelect. This property is used only when importing from databases.

Errors
No error codes are returned.

TextFieldLen
See also

Property: TextFieldLen as long

Description
The property TextFieldLen sets the length for created text fields during the export. Default is 255. This property is used only when exporting to databases.
Errors
No error codes are returned.

UniqueKeys

See also

Property: UniqueKeys as Boolean

Description
Specifies whether the Unique Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.
3.2.9 Dialogs

See also

Properties and Methods

Standard automation properties
Application
Parent

Various dialog objects
CodeGenDlg
FileSelectionDlg
SchemaDocumentationDlg
GenerateSampleXMLDlg
DTDSchemaGeneratorDlg
FindInFilesDlg

Description
The Dialogs object provides access to different built-in dialogs of Authentic Desktop. These dialog objects allow to initialize the fields of user dialogs before they get presented to the user or allow to simulate complete user input by your program.

Application

See also

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

CodeGenDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: CodeGeneratorDlg as CodeGeneratorDlg (read-only)

Description
Get a new instance of a code generation dialog object. You will need this object to pass the necessary parameters to the code generation methods. Initial values are taken from last usage of the code generation dialog.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.
FileSelectionDlg

See also

**Property:** FileSelectionDlg as FileSelectionDlg (read-only)

**Description**
Get a new instance of a file selection dialog object.

File selection dialog objects are passed to you with the some events that signal opening or saving of documents and projects.

**Errors**
- 2300 The Dialogs object or one of its parents is no longer valid.
- 2301 Invalid address for the return parameter was specified.

Parent

See also

**Property:** Parent as Application (read-only)

**Description**
Access the Authentic Desktop application object.

**Errors**
- 2300 The object is no longer valid.
- 2301 Invalid address for the return parameter was specified.

SchemaDocumentationDlg

See also

**Property:** SchemaDocumentationDlg as SchemaDocumentationDlg (read-only)

**Description**

**Errors**
- 2300 The Dialogs object or one of its parents is no longer valid.
- 2301 Invalid address for the return parameter was specified.

GenerateSampleXMLDlg

See also

**Property:** GenerateSampleXMLDlg as GenerateSampleXMLDlg (read-only)

**Description**
Get a new instance of a dialog object that parameterizes generation of a sample XML based on a
W3C schema or DTD. See GenerateSampleXML for its usage.

Errors
2300  The Dialogs object or one of its parents is no longer valid.
2301  Invalid address for the return parameter was specified.

DTDSchemaGeneratorDlg
See also

Property: DTDSchemaGeneratorDlg as DTDSchemaGeneratorDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a schema or DTD. See Document.GenerateDTDOrSchemaEx for its usage.

Errors
2300  The Dialogs object or one of its parents is no longer valid.
2301  Invalid address for the return parameter was specified.

FindInFilesDlg
See also

Property: FindInFilesDlg as FindInFilesDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes the search (or replacement) of strings in files. See Application.FindInFiles for its usage.

Errors
2300  The Dialogs object or one of its parents is no longer valid.
2301  Invalid address for the return parameter was specified.

WSDLDocumentationDlg
See also

Property: WSDLDocumentationDlg as WSDLDocumentationDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL documentation. See Document.GenerateWSDLDocumentation for its usage.

Errors
2300  The Dialogs object or one of its parents is no longer valid.
2301  Invalid address for the return parameter was specified.

WSDL20DocumentationDlg
See also

Property: WSDL20DocumentationDlg as WSDL20DocumentationDlg (read-only)
Description
Get a new instance of a dialog object that parameterizes generation of WSDL 2.0 documentation. See Document.GenerateWSDL20LDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

XBRLDocumentationDlg
See also

Property: XBRL20DocumentationDlg as XBRL20DocumentationDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL 2.0 documentation. See Document.GenerateXBRLDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.
3.2.10 Document

See also

Properties and Methods

Standard automation properties
Application
Parent

Various document properties and methods
SetActiveDocument
Encoding
SetEncoding (obsolete)
Suggestions

XML validation
IsValid
SetExternalIsValid

Document conversion and transformation
AssignDTD
AssignSchema
AssignXSL
AssignXSLFO
ConvertDTDOrSchema
ConvertDTDOrSchemaEx
GenerateDTDOrSchema
GenerateDTDOrSchemaEx
FlattenDTDOrSchema
CreateSchemaDiagram
ExecuteXQuery
TransformXSL
TransformXSLEx
TransformXSLFO
GenerateProgramCode  (Enterprise Edition only)
GenerateSchemaDocumentation
GenerateSampleXML
ConvertToWSDL20

Document export
GetExportElementList
ExportToText
ExportToDatabase
CreateDBStructureFromXMLSchema
GetDBStructureList

File saving and naming
FullName
Name
Path
GetPathName (obsolete)
SetPathName (obsolete)
Document objects represent XML documents opened in Authentic Desktop.

Use one of the following properties to access documents that are already open Authentic Desktop:
- `Application.ActiveDocument`
- `Application.Documents`

Use one of the following methods to open a new document in Authentic Desktop:
- `Documents.OpenFile`
- `Documents.OpenURL`
- `Documents.OpenURLDialog`
- `Documents.NewFile`
- `Documents.NewFileFromText`
- `SpyProjectItem.Open`
- `Application.ImportFromDatabase`
- `Application.ImportFromSchema`
- `Application.ImportFromText`
- `Application.ImportFromWord`
- `Document(ConvertDTDOrSchema)`
- `Document.GenerateDTDOrSchema`
Events

OnBeforeSaveDocument

See also


XMLSpy scripting environment - VBScript:
    Function On_BeforeSaveDocument(objDocument, objDialog)
    End Function

    ' old handler - now obsolete
    ' return string to save to new file name
    ' return empty string to cancel save operation
    ' return nothing to save to original name
    Function On_SaveDocument(objDocument, strFilePath)
    End Function

XMLSpy scripting environment - JScript:
    function On_BeforeSaveDocument(objDocument, objDialog)
    {
    }

    // old handler - now obsolete
    // return string to save to new file name
    // return empty string to cancel save operation
    // return nothing to save to original name
    function On_SaveDocument(objDocument, strFilePath)
    {
    }

XMLSpy IDE Plugin:
    IXMLSpyPlugIn.OnEvent(27, ...) // nEventId = 27

Description
This event gets fired on any attempt to save a document. The file selection dialog object is initialized with the name chosen for the document file. You can modify this selection. To continue saving the document leave the FileSelectionDlg.DialogResult property of io_objDialog at its default value spyDialogOK. To abort saving of the document set this property to spyDialogCancel.

OnBeforeCloseDocument

See also

Event: OnBeforeCloseDocument(objDocument as Document) as Boolean

XMLSpy scripting environment - VBScript:
    Function On_BeforeCloseDocument(objDocument)
    End Function
' On_BeforeCloseDocument = False ' to prohibit closing of document
End Function

**XMLSpy scripting environment - JScript:**
function On_BeforeCloseDocument(objDocument)
{
    // return false; /* to prohibit closing of document */
}

**XMLSpy IDE Plugin:**
IXMLSpyPlugIn.OnEvent (28, ...) // nEventId = 28

**Description**
This event gets fired on any attempt to close a document. To prevent the document from being closed return false.

**OnBeforeValidate**

**See also**

**Event:** OnBeforeValidate(objDocument as Document, bOnLoading as Boolean, bOnCommand as Boolean) as Boolean

**XMLSpy scripting environment - VBScript:**
Function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
    On_BeforeValidate = bCancelDefaultValidation 'set by the script if necessary
End Function

**XMLSpy scripting environment - JScript:**
function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
{
    return bCancelDefaultValidation //set by the script if necessary
}

**XMLSpy IDE Plugin:**
IXMLSpyPlugIn.OnEvent (32, ...) // nEventId = 32

**Description**
This event gets fired before the document is validated. It is possible to suppress the default validation by returning false from the event handler. In this case the script should also set the validation result using the SetExternalValid method.

bOnLoading is true if the event is raised on the initial validation on loading the document.

bOnCommand is true whenever the user selected the Validate command from the Toolbar or menu.

Available with TypeLibrary version 1.5
**OnCloseDocument**

See also

**Event:** OnCloseDocument (objDocument as Document)

**XMLSpy scripting environment - VBScript:**
Function On_Close_Document (objDocument)
End Function

**XMLSpy scripting environment - JScript:**
function On_Close_Document (objDocument)
{
}

**XMLSpy IDE Plugin:**
IXMLSpyPlugIn.OnEvent (8, ...) // nEventId = 8

**Description**
This event gets fired as a result of closing a document. Do not modify the document from within this event.

**OnViewActivation**

See also

**Event:** OnViewActivation (objDocument as Document, eViewMode as SPYViewModes, bActivated as Boolean)

**XMLSpy scripting environment - VBScript:**
Function On_ViewActivation (objDocument, eViewMode, bActivated)
End Function

**XMLSpy scripting environment - JScript:**
function On_ViewActivation (objDocument, eViewMode, bActivated)
{
}

**XMLSpy IDE Plugin:**
IXMLSpyPlugIn.OnEvent (29, ...) // nEventId = 29

**Description**
This event gets fired whenever a view of a document becomes visible (i.e. becomes the active view) or invisible (i.e. another view becomes the active view or the document gets closed). However, the first view activation event after a document gets opened cannot be received, since there is no document object to get the event from. Use the Application.OnDocumentOpened event instead.
Application
See also

Property: Application as Application (read-only)

Description
Accesses the Authentic Desktop application object.

Errors
1400  The object is no longer valid.
1407  Invalid address for the return parameter was specified.

AssignDTD
See also

Method: AssignDTD(strDTDFile as String, bDialog as Boolean)

Description
The method places a reference to the DTD file "strDTDFile" into the document. Note that no error occurs if the file does not exist, or is not accessible. If bDialog is true Authentic Desktop presents a dialog to set the file.

Errors
1400  The object is no longer valid.
1409  You are not allowed to assign a DTD to the document.

AssignSchema
See also

Method: AssignSchema(strSchemaFile as String, bDialog as Boolean)

Description
The method places a reference to the schema file "strSchemaFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true Authentic Desktop presents a dialog to set the file.

Errors
1400  The object is no longer valid.
1409  You are not allowed to assign a schema file to the document.

AssignXSL
See also

Method: AssignXSL(strXSLFile as String, bDialog as Boolean)

Description
The method places a reference to the XSL file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true Authentic Desktop
presents a dialog to set the file.

**Errors**
- 1400  The object is no longer valid.
- 1409  You are not allowed to assign an XSL file to the document.

**AssignXSLFO**

**See also**

*Method:* AssignXSLFO *(strXSLFOFile as String, bDialog as Boolean)*

**Description**
The method places a reference to the XSLFO file "strXSLFOFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If *bDialog* is true Authentic Desktop presents a dialog to set the file.

**Errors**
- 1400  The object is no longer valid.
- 1409  You are not allowed to assign an XSL file to the document.

**AsXMLString**

**See also**

*Property:* AsXMLString  as String

**Description**
This property can be used to get or set the document content.

**Errors**
- 1400  The document object is no longer valid.
- 1404  Cannot create XMLData object.
- 1407  View mode cannot be switched.

**AuthenticView**

**See also**

*Method:* AuthenticView as AuthenticView (read-only)

**Description**
Returns an object that gives access to properties and methods specific to Authentic view. The object returned is only valid if the current document is opened in Authentic view mode. The lifetime of an object ends with the next view switch. Any attempt to access objects or any of its children afterwards will result in an error indicating that the object is invalid.

**Errors**
- 1400  The object is no longer valid.
- 1417  Document needs to be open in authentic view mode.

**Examples**

`---------------------------------------`
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ---------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
    ' we have an active document, now check for view mode
    If (objDocument.CurrentViewMode <> spyViewAuthentic) Then
        If (Not objDocument.SwitchViewMode (spyViewAuthentic)) Then
            MsgBox "Active document does not support authentic view mode"
        Else
            ' now it is safe to access the authentic view object
            Dim objAuthenticView
            Set objAuthenticView = objDocument.AuthenticView
            ' now use the authentic view object
        End If
    Else
        ' now it is safe to access the authentic view object
        Dim objAuthenticView
        Set objAuthenticView = objDocument.AuthenticView
        ' now use the authentic view object
    End If
Else
    MsgBox "No document is open"
End If

Close

See also

Method: Close (bDiscardChanges as Boolean)

Description
To close the document call this method. If bDiscardChanges is true and the document is modified, the document will be closed but not saved.

Errors
1400  The object is no longer valid.
1401  Document needs to be saved first.

ConvertDTDOrSchema

See also

Method: ConvertDTDOrSchema (nFormat as SPYDTDSchemaFormat, nFrequentElements as SPYFrequentElements)

Parameters

nFormat
Sets the schema output format to DTD or W3C.

nFrequentElements
Create complex elements as elements or complex types.

Description
ConvertDTDOrSchema takes an existing schema format and converts it into a different format. For a finer tuning of DTD/XSD conversion, use ConvertDTDOrSchemaEx.

Errors

1400 The object is no longer valid.
1412 Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead and Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead.

ConvertDTDOrSchemaEx

See also

Method: ConvertDTDOrSchemaEx (nFormat as SPYDTDSchemaFormat, nFrequentElements as SPYFrequentElements, sOutputPath as String, nOutputPathDialogAction as SPYDialogAction)

Parameters

nFormat
Sets the schema output format to DTD, or W3C.

nFrequentElements
Create complex elements as elements or complex types.

sOutputPath
The file path for the newly generated file.

nOutputPathDialogAction
Defines the dialog interaction for this call.

Description

ConvertDTDOrSchemaEx takes an existing schema format and converts it into a different format.

Errors

1400 The object is no longer valid.
1412 Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead and Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead.

ConvertToWSDL20

Method: ConvertToWSDL20 (sFilePath as String, bShowDialogs as Boolean)
**Parameters**

*sFilePath*
This specifies the file name of the converted WSDL. In case the source WSDL includes files which also must be converted, then only the directory part of the given path is used and the file names are generated automatically.

*bShowDialogs*
Defines whether file/folder selection dialogs are shown.

**Description**

Converts the WSDL 1.1 document to a WSDL 2.0 file. It will also convert any referenced WSDL files that are referenced from within this document. Note that this functionality is limited to WSDL View only. See `Document.CurrentViewMode` and `SPYViewModes`.

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1400</td>
<td>The document object is no longer valid.</td>
</tr>
<tr>
<td>1407</td>
<td>Invalid parameters have been passed or an empty file name has been specified as output target.</td>
</tr>
<tr>
<td>1417</td>
<td>The document is not opened in WSDL view, maybe it is not an '.wsdl' file.</td>
</tr>
<tr>
<td>1421</td>
<td>Feature is not available in this edition.</td>
</tr>
<tr>
<td>1433</td>
<td>WSDL 1.1 to WSDL 2.0 conversion failed.</td>
</tr>
</tbody>
</table>

**CreateChild**

**See also**

*Method:* `CreateChild(nKind as SPYXMLDataKind) as XMLData`

**Return Value**

The method returns the new `XMLData` object.

**Description**

To create a new `XMLData` object use the `CreateChild()` method.

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1400</td>
<td>The object is no longer valid.</td>
</tr>
<tr>
<td>1404</td>
<td>Cannot create <code>XMLData</code> object.</td>
</tr>
<tr>
<td>1407</td>
<td>Invalid address for the return parameter was specified.</td>
</tr>
</tbody>
</table>

**CreateDBStructureFromXMLSchema**

**See also**

*Method:* `CreateDBStructureFromXMLSchema(pDatabase as DatabaseConnection, pTables as ElementList, bDropTableWithExistingName as Boolean) as String`

**Description**

`CreateDBStructureFromXMLSchema` exports the given tables to the specified database. The function returns the SQL statements that were necessary to perform the changes.

See also `GetDBStructureList`. 
CreateSchemaDiagram

See also

*Method:* `CreateSchemaDiagram (nKind as SPYSchemaDefKind, strName as String, strFile as String)`

*Return Value*
None.

*Description*
The method creates a diagram of the schema type `strName` of kind `nKind` and saves the output file into `strFile`. Note that this functionality is limited to Schema View only. See `Document.CurrentViewMode` and `SPYViewModes`.

*Errors*
- 1400: The object is no longer valid.
- 1414: Failed to save diagram.
- 1415: Invalid schema definition type specified.

CurrentViewMode

See also

*Method:* `CurrentViewMode as SPYViewModes`

*Description*
The property holds the current view mode of the document. See also `Document.SwitchViewMode`.

*Errors*
- 1400: The object is no longer valid.
- 1407: Invalid address for the return parameter was specified.

DataRoot

See also

*Property:* `DataRoot as XMLData` (read-only)

*Description*
This property provides access to the document's first XMLData object of type `spyXMLDataElement`. This is typically the root element for all document content data. See `XMLSpyDocument.RootElement` to get the root element of the whole document including XML.
prolog data. If the `CurrentViewMode` is not `spyViewGrid` or `spyViewAuthentic` an `UpdateXMLData` may be necessary to get access to the latest `XMLData`.

**Errors**

1400  The document object is no longer valid.
1407  Invalid address for the return parameter was specified.

**DocEditView**

See also

*Method:* `DocEditView as DocEditView`

**Description**

Holds a reference to the current Authentic View object.

**Errors**

1400  The object is no longer valid.
1407  Invalid address for the return parameter was specified.
1417  Document needs to be open in authentic view mode.

**Encoding**

See also

*Property:* `Encoding as String`

**Description**

This property provides access to the document's encoding value. However, this property can only be accessed when the document is opened in `spyViewGrid`, `spyViewText` or `spyViewAuthentic`. See `CurrentViewMode` on how to detect that a document's actual view mode.

This property makes the method `SetEncoding` obsolete.

Possible values are, for example:

- 8859-1,
- 8859-2,
- ASCII, ISO-646,
- 850,
- 1252,
- 1255,
- SHIFT-JIS, MS-KANJI,
- BIG5, FIVE,
- UTF-7,
- UTF-8,
- UTF-16

**Errors**

1400  The document object is no longer valid.
1407  Invalid address for the return parameter was specified.
1416  Operation not supported in current view mode.
EndChanges
See also

*Method:* EndChanges()

*Description*
Use the method `EndChanges` to display all changes since the call to `Document.StartChanges`.

*Errors*
1400 The object is no longer valid.

ExecuteXQuery
See also

*Method:* ExecuteXQuery(*strXMLFileName* as String)

*Description*
Execute the XQuery statements contained in the document of the document object. Either an XQuery execution or an XQuery Update is performed depending on the file extension of the document. Use the XML file specified in the argument as the XML target document that the XQuery document processes.

- If the document has an XQuery file extension as defined in the Options dialog of Authentic Desktop, then an XQuery execution is performed. By default: .xq, .xql, and .xquery are set as XQuery file extensions in Authentic Desktop.
- If the document has an XQuery Update file extension as defined in the Options dialog of Authentic Desktop, then an XQuery Update action is performed. By default: .xqu is set as an XQuery Update file extension in Authentic Desktop.

If your XQuery script does not use an XML source, set the parameter `strXMLFileName` to an empty string.

*Errors*
1400 The document object is no longer valid.
1423 XQuery transformation error.
1424 Not all files required for operation could be loaded. Most likely, the file specified in `strXMLFileName` does not exist or is not valid.

ExportToDatabase
See also

*Method:* ExportToDatabase (*pFromChild* as XMLData, *pExportSettings* as ExportSettings, *pDatabase* as DatabaseConnection)

*Description*
`ExportToDatabase` exports the XML document starting with the element `pFromChild`. The
parameter `pExportSettings` defines the behaviour of the export (see `Application.GetExportSettings`). The parameter `pDatabase` specifies the destination of the export (see `Application.GetDatabaseSettings`). `UpdateXMLData()` might be indirectly needed as you have to pass the `XMLData` as parameter to this function.

**Errors**

1400  The object is no longer valid.
1407  Invalid parameter or invalid address for the return parameter was specified.
1416  Error during export.
1429  Database selection missing.
1430  Document export failed.

**Example**

```vba
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument

' set the behaviour of the export with ExportSettings
Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings

' set the destination with DatabaseConnection
Dim objDB As DatabaseConnection
Set objDB = objSpy.GetDatabaseSettings

objDB.CreateMissingTables = True
objDB.CreateNew = True
objDB.File = "C:\Export.mdb"

objDoc.ExportToDatabase objDoc.RootElement, objExpSettings, objDB
If Err.Number <> 0 Then
    a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) & "Description: " & Err.Description)
End If
```

**ExportToText**

See also

**Method:** `ExportToText` *(`pFromChild` as `XMLData`, `pExportSettings` as `ExportSettings`, `pTextSettings` as `TextImportExportSettings`)*

**Description**

`ExportToText` exports tabular information from the document starting at `pFromChild` into one or many text files. Columns of the resulting tables are generated in alphabetical order of the column header names. Use `GetExportElementList` to learn about the data that will be exported. The parameter `pExportSettings` defines the specifics for the export. Set the property `ExportSettings.ElementList` to the - possibly modified - list returned by `GetExportElementList` to avoid exporting all contained tables. The parameter `pTextSettings` defines the options specific to text export and import. You need to set the property `TextImportExportSettings.DestinationFolder` before you call `ExportToText`. `UpdateXMLData()` might be indirectly needed as you have to pass the `XMLData`
as parameter to this function.

**Errors**

- **1400** The object is no longer valid.
- **1407** Invalid parameter or invalid address for the return parameter was specified.
- **1416** Error during export.
- **1430** Document export failed.

**Example**

' ---------------------------------------------------------
' VBA client code fragment - export document to text files
' ---------------------------------------------------------
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument

Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings
objExpSettings.ElementList = objDoc.GetExportElementList(
    objDoc.RootElement,
    objExpSettings)

Dim objTextExp As TextImportExportSettings
Set objTextExp = objSpy.GetTextExportSettings
objTextExp.HeaderRow = True
objTextExp.DestinationFolder = "C:\Exports"

On Error Resume Next
objDoc.ExportToText objDoc.RootElement, objExpSettings, objTextExp

If Err.Number <> 0 Then
    a = MsgBox("Error: ") & (Err.Number - vbObjectError) & Chr(13) & "Description: ") & Err.Description
End If

**FlattenDTDOrSchema**

*Method: FlattenDTDOrSchema (sOutputPath as String, nOutputPathDialogAction as SPYDialogAction)*

**Parameters**

- **sOutputPath**
The file path for the newly generated file.

- **nOutputPathDialogAction**
Defines the dialog interaction for this call.

**Description**

FlattenDTDOrSchema takes an existing DTD or schema, generates a flattened file, and saves the generated file at the specified location. In the case of DTDs, flattening removes parameter entities and produces a single DTD from a collection of modules; sections marked **IGNORE** are suppressed and unused parameter entities are deleted. When an XML Schema is flattened, the components of all included schemas are added as global components of the active schema, and
Errors
1400 The object is no longer valid.
1412 Error during conversion.

FullName
See also

Property: FullName as String

Description
This property can be used to get or set the full file name - including the path - to where the
document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName and SetPathName obsolete.

Errors
1400 The document object is no longer valid.
1402 Empty string has been specified as full file name.

GenerateDTDOrSchema
See also

Method: GenerateDTDOrSchema (nFormat as SPYDTDSchemaFormat, nValuesList as integer, nDetection as SPYTypeDetection, nFrequentElements as SPYFrequentElements)

Parameters
nFormat
Sets the schema output format to DTD, or W3C.

nValuesList
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

nDetection
Specifies granularity of simple type detection.

nFrequentElements
Shall the types for all elements be defined as global? Use that value spyGlobalComplexType to
define them on global scope. Otherwise, use the value spyGlobalElements.

Description
Use this method to automatically generate a DTD or schema for the current XML document.
For a finer tuning of DTD / schema generation, use GenerateDTDOrSchemaEx.
Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode and SPYViewModes.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateDTDOrSchemaEx

See also

Method: GenerateDTDOrSchemaEx (objDlg as DTDSchemaGeneratorDlg) as Document

Description
Use this method to automatically generate a DTD or schema for the current XML document. A DTDSchemaGeneratorDlg object is used to pass information to the schema/DTD generator. The generation process can be configured to allow user interaction or run without further user input. Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode and SPYViewModes.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateProgramCode

Method: GenerateProgramCode (objDlg as CodeGeneratorDlg)

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Generate Java, C++ or C# class files from the XML Schema definitions in your document. A CodeGeneratorDlg object is used to pass information to the code generator. The generation process can be configured to allow user interaction or run without further user input.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1421 Feature not available in this edition

GenerateSampleXML

Method: GenerateSampleXML (objDlg as GenerateSampleXMLDlg) as Document

Description
Generates a sample XML if the document is a schema or DTD. Use Dialogs.GenerateSampleXMLDlg to get an initialized set of options.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.
GenerateSchemaDocumentation

**Method:** GenerateSchemaDocumentation *(objDlg as SchemaDocumentationDlg)*

**Description**
Generate documentation for a schema definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use *Dialogs.SchemaDocumentationDlg* to get an initialized set of options. As a minimum, you will need to set the property *SchemaDocumentationDlg.OutputFile* before starting the generation process. Note that this functionality is limited to Schema View only. See *Document.CurrentViewMode* and *SPYViewModes*.

**Errors**
- 1400 The document object is no longer valid.
- 1407 Invalid parameters have been passed or an empty file name has been specified as output target.
- 1417 The document is not opened in schema view, maybe it is not an `.xsd` file.
- 1421 Feature is not available in this edition.
- 1422 Error during generation

GenerateWSDL20Documentation

**Method:** GenerateWSDL20Documentation *(objDlg as WSDL20DocumentationDlg)*

**Description**
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use *Dialogs.WSDL20DocumentationDlg* to get an initialized set of options. As a minimum, you will need to set the property *WSDL20DocumentationDlg.OutputFile* before starting the generation process. Note that this functionality is limited to WSDL View only. See *Document.CurrentViewMode* and *SPYViewModes*.

**Errors**
- 1400 The document object is no longer valid.
- 1407 Invalid parameters have been passed or an empty file name has been specified as output target.
- 1417 The document is not opened in schema view, maybe it is not an `.xsd` file.
- 1421 Feature is not available in this edition.
- 1422 Error during generation

GenerateWSDLDocumentation

**Method:** GenerateWSDLDocumentation *(objDlg as WSDLDocumentationDlg)*

**Description**
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use *Dialogs.WSDLDocumentationDlg* to get an initialized set of options. As a minimum, you will need to set the property *WSDLDocumentationDlg.OutputFile* before starting the generation process. Note that this functionality is limited to WSDL View only. See
**GenerateXBRLDocumentation**

**Method:** GenerateXBRLDocumentation *(objDlg as XBRLDocumentationDlg)*  

**Description**  
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use `Dialogs.XBRLDocumentationDlg` to get an initialized set of options. As a minimum, you will need to set the property `XBRLDocumentationDlg.OutputFile` before starting the generation process. Note that this functionality is limited to XBRL View only. See `Document.CurrentViewMode` and `SPYViewModes`.

**Errors**  
1400  The document object is no longer valid.  
1407  Invalid parameters have been passed or an empty file name has been specified as output target.  
1417  The document is not opened in schema view, maybe it is not an '.xsd' file.  
1421  Feature is not available in this edition.  
1422  Error during generation

**GetDBStructureList**

**See also**

**Method:** GetDBStructureList *(pDatabase as DatabaseConnection) as ElementList*

**Description**  
GetDBStructureList creates a collection of elements from the Schema document for which tables in the specified database are created. The function returns a collection of `ElementListItem` where the properties `ElementListItem.Name` contain the names of the tables.

See also [CreateDBStructureFromXMLSchema](#).

**Errors**  
1400  The object is no longer valid.  
1427  Failed creating parser for the specified XML.  
1428  Export of element list failed.  
1429  Database selection missing.
GetExportElementList
See also

Method: GetExportElementList \((p\text{FromChild} \text{ as } \text{XMLData}, \ p\text{ExportSettings} \text{ as } \text{ExportSettings})\) as \text{ElementList}

Description
GetExportElementList creates a collection of elements to export from the document, depending on the settings in \text{pExportSettings} and starting from the element \text{pFromChild}. The function returns a collection of \text{ElementListItems} where the properties \text{ElementListItem.Name} contain the names of the tables that can be exported from the document. The property \text{ElementListItem.FieldCount} contains the number of columns in the table. The property \text{ElementListItem.RecordCount} contains the number of records in the table. The property \text{ElementListItem.ElementKind} is unused. \text{UpdateXMLData()} might be indirectly needed as you have to pass the \text{XMLData} as parameter to this function.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1427 Failed creating parser for the specified XML.
1428 Export of element list failed.

GetPathName (obsolete)

Superseded by \textbf{Document.FullName}

// ------ javascript sample ------
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;

See also

Method: GetPathName() as \text{String}

Description
The method \text{GetPathName} gets the path of the active document.

See also \textbf{Document.SetPathName} (obsolete).

GridView
See also

Property: GridView \text{ as GridView}
Description
This property provides access to the grid view functionality of the document.

Errors
1400  The object is no longer valid.
1407  Invalid address for the return parameter was specified.
1417  Document needs to be open in enhanced grid view mode.

IsModified
See also

Property: IsModified as Boolean

Description
True if the document is modified.

Errors
1400  The object is no longer valid.
1407  Invalid address for the return parameter was specified.

IsValid
See also

Method: IsValid(strError as Variant) as Boolean

Return Value
True if the document is valid, false if not. To call IsValid(), the application GUI must be visible. (If you wish to validate without the GUI being visible, please use Altova RaptorXML Server.)

Description
IsValid validates the document against its associated schema or DTD. strError gives you the same error message as when you validate the file within the GUI.

Errors
1400  The object is no longer valid.
1407  Invalid parameter or invalid address for the return parameter was specified.
1408  Unable to validate file.

IsValidEx

Method: IsValidEx(nXSDVersion as SPYValidateXSDVersion, nErrorLimit as int, nErrorFormat as SPYValidateErrorFormat, out strError as Variant) as Boolean

Return Value
True if the document is valid, false if not.
Description
IsValidEx validates the document against its associated schema or DTD.

In parameters:
nXSDVersion which is an enumeration value of SPYValidateXSDVersion that selects the XSD version to validate against.
nErrorLimit which is an integer. Values must be 1 to 999.
nErrorFormat which is an enumeration value of SPYValidateErrorFormat that selects the XSD version to validate against.

Out parameter:
strError is the error message, and is the same as that received when validating the file within the GUI.

Errors
1400  The object is no longer valid.
1407  Invalid parameter or invalid address for the return parameter was specified.
1408  Unable to validate file.

IsWellFormed

See also

Method: IsWellFormed (pData as XMLData, bWithChildren as Boolean, strError as Variant, nErrorPos as Variant, pBadXMLData as Variant) as Boolean

Return Value
True if the document is well formed.

Description
IsWellFormed checks the document for well-formedness starting at the element pData.

If the document is not well formed, strError contains an error message, nErrorPos the position in the file and pBadXMLData holds a reference to the element which breaks the well-formedness. These out-parameters are defined as VARIANTs to support scripting languages like VBScript.

Errors
1400  The object is no longer valid.
1407  Invalid parameter or invalid address for the return parameter was specified.

Example
See IsValid.

Name

See also

Property: Name as String (read-only)
Description
Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Parent
See also

Property: Parent as Documents (read-only)

Description
Access the parent of the document object.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Property: Parent as Application (read-only)

Path
See also

Property: Path as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

RootElement
See also

Property: RootElement as XMLData (read-only)

Description
The property RootElement provides access to the root element of the XML structure of the document including the XML prolog data. To access the first element of a document's content navigate to the first child of kind spyXMLDataElement or use the Document.DataRoot property. If the CurrentViewMode is not spyViewGrid or spyViewAuthentic an UpdateXMLData may be necessary to get access to the latest XMLData.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

**Save**

See also

*Method: Save()*

**Description**
The method writes any modifications of the document to the associated file. See also `Document.FullName`.

**Errors**
- 1400 The document object is no longer valid.
- 1407 An empty file name has been specified.
- 1403 Error when saving file, probably the file name is invalid.

**SaveAs**

See also

*Method: SaveAs *(strFileName as String)*

**Description**
Save the document to the file specified. If saving was successful, the `FullName` property gets set to the specified file name.

**Errors**
- 1400 The document object is no longer valid.
- 1407 An empty file name has been specified.
- 1403 Error when saving file, probably the file name is invalid.

**Saved**

See also

*Property: Saved* as Boolean (read-only)

**Description**
This property can be used to check if the document has been saved after the last modifications. It returns the negation of `IsModified`.

**Errors**
- 1400 The document object is no longer valid.
- 1407 Invalid address for the return parameter was specified.

**SaveInString**

See also

*Method: SaveInString*(pData as XMLData, bMarked as Boolean) as String
**Parameters**

pData

XMLData element to start. Set pData to Document.RootElement if you want to copy the complete file.

bMarked

If bMarked is true, only the elements selected in the grid view are copied.

**Return Value**

Returns a string with the XML data.

**Description**

SaveToString starts at the element pData and converts the XMLData objects to a string representation. UpdateXMLData() might be indirectly needed as you have to pass the XMLData as parameter to this function.

**Errors**

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

**SaveToURL**

**See also**

**Method:** SaveToURL (strURL as String, strUser as String, strPassword as String)

**Return Value**

**Description**

SaveToURL() writes the document to the URL strURL. This method does not set the permanent file path of the document.

**Errors**

1400 The object is no longer valid.
1402 Invalid URL specified.
1403 Error while saving to URL.

**SetActiveDocument**

**See also**

**Method:** SetActiveDocument()

**Description**

The method sets the document as the active and brings it to the front.

**Errors**

1400 The object is no longer valid.

**SetEncoding (obsolete)**

Superseded by Document.Encoding
// ----- javascript sample -----  
// instead of:  
// Application.ActiveDocument.SetEncoding("UTF-16");  
// use now:  
Application.ActiveDocument.Encoding = "UTF-16";

See also

**Method:** *SetEncoding (strEncoding as String)*

**Description**

*SetEncoding* sets the encoding of the document like the menu item "File/Encoding..." in Authentic Desktop. Possible values for *strEncoding* are, for example:

- 8859-1,
- 8859-2,
- ASCII, ISO-646,
- 850,
- 1252,
- 1255,
- SHIFT-JIS, MS-KANJI,
- BIG5, FIVE,
- UTF-7,
- UTF-8,
- UTF-16
**SetExternallsIsValid**

See also

*Methode*: **SetExternallsIsValid** *(bValid as Boolean)*

**Parameters**

bValid
Sets the result of an external validation process.

**Description**

The internal information set by this method is only queried on cancelling the default validation in any `OnBeforeValidate` handler.

Available with TypeLibrary version 1.5

**Errors**

1400  The object is no longer valid.

---

**SetPathName (obsolete)**

Superseded by `Document.FullName`

```javascript
// ----- javascript sample -----  
// instead of:  
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");  
// use now:  
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

See also

*Methode*: **SetPathName** *(strPath as String)*

**Description**

The method `SetPathName` sets the path of the active document. `SetPathName` only copies the string and does not check if the path is valid. All succeeding save operations are done into this file.

---

**StartChanges**

See also

*Methode*: **StartChanges()**

**Description**

After `StartChanges` is executed Authentic Desktop will not update its editor windows until `Document.EndChanges` is called. This increases performance of complex tasks to the XML structure.
Errors
1400  The object is no longer valid.

Suggestions
Property: Suggestions as Array

Description
This property contains the last valid user suggestions for this document. The XMLSpy generated suggestions can be modified before they are shown to the user in the OnBeforeShowSuggestions event.

Errors
1400  The object is no longer valid.
1407  Invalid parameter or invalid address for the return parameter was specified.

SwitchViewMode
See also

Method: SwitchViewMode (nMode as SPYViewModes) as Boolean

Return value
Returns true if view mode is switched.

Description
The method sets the current view mode of the document in Authentic Desktop. See also Document.CurrentViewMode.

Errors
1400  The object is no longer valid.
1407  Invalid address for the return parameter was specified.
1417  Invalid view mode specified.

TextView
See also

Property: TextView as TextView

Description
This property provides access to the text view functionality of the document.

Errors
1400  The object is no longer valid.
1407  Invalid address for the return parameter was specified.

Title
See also

Property: Title as String (read-only)
Description
Title contains the file name of the document. To get the path and filename of the file use FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

TransformXSL
See also

Method: TransformXSL()

Description
TransformXSL processes the XML document via the associated XSL file. See Document.AssignXSL on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLEx
See also

Method: TransformXSLEx(nAction as SPYDialogAction)

Description
TransformXSLEx processes the XML document via the associated XSL file. The parameter specifies whether a dialog asking for the result document name should pop up or not. See Document.AssignXSL on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLFO
See also

Method: TransformXSLFO()

Description
TransformXSLFO processes the XML document via the associated XSLFO file. See AssignXSLFO on how to place a reference to a XSLFO file into the document. You need to assign a FOP processor to Authentic Desktop before you can use this method.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.
TreatXBRLInconsistenciesAsErrors

Property: TreatXBRLInconsistenciesAsErrors as Boolean

Description
If this is set to true the Document.IsValid() method will return false for XBRL instances containing inconsistencies as defined by the XBRL Specification. The default value of this property is false.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

UpdateViews

See also

Method: UpdateViews()

Description
To redraw the Enhanced Grid View and the Tree View call UpdateViews. This can be important after you changed the XMLData structure of a document. This method does not redraw the text view of Authentic Desktop.

Errors
1400 The document object is no longer valid.

UpdateXMLData

See also

Method: UpdateXMLData() as Boolean

Description
The XMLData tree is updated from the current view. Please note that this can fail in case of the TextView if the current XML text is not well-formed. This is not necessary if CurrentViewMode is spyViewGrid or spyViewAuthentic because these views keep the XMLData updated.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.
3.2.11 Documents

See also

Properties
Count
Item

Methods
NewAuthenticFile
NewFile
NewFileFromText
OpenAuthenticFile
OpenFile
OpenURL
OpenURLDialog

Description
This object represents the set of documents currently open in Authentic Desktop. Use this object to open further documents or iterate through already opened documents.

Examples
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' iterate through open documents
' ---------------------------------------
Dim objDocuments
Set objDocuments = Application.Documents
For Each objDoc In objDocuments
  'do something useful with your document
  objDoc.SetActiveDocument()
Next

// ---------------------------------------
// XMLSpy scripting environment - JScript
// close all open documents
// ---------------------------------------
for (var iter = new Enumerator (Application.Documents);
    ! iter.atEnd();
    iter.moveNext())
{
  // MsgBox ("Closing file " + iter.item().Name);
  iter.item().Close (true);
}

Count
See also

Property: Count as long

Description
Count of open documents.

**Errors**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1600</td>
<td>Invalid <code>Documents</code> object</td>
</tr>
<tr>
<td>1601</td>
<td>Invalid input parameter</td>
</tr>
</tbody>
</table>

**Item**

**See also**

**Method:** Item \( n \) as `Document`

**Description**

Gets the document with the index \( n \) in this collection. Index is 1-based.

**Errors**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1600</td>
<td>Invalid <code>Documents</code> object</td>
</tr>
<tr>
<td>1601</td>
<td>Invalid input parameter</td>
</tr>
</tbody>
</table>

**NewAuthenticFile**

**See also**

**Method:** NewAuthenticFile \( strSPSPath \) as String, \( strXMLPath \) as String as `Document`

**Parameters**

- **strSPSPath**
  The path to the SPS document.

- **strXMLPath**
  The new XML document name.

**Return Value**

The method returns the new document.

**Description**

NewAuthenticFile creates a new XML file and opens it in Authentic View using SPS design \( strSPSPath \).

**NewFile**

**See also**

**Method:** NewFile \( strFile \) as String, \( strType \) as String as `Document`

**Parameters**

- **strFile**
  Full path of new file.

- **strType**
  Type of new file as string (i.e. "xml", "xsd", ... )
**Return Value**
Returns the new file.

**Description**
NewFile creates a new file of type strType (i.e. "xml"). The newly created file is also the ActiveDocument.

**NewFileFromText**

See also

*Method:* NewFileFromText (strText as String, strType as String) as Document

**Parameters**

strText
The content of the new document in plain text.

strType
Type of the document to create (i.e. "xml").

**Return Value**
The method returns the new document.

**Description**
NewFileFromText creates a new document with strText as its content.

**OpenAuthenticFile**

See also

*Method:* OpenAuthenticFile (strSPSPath as String, strXMLPath as String) as Document

**Parameters**

strSPSPath
The path to the SPS document.

strXMLPath
The path to the XML document (can be empty).

**Return Value**
The method returns the new document.

**Description**
OpenAuthenticFile opens an XML file or database in Authentic View using SPS design strSPSPath.

**OpenFile**

See also

*Method:* OpenFile (strPath as String, bDialog as Boolean) as Document
Parameters

strPath
Path and file name of file to open.

bDialog
Show dialogs for user input.

Return Value
Returns the opened file on success.

Description
OpenFile opens the file strPath. If bDialog is TRUE, a file-dialog will be displayed.

Example

Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile(strFile, False)

OpenURL

See also

Method: OpenURL (strURL as String, nURLType as SPYURLTypes, nLoading as SPYLoading, strUser as String, strPassword as String) as Document

Parameters

strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentication. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURL opens the URL strURL.
**OpenURLDialog**

See also

*Method:* `OpenURLDialog(strURL as String, nURLType as SPYURLTypes, nLoading as SPYLoading, strUser as String, strPassword as String) as Document`

**Parameters**

- **strURL**
  URL to open as document.

- **nURLType**
  Type of document to open. Set to -1 for auto detection.

- **nLoading**
  Set `nLoading` to 0 (zero) if you want to load it from cache or proxy. Otherwise set `nLoading` to 1.

- **strUser**
  Name of the user if required. Can be empty.

- **strPassword**
  Password for authentication. Can be empty.

**Return Value**

The method returns the opened document.

**Description**

`OpenURLDialog` displays the "open URL" dialog to the user and presets the input fields with the given parameters.
3.2.12 DTDSchemaGeneratorDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

DTDSchemaFormat
ValueList
TypeDetection
FrequentElements
MergeAllEqualNamed
ResolveEntities
AttributeTypeDefinition
GlobalAttributes
OnlyStringEnums
MaxEnumLength
OutputPath
OutputPathDialogAction

Description
Use this object to configure the generation of a schema or DTD. The method GenerateDTDSchemaEx expects a DTDSchemaGeneratorDlg as parameter to configure the generation as well as the associated user interactions.

Application

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

AttributeTypeDefinition

Property: AttributeTypeDefinition as SPYAttributeTypeDefinition

Description
Specifies how attribute definitions get merged.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.
DTDSchemaFormat

Property: DTDSchemaFormat as SPYDTDSchemaFormat

Description
Sets the schema output format to DTD, or W3C.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

FrequentElements

Property: FrequentElements as SPYFrequentElements

Description
Shall the types for all elements be defined as global? Use that value spyGlobalComplexType to define them on global scope. Otherwise, use the value spyGlobalElements.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

GlobalAttributes

Property: GlobalAttributes as Boolean

Description
Shall attributes with same name and type be resolved globally?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

MaxEnumLength

Property: MaxEnumLength as Integer

Description
Specifies the maximum number of characters allowed for enumeration names. If one value is longer than this, no enumeration will be generated.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.
MergeAllEqualNamed

**Property:** MergeAllEqualNamed as Boolean

**Description**
Shall types of all elements with the same name be merged into one type?

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.

OnlyString Enums

**Property:** OnlyString Enums as Boolean

**Description**
Specifies if enumerations will be created only for plain strings or all types of values.

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.

OutputPath

**Property:** OutputPath as String

**Description**
Selects the file name for the generated schema/DTD.

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.

OutputPathDialogAction

**Property:** OutputPathDialogAction as SPYDialogAction

**Description**
Defines how the sub-dialog for selecting the schema/DTD output path gets handled. Set this value to spyDialogUserInput(2) to show the dialog with the current value of the **OutputPath** property as default. Use spyDialogOK(0) to hide the dialog from the user.

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.
Parent

**Property:** Parent as *Dialogs* (read-only)

**Description**
Access the parent of the object.

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.

ResolveEntities

**Property:** ResolveEntities as Boolean

**Description**
Shall all entities be resolved before generation starts? If yes, an info-set will be built.

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.

TypeDetection

**Property:** TypeDetection as *SPTTypeDetection*

**Description**
Specifies granularity of simple type detection.

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.

ValueList

**Property:** ValueList as Integer

**Description**
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

**Errors**
- 3000  The object is no longer valid.
- 3001  Invalid address for the return parameter was specified.
3.2.13 ElementList

See also

Properties
Count
Item

Methods
RemoveElement

Description
Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of ElementListItem are used.

It can hold
- a list of table names returned by a call to Application.GetDatabaseTables,
- a list of field names returned by a call to Application.GetDatabaseImportElementList or Application.GetTextImportElementList,
- a field name filter list used in Application.ImportFromDatabase and Application.ImportFromText,
- a list of table names and counts for their rows and columns as returned by calls to GetExportElementList or GetExportElementCount or
- a field name filter list used in Document.ExportToDatabase and Document.ExportToText.

Count
See also

Property: Count as long (read-only)

Description
Count of elements in this collection.

Item
See also

Method: Item(n as long) as ElementListItem

Description
Gets the element with the index n from this collection. The first item has index 1.

RemoveElement
See also

Method: RemoveElement(Index as long)

Description
RemoveElement removes the element Index from the collection. The first item has index 1.
3.2.14 **ElementListItem**

**See also**

**Properties**
- **Name**
- **ElementKind**
- **FieldCount**
- **RecordCount**

**Description**
An element in an **ElementList**. Usage of its properties depends on the purpose of the element list. For details see **ElementList**.

**ElementKind**

**See also**

*Property:* ElementKind as **SPYXMLDataKind**

**Description**
Specifies if a field should be imported as XML element (data value of spyXMLDataElement) or attribute (data value of spyXMLDataAttr).

**FieldCount**

**See also**

*Property:* FieldCount as long (read-only)

**Description**
Count of fields (i.e. columns) in the table described by this element. This property is only valid after a call to **Document.GetExportElementList**.

**Name**

**See also**

*Property:* Name as String (read-only)

**Description**
Name of the element. This is either the name of a table or a field, depending on the purpose of the element list.

**RecordCount**

**See also**

*Property:* RecordCount as long (read-only)
Description
Count of records (i.e. rows) in the table described by this element. This property is only valid after a call to Document.GetExportElementList.
3.2.15 ExportSettings

See also

Properties

- **ElementList**
- **EntitiesToText**
- **ExportAllElements**
- **SubLevelLimit**
- **FromAttributes**
- **FromSingleSubElements**
- **FromTextValues**
- **CreateKeys**
- **IndependentPrimaryKey**

**Namespace**

- **ExportCompleteXML**
- **StartFromElement**

**Description**

ExportSettings contains options used during export of XML data to a database or text file.

**CreateKeys**

See also

*Property:* CreateKeys as Boolean

**Description**

This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

**ElementList**

See also

*Property:* ElementList as ElementList

**Description**

Default is empty list. This list of elements defines which fields will be exported. To get the list of available fields use **Document.GetExportElementList**. It is possible to prevent exporting columns by removing elements from this list with **ElementList.RemoveElement** before passing it to **Document.ExportToDatabase** or **Document.ExportToText**.

**EntitiesToText**

See also
Property: EntitiesToText as Boolean

Description
Defines if XML entities should be converted to text or left as they are during export. Default is True.

ExportAllElements
See also

Property: ExportAllElements as Boolean

Description
If set to true, all elements in the document will be exported. If set to false, then ExportSettings.SubLevelLimit is used to restrict the number of sub levels to export. Default is true.

ExportCompleteXML
See also

Property: ExportCompleteXML as Boolean

Description
Defines whether the complete XML is exported or only the element specified by StartFromElement and its children. Default is True.

FromAttributes
See also

Property: FromAttributes as Boolean

Description
Set FromAttributes to false if no export data should be created from attributes. Default is True.

FromSingleSubElements
See also

Property: FromSingleSubElements as Boolean

Description
Set FromSingleSubElements to false if no export data should be created from elements. Default is True.

FromTextValues
See also

Property: FromTextValues as Boolean
Description
Set `FromTextValues` to false if no export data should be created from text values. Default is True.

IndependentPrimaryKey
See also

*Property*: `IndependentPrimaryKey` as Boolean

Description
Turns creation of independent primary key counter for every element on or off. If `ExportSettings.CreateKeys` is False, this property will be ignored. Default is True.

Namespace
See also

*Property*: `Namespace` as `SPYExportNamespace`

Description
The default setting removes all namespace prefixes from the element names. In some database formats the colon is not a legal character. Default is `spyNoNamespace`.

StartFromElement
See also

*Property*: `StartFromElement` as String

Description
Specifies the start element for the export. This property is only considered when `ExportCompleteXML` is false.

SubLevelLimit
See also

*Property*: `SubLevelLimit` as Integer

Description
Defines the number of sub levels to include for the export. Default is 0. This property is ignored if `ExportSettings.ExportAllElements` is true.
3.2.16 FileSelectionDlg

See also

Properties and Methods

Standard automation properties

- **Application**
- **Parent**

Dialog properties

- **FullName**

Acceptance or cancellation of action that caused event

- **DialogAction**

Description

The dialog object allows you to receive information about an event and pass back information to the event handler in the same way as with a user dialog. Use the `FileSelectionDlg.FullName` to select or modify the file path and set the `FileSelectionDlg.DialogAction` property to cancel or agree with the action that caused the event.

Application

See also

**Property:** *Application* as *Application* (read-only)

Description

Access the Authentic Desktop application object.

Errors

2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

DialogAction

**Property:** *DialogAction* as *SPYDialogAction*

Description

If you want your script to perform the file selection operation without any user interaction necessary, simulate user interaction by either setting the property to `spyDialogOK(0)` or `spyDialogCancel(1)`.

To allow your script to fill in the default values but let the user see and react on the dialog, use the value `spyDialogUserInput(2)`. If you receive a FileSelectionDlg object in an event handler, `spyDialogUserInput(2)` is not supported and will be interpreted as `spyDialogOK(0)`.

Errors

2400 The object is no longer valid.
2401 Invalid value for dialog action or invalid address for the return parameter was specified.
**FullName**

*Property:* FullName as String

**Description**
Access the full path of the file the gets selected by the dialog. Most events that pass a FileSelectionDlg object to you allow you modify this value and thus influence the action that caused the event (e.g. load or save to a different location).

**Errors**
- 2400  The object is no longer valid.
- 2401  Invalid address for the return parameter was specified.

**Parent**

**See also**

*Property:* Parent as Dialogs (read-only)

**Description**
Access the parent of the object.

**Errors**
- 2400  The object is no longer valid.
- 2401  Invalid address for the return parameter was specified.
3.2.17 FindInFilesDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Find
RegularExpression
Replace
DoReplace
ReplaceOnDisk
MatchWholeWord
MatchCase
SearchLocation
StartFolder
IncludeSubfolders
SearchInProjectFilesDoExternal
FileExtension
AdvancedXMLSearch
XMLElementNames
XMLElementContents
XMLAttributeNames
XMLAttributeContents
XMLComments
XMLCData
XMLPI
XMLRest
ShowResult

Description
Use this object to configure the search (or replacement) for strings in files. The method FindInFiles expects a FindInFilesDlg as parameter.

AdvancedXMLSearch

Property: AdvancedXMLSearch as Boolean

Description
Specifies if the XML search properties (XMLElementNames, XElementContents, XMLAttributeNames, XMLAttributeContents, XMLComments, XMLCData, XMLPI and XMLRest) are considered. The default is false.

Errors
3500  The object is no longer valid.
3501  invalid address for the return parameter was specified.
Application

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

DoReplace

Property: DoReplace as Boolean

Description
Specifies if the matched string is replaced by the string defined in Replace. The default is false.

Errors
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

FileExtension

Property: FileExtension as String

Description
Specifies the file filter of the files that should be considered during the search. Multiple file filters must be delimited with a semicolon (eg: *.xml;*.dtd;a*.xsd). Use the wildcards * and ? to define the file filter.

Errors
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

Find

Property: Find as String

Description
Specifies the string to search for.

Errors
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

IncludeSubfolders

Property: IncludeSubfolders as Boolean

Description
Specifies if subfolders are searched too. The default is true.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

**MatchCase**

*Property:* MatchCase as Boolean

**Description**
Specifies if the search is case sensitive. The default is true.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

**MatchWholeWord**

*Property:* MatchWholeWord as Boolean

**Description**
Specifies whether the whole word or just a part of it must match. The default is false.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

**Parent**

*Property:* Parent as Dialogs (read-only)

**Description**
Access the parent of the object.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

**RegularExpression**

*Property:* RegularExpression as Boolean

**Description**
Specifies if Find contains a regular expression. The default is false.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.
Replace

*Property:* Replace as String

**Description**
Specifies the replacement string. The matched string is only replaced if [DoReplace](#) is set true.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

ReplaceOnDisk

*Property:* ReplaceOnDisk as Boolean

**Description**
Specifies if the replacement is done directly on disk. The modified file is not opened. The default is false.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

SearchInProjectFilesDoExternal

*Property:* SearchInProjectFilesDoExternal as Boolean

**Description**
Specifies if the external folders in the open project are searched, when a project search is performed. The default is false.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

SearchLocation

*Property:* SearchLocation as [SPYFindInFilesSearchLocation](#)

**Description**
Specifies the location of the search. The default is spyFindInFiles_Documents.

**Errors**
- 3500 The object is no longer valid.
- 3501 Invalid address for the return parameter was specified.

ShowResult

*Property:* ShowResult as Boolean

**Description**
Specifies if the result is displayed in the Find in Files output window. The default is false.
Errors
  3500  The object is no longer valid.
  3501  Invalid address for the return parameter was specified.

StartFolder
Property: StartFolder as String

Description
Specifies the folder where the disk search starts.

Errors
  3500  The object is no longer valid.
  3501  Invalid address for the return parameter was specified.

XMLAttributeContents
Property: XMLAttributeContents as Boolean

Description
Specifies if attribute contents are searched when AdvancedXMLSearch is true. The default is true.

Errors
  3500  The object is no longer valid.
  3501  Invalid address for the return parameter was specified.

XMLAttributeNames
Property: XMLAttributeNames as Boolean

Description
Specifies if attribute names are searched when AdvancedXMLSearch is true. The default is true.

Errors
  3500  The object is no longer valid.
  3501  Invalid address for the return parameter was specified.

XMLCData
Property: XMLCData as Boolean

Description
Specifies if CData tags are searched when AdvancedXMLSearch is true. The default is true.

Errors
  3500  The object is no longer valid.
  3501  Invalid address for the return parameter was specified.

XMLComments
Property: XMLComments as Boolean
Description
Specifies if comments are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLElementContents
Property: XMLElementContents as Boolean

Description
Specifies if element contents are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLElementNames
Property: XMLElementNames as Boolean

Description
Specifies if element names are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLPI
Property: XMLPI as Boolean

Description
Specifies if XML processing instructions are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLRest
Property: XMLRest as Boolean

Description
Specifies if the rest of the XML (which is not covered by the other XML search properties) is searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.
3.2.18 FindInFilesResult

See also

Properties and Methods

Standard automation properties
Application
Parent
Count
Item
Path
Document

Description
This object represents a file that matched the search criteria. It contains a list of FindInFilesResultMatch objects that describe the matching position.

Application

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

Document

Property: Path as Document (read-only)

Description
This property returns the Document object if the matched file is already open in XMLSpy.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.
**Item**

*Method:* Item(n as long) as FindInFilesResultMatch

*Description*

Gets the element with the index n from this collection. The first item has index 1.

**Parent**

*Property:* Parent as FindInFilesResults (read-only)

*Description*

Access the parent of the object.

**Errors**

- 3700  The object is no longer valid.
- 3701  Invalid address for the return parameter was specified.

**Path**

*Property:* Path as String (read-only)

*Description*

Returns the path of the file that matched the search criteria.

**Errors**

- 3700  The object is no longer valid.
- 3701  Invalid address for the return parameter was specified.
3.2.19 FindInFilesResultMatch

See also

Properties and Methods

Standard automation properties

**Application**

**Parent**

**Line**

**Position**

**Length**

**LineText**

**Replaced**

Description

Contains the exact position in the file of the matched string.

Application

**Property:** Application as Application (read-only)

Description

Access the Authentic Desktop application object.

Errors

3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Length

**Property:** Length as Long (read-only)

Description

Returns the length of the matched string.

Errors

3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Line

**Property:** Line as Long (read-only)

Description

Returns the line number of the match. The line numbering starts with 0.

Errors

3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.
**LineText**

*Property:* `LineText` as `String` (read-only)

**Description**
Returns the text of the line.

**Errors**
- 3800  The object is no longer valid.
- 3801  Invalid address for the return parameter was specified.

**Parent**

*Property:* `Parent` as `FindInFilesResult` (read-only)

**Description**
Access the parent of the object.

**Errors**
- 3800  The object is no longer valid.
- 3801  Invalid address for the return parameter was specified.

**Position**

*Property:* `Position` as `Long` (read-only)

**Description**
Returns the start position of the match in the line. The position numbering starts with 0.

**Errors**
- 3800  The object is no longer valid.
- 3801  Invalid address for the return parameter was specified.

**Replaced**

*Property:* `Replaced` as `Boolean` (read-only)

**Description**
True if the matched string was replaced.

**Errors**
- 3800  The object is no longer valid.
- 3801  Invalid address for the return parameter was specified.
3.2.20 FindInFilesResults

See also

Properties and Methods

Standard automation properties

- **Application**
- **Parent**
- **Count**
- **Item**

Description

This is the result of the `FindInFiles` method. It is a list of `FindInFilesResult` objects.

**Application**

Property: **Application** as `Application` (read-only)

Description

Access the Authentic Desktop application object.

Errors

- **3600** The object is no longer valid.
- **3601** Invalid address for the return parameter was specified.

**Count**

Property: **Count** as long (read-only)

Description

Count of elements in this collection.

**Item**

Method: **Item**(n as long) as `FindInFilesResult`

Description

Gets the element with the index n from this collection. The first item has index 1.

**Parent**

Property: **Parent** as `Application` (read-only)

Description

Access the parent of the object.

Errors

- **3600** The object is no longer valid.
- **3601** Invalid address for the return parameter was specified.
3.2.21 GenerateSampleXMLDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

NonMandatoryAttributes
NonMandatoryElements
RepeatCount
FillAttributesWithSampleData
FillElementsWithSampleData
ContentOfNullableElementsIsNonMandatory
TryToUseNonAbstractTypes
SchemaOrDTDAssignment
LocalNameOfRootElement
NamespaceURIOfRootElement
OptionsDialogAction

Properties that are no longer supported
TakeFirstChoice - obsolete
FillWithSampleData - obsolete
Optimization - obsolete

Description
Used to set the parameters for the generation of sample XML instances based on a W3C schema or DTD.

Application

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ChoiceMode

Property: ChoiceMode as SPYSampleXMLGenerationChoiceMode

Description
Specifies which elements will be generated.
Errors

2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.

ConsiderSampleValueHints

Property: ConsiderSampleValueHints as Boolean

Description
Selects whether to use SampleValueHints or not.

Errors

2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.

ContentOfNillableElementsIsNonMandatory

Property: ContentOfNillableElementsIsNonMandatory as Boolean

Description
If true, the contents of elements that are nillable will not be treated as mandatory.

Errors

2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.

FillAttributesWithSampleData

Property: FillAttributesWithSampleData as Boolean

Description
If true, attributes will have sample content.

Errors

2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.

FillElementsWithSampleData

Property: FillElementsWithSampleData as Boolean

Description
If true, elements will have sample content.

Errors

2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.
**FillWithSampleData - obsolete**

*Property:* `FillWithSampleData` as Boolean

*Description*
Do no longer access this property. Use `FillAttributesWithSampleData` and `FillElementsWithSampleData`, instead.

*Errors*
- 0001  The property is no longer accessible.

**LocalNameOfRootElement**

*Property:* `LocalNameOfRootElement` as String

*Description*
Specifies the local name of the root element for the generated sample XML.

*Errors*
- 2200  The object is no longer valid.
- 2201  Invalid address for the return parameter was specified.

**NamespaceURIOfRootElement**

*Property:* `NamespaceURIOfRootElement` as String

*Description*
Specifies the namespace URI of the root element for the generated sample XML.

*Errors*
- 2200  The object is no longer valid.
- 2201  Invalid address for the return parameter was specified.

**NonMandatoryAttributes**

*Property:* `NonMandatoryAttributes` as Boolean

*Description*
If true attributes which are not mandatory are created in the sample XML instance file.

*Errors*
- 2200  The object is no longer valid.
- 2201  Invalid address for the return parameter was specified.

**NonMandatoryElements**

*Property:* `NonMandatoryElements` as Boolean

*Description*
If true, elements which are not mandatory are created in the sample XML instance file.
Errors
2200 The object is no longer valid.
2201 Invalid address was specified for the return parameter.

Optimization - obsolete
*Property:* Optimization as [SPYSampleXMLGenerationOptimization](#)

Description
Do not use this property any longer. Use ChoiceMode and NonMandatoryElements.

Errors
0001 The property is no longer accessible.

OptionsDialogAction
*Property:* OptionsDialogAction as [SPYDialogAction](#)

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value `spyDialogUserInput(2)`. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use `spyDialogOK(0)`. Default is `spyDialogOK`.

Errors
2200 The object is no longer valid.
2201 Invalid value has been used to set the property.
    Invalid address for the return parameter was specified.

Parent
*Property:* Parent as [Dialogs](#) (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

RepeatCount
*Property:* RepeatCount as long

Description
Number of elements to create for repeated types.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.
SampleValueHints

Property: SampleValueHints as SPYSampleXMLGenerationSampleValueHints

Description
Specifies how to select data for the generated sample file.

Errors
2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.

SchemaOrDTDAssignment

Property: SchemaOrDTDAssignment as SPYSampleXMLGenerationSchemaOrDTDAssignment

Description
Specifies in which way a reference to the related schema or DTD - which is this document - will be generated into the sample XML.

Errors
2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.

TakeFirstChoice - obsolete

Property: TakeFirstChoice as Boolean

Description
Do no longer use this property.

Errors
0001  The property is no longer accessible.

TryToUseNonAbstractTypes

Property: TryToUseNonAbstractTypes as Boolean

Description
If true, tries to use a non-abstract type for xsi:type, if element has an abstract type.

Errors
2200  The object is no longer valid.
2201  Invalid address for the return parameter was specified.
3.2.22 GridView

See also

Methods
Deselect
Select
SetFocus

Properties
CurrentFocus
isVisible

Description
GridView Class

Events

OnBeforeDrag

See also

Event: OnBeforeDrag() as Boolean

XMLSpy scripting environment - VBScript:
Function On_BeforeDrag()
    ' On_BeforeStartEditing = False  ' to prohibit dragging
End Function

XMLSpy scripting environment - JScript:
function On_BeforeDrag()
{
    // return false;  /* to prohibit dragging */
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (4, ...)  // nEventId = 4

Description
This event gets fired on an attempt to drag an XMLData element on the grid view. Return false to prevent dragging the data element to a different position.

OnBeforeDrop

See also

Event: OnBeforeDrop(objXMLData as XMLData) as Boolean

XMLSpy scripting environment - VBScript:
Function On_BeforeDrop(objXMLData)
  ' On_BeforeStartEditing = False ' to prohibit dropping
End Function

**XMLSpy scripting environment - JScript:**

function On_BeforeDrop(objXMLData)
{
  // return false; /* to prohibit dropping */
}

**XMLSpy IDE Plugin:**

IXMLSpyPlugIn.OnEvent (5, ...) // nEventId = 5

**Description**

This event gets fired on an attempt to drop a previously dragged XMLData element on the grid view. Return `false` to prevent the data element to be moved from its original position to the drop destination position.

---

**OnBeforeStartEditing**

**See also**

**Event:** OnBeforeStartEditing(objXMLData as XMLData, bEditingName as Boolean) as Boolean

**XMLSpy scripting environment - VBScript:**

Function On_BeforeStartEditing(objXMLData, bEditingName)
  ' On_BeforeStartEditing = False ' to prohibit editing the field
End Function

**XMLSpy scripting environment - JScript:**

function On_BeforeStartEditing(objXMLData, bEditingName)
{
  // return false; /* to prohibit editing the field */
}

**XMLSpy IDE Plugin:**

IXMLSpyPlugIn.OnEvent (1, ...) // nEventId = 1

**Description**

This event gets fired before the editing mode for a grid cell gets entered. If the parameter `bEditingName` is true, the name part of the element will be edited, if its value is false, the value part will be edited.

---

**OnEditingFinished**

**See also**
Event: OnEditingFinished(objXMLData as XMLData, bEditingName as Boolean)

XMLSpy scripting environment - VBScript:
Function On_EditingFinished(objXMLData, bEditingName)
End Function

XMLSpy scripting environment - JScript:
function On_EditingFinished(objXMLData, bEditingName)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (2, ...)  // nEvent Id = 2

Description
This event gets fired when the editing mode of a grid cell gets left. The parameter bEditingName specifies if the name part of the element has been edited.

OnFocusChanged
See also

Event: OnFocusChanged(objXMLData as XMLData, bSetFocus as Boolean, bEditingName as Boolean)

XMLSpy scripting environment - VBScript:
Function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
End Function

XMLSpy scripting environment - JScript:
function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (3, ...)  // nEvent Id = 3

Description
This event gets fired whenever a grid cell receives or looses the cursor focus. If the parameter bEditingName is true, focus of the name part of the grid element has changed. Otherwise, focus of the value part has changed.

CurrentFocus
See also

Property: CurrentFocus as XMLData

Description
Holds the XML element with the current focus. This property is read-only.

**Deselect**

**See also**

*Method:* `Deselect(pData as XMLData)`

**Description**

Deselects the element `pData` in the grid view.

**IsVisible**

**See also**

*Property:* `IsVisible` as Boolean

**Description**

True if the grid view is the active view of the document. This property is read-only.

**Select**

**See also**

*Method:* `Select(pData as XMLData)`

**Description**

Selects the XML element `pData` in the grid view.

**SetFocus**

**See also**

*Method:* `SetFocus(pFocusData as XMLData)`

**Description**

Sets the focus to the element `pFocusData` in the grid view.
3.2.23 **SchemaDocumentationDlg**

**See also**

**Properties and Methods**

Standard automation properties
- **Application**
- **Parent**

Interaction and visibility properties
- **OutputFile**
- **OutputFileDialogAction**
- **OptionsDialogAction**
- **ShowProgressBar**
- **ShowResult**

Document generation options and methods
- **OutputFormat**
- **UseFixedDesign**
- **SPSFile**
- **EmbedDiagrams**
- **DiagramFormat**
- **MultipleOutputFiles**
- **EmbedCSSInHTML**
- **CreateDiagramsFolder**
- **GenerateRelativeLinks**

- **IncludeAll**
- **IncludeIndex**
- **IncludeGlobalAttributes**
- **IncludeGlobalElements**
- **IncludeLocalAttributes**
- **IncludeLocalElements**
- **IncludeGroups**
- **IncludeComplexTypes**
- **IncludeSimpleTypes**
- **IncludeAttributeGroups**
- **IncludeRedefines**
- **IncludeReferencedSchemas**

- **AllDetails**
- **ShowDiagram**
- **ShowNamespace**
- **ShowType**
- **ShowChildren**
- **ShowUsedBy**
- **ShowProperties**
- **ShowSingleFacets**
- **ShowPatterns**
- **ShowEnumerations**
- **ShowAttributes**
- **ShowIdentityConstraints**
**ShowAnnotations**  
**ShowSourceCode**  

**Description**  
This object combines all options for schema document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are initialized with the values used during the last generation of schema documentation. However, before using the object you have to set the `SetOutputFile` property to a valid file path. Use `OptionsDialogAction`, `OutputFileDialogAction` and `ShowProgressBar` to specify the level of user interaction desired. You can use `IncludeAll` and `AllDetails` to set whole option groups at once or the individual properties to operate on a finer granularity.

**AllDetails**  
**See also**

**Method:** `AllDetails (i_bDetailsOn as Boolean)`  

**Description**  
Use this method to turn all details options on or off.

**Errors**  
2900 The object is no longer valid.

**Application**  
**See also**

**Property:** `Application as Application` (read-only)

**Description**  
Access the Authentic Desktop application object.

**Errors**  
2900 The object is no longer valid.  
2901 Invalid address for the return parameter was specified.

**CreateDiagramsFolder**  
**See also**

**Property:** `CreateDiagramsFolder as Boolean`  

**Description**  
Set this property to `true`, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The default for the first run is false.
DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is PNG.

Errors
2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The default for the first run is true.

Errors
2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.
GenerateRelativeLinks
See also

Property: GenerateRelativeLinks as Boolean

Description
Set this property to true, to create relative paths to local files. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is false.

Errors
  2900  The object is no longer valid.
  2901  Invalid address for the return parameter was specified.

IncludeAll
See also

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
  2900  The object is no longer valid.

IncludeAttributeGroups
See also

Property: IncludeAttributeGroups as Boolean

Description
Set this property to true, to include attribute groups in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
  2900  The object is no longer valid.
  2901  Invalid address for the return parameter was specified.

IncludeComplexTypes
See also

Property: IncludeComplexTypes as Boolean

Description
Set this property to `true`, to include complex types in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
- 2900 The object is no longer valid.
- 2901 Invalid address for the return parameter was specified.

**IncludeGlobalAttributes**

See also

*Property: IncludeGlobalAttributes as Boolean*

**Description**
Set this property to `true`, to include global attributes in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
- 2900 The object is no longer valid.
- 2901 Invalid address for the return parameter was specified.

**IncludeGlobalElements**

See also

*Property: IncludeGlobalElements as Boolean*

**Description**
Set this property to `true`, to include global elements in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
- 2900 The object is no longer valid.
- 2901 Invalid address for the return parameter was specified.

**IncludeGroups**

See also

*Property: IncludeGroups as Boolean*

**Description**
Set this property to `true`, to include groups in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
- 2900 The object is no longer valid.
2901  Invalid address for the return parameter was specified.

**IncludeIndex**

See also

*Property:* **IncludeIndex** *as Boolean*

*Description*
Set this property to `true`, to include an index in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is `true`.

*Errors*

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2900</td>
<td>The object is no longer valid.</td>
</tr>
<tr>
<td>2901</td>
<td>Invalid address for the return parameter was specified.</td>
</tr>
</tbody>
</table>

**IncludeLocalAttributes**

See also

*Property:* **IncludeLocalAttributes** *as Boolean*

*Description*
Set this property to `true`, to include local attributes in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is `true`.

*Errors*

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2900</td>
<td>The object is no longer valid.</td>
</tr>
<tr>
<td>2901</td>
<td>Invalid address for the return parameter was specified.</td>
</tr>
</tbody>
</table>

**IncludeLocalElements**

See also

*Property:* **IncludeLocalElements** *as Boolean*

*Description*
Set this property to `true`, to include local elements in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is `true`.

*Errors*

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2900</td>
<td>The object is no longer valid.</td>
</tr>
<tr>
<td>2901</td>
<td>Invalid address for the return parameter was specified.</td>
</tr>
</tbody>
</table>

**IncludeRedefines**

See also
Property: IncludeRedefines as Boolean

Description
Set this property to true, to include redefines in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeReferencedSchemas

See also

Property: IncludeReferencedSchemas as Boolean

Description
Set this property to true, to include referenced schemas in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeSimpleTypes

See also

Property: IncludeSimpleTypes as Boolean

Description
Set this property to true, to include simple types in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is false.
Errors
2900  The object is no longer valid.
2901  Invalid value has been used to set the property.
       Invalid address for the return parameter was specified.

**OptionsDialogAction**

See also

**Property:** OptionsDialogAction as SPYDialogAction

**Description**
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value `spyDialogUserInput(2)`. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use `spyDialogOK(0)`. Default is `spyDialogOK`.

Errors
2900  The object is no longer valid.
2901  Invalid value has been used to set the property.
       Invalid address for the return parameter was specified.

**OutputFile**

See also

**Property:** OutputFile as String

**Description**
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to Document.GenerateSchemaDocumentation.

Errors
2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.

**OutputFileDialogAction**

See also

**Property:** OutputFileDialogAction as SPYDialogAction

**Description**
To allow the user to select the output file with a file selection dialog, set this property to `spyDialogUserInput(2)`.
If the value stored in OutputFile should be taken and no user interaction should occur, use `spyDialogOK(0)`. Default is `spyDialogOK`.

Errors
2900  The object is no longer valid.
2901  Invalid value has been used to set the property.  
      Invalid address for the return parameter was specified.

**OutputFormat**

**See also**

**Property:** `OutputFormat` as `SPYSchemaDocumentationFormat`

**Description**
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is HTML.

**Errors**

2900  The object is no longer valid.
2901  Invalid value has been used to set the property.  
      Invalid address for the return parameter was specified.

**Parent**

**See also**

**Property:** `Parent` as `Dialogs` (read-only)

**Description**
Access the parent of the object.

**Errors**

2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.

**ShowAnnotations**

**See also**

**Property:** `ShowAnnotations` as `Boolean`

**Description**
Set this property to `true`, to show the annotations to a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

**Errors**

2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.

**ShowAttributes**

**See also**

**Property:** `ShowAttributes` as `Boolean`
**Description**
Set this property to **true**, to show the type definitions attributes in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

**Errors**
- 2900  The object is no longer valid.
- 2901  Invalid address for the return parameter was specified.

**ShowChildren**

**See also**

**Property:** ShowChildren as Boolean

**Description**
Set this property to **true**, to show the children of a type definition as links in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

**Errors**
- 2900  The object is no longer valid.
- 2901  Invalid address for the return parameter was specified.

**ShowDiagram**

**See also**

**Property:** ShowDiagram as Boolean

**Description**
Set this property to **true**, to show type definitions as diagrams in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

**Errors**
- 2900  The object is no longer valid.
- 2901  Invalid address for the return parameter was specified.

**ShowEnumerations**

**See also**

**Property:** ShowEnumerations as Boolean

**Description**
Set this property to **true**, to show the enumerations contained in a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.
Errors
   2900  The object is no longer valid.
   2901  Invalid address for the return parameter was specified.

ShowIdentityConstraints
See also

Property: ShowIdentityConstraints as Boolean

Description
Set this property to true, to show a type definitions identity constraints in the schema
documentation. The property is initialized with the value used during the last call to
Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
   2900  The object is no longer valid.
   2901  Invalid address for the return parameter was specified.

ShowNamespace
See also

Property: ShowNamespace as Boolean

Description
Set this property to true, to show the namespace of type definitions in the schema
documentation. The property is initialized with the value used during the last call to
Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
   2900  The object is no longer valid.
   2901  Invalid address for the return parameter was specified.

ShowPatterns
See also

Property: ShowPatterns as Boolean

Description
Set this property to true, to show the patterns of a type definition in the schema documentation.
The property is initialized with the value used during the last call to
Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
   2900  The object is no longer valid.
   2901  Invalid address for the return parameter was specified.
ShowProgressBar

See also

Property: `ShowProgressBar` as Boolean

Description
Set this property to `true`, to make the window showing the document generation progress visible. Use `false`, to hide it. Default is `false`.

Errors
2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.

ShowProperties

See also

Property: `ShowProperties` as Boolean

Description
Set this property to `true`, to show the type definition properties in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.

ShowResult

See also

Property: `ShowResult` as Boolean

Description
Set this property to `true`, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
2900  The object is no longer valid.
2901  Invalid address for the return parameter was specified.
ShowSingleFacets

See also

Property: ShowSingleFacets as Boolean

Description
Set this property to `true`, to show the facets of a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
- 2900  The object is no longer valid.
- 2901  Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to `true`, to show the XML source code for type definitions in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
- 2900  The object is no longer valid.
- 2901  Invalid address for the return parameter was specified.

ShowType

See also

Property: ShowType as Boolean

Description
Set this property to `true`, to show the type of type definitions in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
- 2900  The object is no longer valid.
- 2901  Invalid address for the return parameter was specified.

ShowUsedBy

See also
Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

SPSFile
See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

UseFixedDesign
See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.
3.2.24 SpyProject

See also

Methods
CloseProject
SaveProject
SaveProjectAs

Properties
RootItems
ProjectFile

Description
SpyProject Class

CloseProject

See also

Declaration: CloseProject (bDiscardChanges as Boolean, bCloseFiles as Boolean, bDialog as Boolean)

Parameters
bDiscardChanges
Set bDiscardChanges to FALSE if you want to save the changes of the open project files and the project.

bCloseFiles
Set bCloseFiles to TRUE to close all open project files.

bDialog
Show dialogs for user input.

Description
CloseProject closes the current project.

ProjectFile

See also

Declaration: ProjectFile as String

Description
Path and filename of the project.

RootItems

See also

Declaration: RootItems as SpyProjectItems
Description
Root level of collection of project items.

SaveProject
See also

Declaration: SaveProject

Description
SaveProject saves the current project.

SaveProjectAs
See also

Declaration: SaveProjectAs (strPath as String, bDialog as Boolean)

Parameters
strPath
Full path with file name of new project file.

bDialog
If bDialog is TRUE, a file-dialog will be displayed.

Description
SaveProjectAs stores the project data into a new location.
3.2.25 SpyProjectItem

See also

Methods
Open

Properties
ChildItems
ParentItem
FileExtensions
ItemType
Name
Path
ValidateWith
XMLForXSLTransformation
XSLForXMLTransformation
XSLTransformationFileExtension
XSLTransformationFolder

Description
SpyProjectItem Class

ChildItems
See also

Declaration: ChildItems as SpyProjectItems

Description
If the item is a folder, ChildItems is the collection of the folder content.

FileExtensions
See also

Declaration: FileExtensions as String

Description
Used to set the file extensions if the project item is a folder.

ItemType
See also

Declaration: ItemType as SPYProjectItemTypes

Description
This property is read-only.
Name
See also

Declaration: Name as String

Description
Name of the project item. This property is read-only.

Open
See also

Declaration: Open as Document

Return Value
The project item opened as document.

Description
Opens the project item.

ParentItem
See also

Declaration: ParentItem as SpyProjectItem

Description
Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

Path
See also

Declaration: Path as String

Description
Path of project item. This property is read-only.

ValidateWith
See also

Declaration: ValidateWith as String

Description
Used to set the schema/DTD for validation.
XMLForXSLTransformation

See also

Declaration: XMLForXSLTransformation as String

Description
Used to set the XML for XSL transformation.

XSLForXMLTransformation

See also

Declaration: XSLForXMLTransformation as String

Description
Used to set the XSL for XML transformation.

XSLTransformationFileExtension

See also

Declaration: XSLTransformationFileExtension as String

Description
Used to set the file extension for XSL transformation output files.

XSLTransformationFolder

See also

Declaration: XSLTransformationFolder as String

Description
Used to set the destination folder for XSL transformation output files.
3.2.26 SpyProjectItems

See also

Methods
AddFile
AddFolder
AddURL
RemoveItem

Properties
Count
Item

Description
SpyProjectItems Class

AddFile
See also

Declaration: AddFile (strPath as String)

Parameters
strPath
Full path with file name of new project item

Description
The method adds a new file to the collection of project items.

AddFolder
See also

Declaration: AddFolder (strName as String)

Parameters
strName
Name of the new folder.

Description
The method AddFolder adds a folder with the name strName to the collection of project items.

AddURL
See also

Declaration: AddURL (strURL as String, nURLType as SPYURLTypes, strUser as String, strPassword as String, bSave as Boolean)

Description
strURL
URL to open as document.

nURLType
Type of document to open. Set to \(-1\) for auto detection.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentication. Can be empty.

bSave
Save user and password information.

Description
The method adds an URL item to the project collection.

Count
See also

*Declaration:* Count as long

Description
This property gets the count of project items in the collection. The property is read-only.

Item
See also

*Declaration:* Item \((n\) as long) as SpyProjectItem

Description
Retrieves the \(n\)-th element of the collection of project items. The first item has index 1.

RemoveItem
See also

*Declaration:* RemoveItem \((pItem\) as SpyProjectItem)

Description
RemoveItem deletes the item pItem from the collection of project items.
3.2.27 TextImportExportSettings

See also

Properties for import only
ImportFile

Properties for export only
DestinationFolder
FileExtension
CommentIncluded
RemoveDelimiter
RemoveNewline

Properties for import and export
HeaderRow
FieldDelimiter
EnclosingCharacter
Encoding
EncodingByteOrder

Description
TextImportExportSettings contains options common to text import and export functions.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated text file. Default is true. This property is used only when exporting to text files.

DestinationFolder

See also

Property: DestinationFolder as String

Description
The property DestinationFolder sets the folder where the created files are saved during text export.

EnclosingCharacter

See also

Property: EnclosingCharacter as SPYTextEnclosing

Description
This property defines the character that encloses all field values for import and export. Default is spyNoEnclosing.
Encoding
See also

Property: Encoding as String

Description
The property Encoding sets the character encoding for the text files for importing and exporting.

EncodingByteOrder
See also

Property: EncodingByteOrder as SPYEncodingByteOrder

Description
The property EncodingByteOrder sets the byte order for Unicode characters. Default is spyNONE.

FieldDelimiter
See also

Property: FieldDelimiter as SPYTextDelimiters

Description
The property FieldDelimiter defines the delimiter between the fields during import and export. Default is spyTabulator.

FileExtension
See also

Property: FileExtension as String

Description
This property sets the file extension for files created on text export.

HeaderRow
See also

Property: HeaderRow as Boolean

Description
The property HeaderRow is used during import and export. Set HeaderRow true on import, if the first line of the text file contains the names of the columns. Set HeaderRow true on export, if the first line in the created text files should contain the name of the columns. Default value is true.
ImportFile
See also

Property: ImportFile as String

Description
This property is used to set the text file for import. The string has to be a full qualified path.

RemoveDelimiter
See also

Property: RemoveDelimiter as Boolean

Description
The property RemoveDelimiter defines whether characters in the text that are equal to the delimiter character are removed. Default is false. This property is used only when exporting to text files.

RemoveNewline
See also

Property: RemoveNewline as Boolean

Description
The property RemoveNewline defines whether newline characters in the text are removed. Default is false. This property is used only when exporting to text files.
3.2.28 TextView

See also

Properties and Methods

- Application
- Parent
- LineFromPosition
- PositionFromLine
- LineLength
- SelText
- GetRangeText
- ReplaceText
- MoveCaret
- GoToLineChar
- SelectText
- SelectionStart
- SelectionEnd
- Text
- LineCount
- Length

Description

Events

OnBeforeShowSuggestions

See also

Event: OnBeforeShowSuggestions() as Boolean

Description
This event gets fired before a suggestion window is shown. The Document property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_BeforeShowSuggestions()
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeShowSuggestions()
{
}
```
**OnChar**

See also

**Event: OnChar** (nChar as Long, bExistSuggestions as Boolean) as Boolean

**Description**

This event gets fired on each key stroke. The parameter nChar is the key that was pressed and bExistSuggestions tells whether a Authentic Desktop generated suggestions window is displayed after this key. The Document property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.

It is also possible to create a new suggestions window when none is provided by Authentic Desktop. Set the Document property Suggestions to a string array with your recommendations and return true.

This event is fired before the OnBeforeShowSuggestions event. If you prevent to show the suggestion window by returning false then OnBeforeShowSuggestions is not fired.

**Examples**

Given below are examples of how this event can be scripted.

**XMLSpy scripting environment - VBScript:**

```vbnet
Function On_Char(nChar, bExistSuggestions)
End Function
```

**XMLSpy scripting environment - JScript:**

```javascript
function On_Char(nChar, bExistSuggestions)
{
}
```

**XMLSpy IDE Plugin:**

```csharp
IXMLSpyPlugIn.OnEvent (35, ...) // nEventId = 35
```

**Application**

**Property: Application as Application** (read-only)

**Description**

Access the Authentic Desktop application object.

**Errors**

- **3900** The object is no longer valid.
- **3901** Invalid address for the return parameter was specified.
GetRangeText

Method: GetRangeText(nStart as Long, nEnd as Long) as String

Description
Returns the text in the specified range.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

GoToLineChar

Method: GoToLineChar(nLine as Long, nChar as Long)

Description
Moves the caret to the specified line and character position.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

Length

Property: Length as Long

Description
Returns the character count of the document.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

LineCount

Property: LineCount as Long

Description
Returns the number of lines in the document.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

LineFromPosition

Method: LineFromPosition(nCharPos as Long) as Long

Description
Returns the line number of the character position.
Errors
   3900  The object is no longer valid.
   3901  Invalid address for the return parameter was specified.

**LineLength**

*Method:* `LineLength(nLine as Long) as Long`

*Description*
Returns the length of the line.

Errors
   3900  The object is no longer valid.
   3901  Invalid address for the return parameter was specified.

**MoveCaret**

*Method:* `MoveCaret(nDiff as Long)`

*Description*
Moves the caret `nDiff` characters.

Errors
   3900  The object is no longer valid.
   3901  Invalid address for the return parameter was specified.

**Parent**

*Property:* `Parent as Document` (read-only)

*Description*
Access the parent of the object.

Errors
   3900  The object is no longer valid.
   3901  Invalid address for the return parameter was specified.

**PositionFromLine**

*Method:* `PositionFromLine(nLine as Long) as Long`

*Description*
Returns the start position of the line.

Errors
   3900  The object is no longer valid.
   3901  Invalid address for the return parameter was specified.

**ReplaceText**

*Method:* `ReplaceText(nPosFrom as Long, nPosTill as Long, sText as String)`

*Description*
Replaces the text in the specified range.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.

SelectionEnd

Property: SelectionEnd as Long

Description
Returns/sets the text selection end position.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.

SelectionStart

Property: SelectionStart as Long

Description
Returns/sets the text selection start position.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.

SelectText

Method: SelectText(nPosFrom as Long, nPosTill as Long)

Description
Selects the text in the specified range.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.

SelText

Property: SelText as String

Description
Returns/sets the selected text.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.
Text

Property: Text as String

Description
Returns/sets the document text.

Errors
- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.
3.2.29  WSDLDocumentationDlg

See also

Properties and Methods

Standard automation properties
- Application
- Parent

Interaction and visibility properties
- GlobalElementsAndTypesOnly
- OptionsDialogAction
- OutputFile
- OutputFileDialogAction
- SeparateSchemaDocument
- ShowProgressBar
- ShowResult

Document generation options and methods
- OutputFormat
- UseFixedDesign
- SPSFile
- EmbedDiagrams
- DiagramFormat
- MultipleOutputFiles
- EmbedCSSInHTML
- CreateDiagramsFolder
- IncludeAll
- IncludeBinding
- IncludeImportedWSDLFiles
- IncludeMessages
- IncludeOverview
- IncludePortType
- IncludeService
- IncludeTypes
- AllDetails
- ShowBindingDiagram
- ShowExtensibility
- ShowMessageParts
- ShowPort
- ShowPortTypeDiagram
- ShowPortTypeOperations
- ShowServiceDiagram
- ShowSourceCode
- ShowTypesDiagram
- ShowUsedBy

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of
user interaction desired. You can use `IncludeAll` and `AllDetails` to set whole option groups at once or the individual properties to operate on a finer granularity.

**AllDetails**

See also

Method: `AllDetails (i_bDetailsOn as Boolean)`

Description
Use this method to turn all details options on or off.

Errors
- 4300 The object is no longer valid.

**Application**

See also

Property: `Application as Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.

**CreateDiagramsFolder**

See also

Property: `CreateDiagramsFolder as Boolean`

Description
Set this property to `true`, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to `Document.GenerateWSDLDocumentation`. The default for the first run is false.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.
DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is PNG.

Errors

- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors

- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors

- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.
GlobalElementsAndTypesOnly
See also

Property: GlobalElementsAndTypesOnly as Boolean

Description
Returns/sets a value indicating whether a full Schema documentation is done or only Global Elements and Types are documented.

Errors
  3900  The object is no longer valid.
  3901  Invalid address for the return parameter was specified.

IncludeAll
See also

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
  4300  The object is no longer valid.

IncludeBinding
See also

Property: IncludeBinding as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
  3900  The object is no longer valid.
  3901  Invalid address for the return parameter was specified.

IncludeImportedWSDLFiles
See also

Property: IncludeImportedWSDLFiles as Boolean

Description
Set this property to true, to include imported WSDL files in the WSDL documentation. The
property is initialized with the value used during the last call to
Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

IncludeMessages
See also

Property: IncludeMessages as Boolean

Description
Set this property to true, to include messages in the WSDL documentation. The property is
initialized with the value used during the last call to
Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

IncludeOverview
See also

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the WSDL documentation. The property is
initialized with the value used during the last call to
Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

IncludePortType
See also

Property: IncludePortType as Boolean

Description
Set this property to true, to include port types in the WSDL documentation. The property is
initialized with the value used during the last call to
Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.
IncludeService
See also

Property: IncludeService as Boolean

Description
Set this property to true, to include services in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.

IncludeTypes
See also

Property: IncludeTypes as Boolean

Description
Set this property to true, to include types in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.

MultipleOutputFiles
See also

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is false.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid value has been used to set the property.

OptionsDialogAction
See also

Property: OptionsDialogAction as SPYDialogAction
Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value `spyDialogUserInput(2)`. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use `spyDialogOK(0)`. Default is `spyDialogOK`.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid value has been used to set the property.
  Invalid address for the return parameter was specified.

OutputFile
See also

Property: `OutputFile` as `String`

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional `.png` files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to `Document.GenerateWSDLDocumentation`.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.

OutputFileDialogAction
See also

Property: `OutputFileDialogAction` as `SPYDialogAction`

Description
To allow the user to select the output file with a file selection dialog, set this property to `spyDialogUserInput(2)`. If the value stored in `OutputFile` should be taken and no user interaction should occur, use `spyDialogOK(0)`. Default is `spyDialogOK`.

Errors
- 3900 The object is no longer valid.
- 3901 Invalid value has been used to set the property.
  Invalid address for the return parameter was specified.

OutputFormat
See also

Property: `OutputFormat` as `SPYSchemaDocumentationFormat`

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to
**Document.GenerateWSDLDocumentation.** The default for the first run is HTML.

**Errors**
- 3900 The object is no longer valid.
- 3901 Invalid value has been used to set the property.
  Invalid address for the return parameter was specified.

**Parent**

See also

**Property:** Parent as Dialogs (read-only)

**Description**
Access the parent of the object.

**Errors**
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.

**SeparateSchemaDocument**

See also

**Property:** SeparateSchemaDocument as Boolean

**Description**
Returns/sets a value indicating whether the Schema documentation should be placed in a separate document.

**Errors**
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.

**ShowBindingDiagram**

See also

**Property:** ShowBindingDiagram as Boolean

**Description**
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

**Errors**
- 3900 The object is no longer valid.
- 3901 Invalid address for the return parameter was specified.
ShowExtensibility

See also

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.

ShowMessageParts

See also

Property: ShowMessageParts as Boolean

Description
Set this property to true, to show message parts of messages in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.

ShowPort

See also

Property: ShowPort as Boolean

Description
Set this property to true, to show service ports in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
   3900   The object is no longer valid.
   3901   Invalid address for the return parameter was specified.

ShowPortTypeDiagram

See also
Property: ShowPortTypeDiagram as Boolean

Description
Set this property to true, to show port type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPortTypeOperations
See also

Property: ShowPortTypeOperations as Boolean

Description
Set this property to true, to show port type operations in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowProgressBar
See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowResult
See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during
the last call to `Document.GenerateWSDLDocumentation`. The default for the first run is true.

**Errors**
- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

### ShowServiceDiagram

**See also**

**Property:** `ShowServiceDiagram` as Boolean

**Description**
Set this property to `true`, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDLDocumentation`. The default for the first run is true.

**Errors**
- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

### ShowSourceCode

**See also**

**Property:** `ShowSourceCode` as Boolean

**Description**
Set this property to `true`, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDLDocumentation`. The default for the first run is true.

**Errors**
- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

### ShowTypesDiagram

**See also**

**Property:** `ShowTypesDiagram` as Boolean

**Description**
Set this property to `true`, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDLDocumentation`. The default for the first run is true.

**Errors**
- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.
**ShowUsedBy**

See also

*Property: ShowUsedBy as Boolean*

**Description**
Set this property to `true`, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDLDocumentation`. The default for the first run is true.

**Errors**

- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

**UseFixedDesign**

See also

*Property: UseFixedDesign as Boolean*

**Description**
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

**Errors**

- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

**SPSFile**

See also

*Property: SPSFile as String*

**Description**
Full path and name of the SPS file that will be used to generate the documentation.

**Errors**

- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.
3.2.30 WSDL20DocumentationDlg

See also

Properties and Methods

Standard automation properties
- Application
- Parent

Interaction and visibility properties
- GlobalElementsAndTypesOnly
- OptionsDialogAction
- OutputFile
- OutputFileDialogAction
- SeparateSchemaDocument
- ShowProgressBar
- ShowResult

Document generation options and methods
- OutputFormat
- UseFixedDesign
- SPSFile
- EmbedDiagrams
- DiagramFormat
- MultipleOutputFiles
- EmbedCSSInHTML
- CreateDiagramsFolder
- IncludeAll
- IncludeBinding
- IncludeImportedWSDLFiles
- IncludeInterface
- IncludeOverview
- IncludeService
- IncludeTypes
- AllDetails
- ShowBindingDiagram
- ShowExtensibility
- ShowEndpoint
- ShowFault
- ShowInterfaceDiagram
- ShowOperation
- ShowServiceDiagram
- ShowSourceCode
- ShowTypesDiagram
- ShowUsedBy

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at
once or the individual properties to operate on a finer granularity.

**AllDetails**

See also

*Method: AllDetails*(i_bDetailsOn as Boolean)*

*Description*
Use this method to turn all details options on or off.

*Errors*
4300 The object is no longer valid.

**Application**

See also

*Property: Application as Application*(read-only)*

*Description*
Access the Authentic Desktop application object.

*Errors*
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

**CreateDiagramsFolder**

See also

*Property: CreateDiagramsFolder as Boolean*

*Description*
Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to Document.GenerateWSD20LDocumentation. The default for the first run is false.

*Errors*
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

**DiagramFormat**

See also
Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is PNG.

Errors
- 4300  The object is no longer valid.
- 4301  Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
- 4300  The object is no longer valid.
- 4301  Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
- 4300  The object is no longer valid.
- 4301  Invalid address for the return parameter was specified.

GlobalElementsAndTypesOnly

See also

Property: GlobalElementsAndTypesOnly as Boolean

Description
Returns/sets a value indicating whether a full Schema documentation is done or only Global Elements and Types are documented.

Errors
3900  The object is no longer valid.
3901  Invalid address for the return parameter was specified.

IncludeAll
See also

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4300  The object is no longer valid.

IncludeImportedWSDLFiles
See also

Property: IncludeImportedWSDLFiles as Boolean

Description
Set this property to true, to include imported WSDL files in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
4300  The object is no longer valid.
4301  Invalid address for the return parameter was specified.

IncludeBinding
See also

Property: IncludeBinding as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
4300  The object is no longer valid.
4301  Invalid address for the return parameter was specified.
**IncludeInterface**

**See also**

*Property:* `IncludeInterface` as `Boolean`

**Description**

Set this property to `true`, to include interfaces in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

**Errors**

- 4300 The object is no longer valid.
- 4301 Invalid address for the return parameter was specified.

**IncludeOverview**

**See also**

*Property:* `IncludeOverview` as `Boolean`

**Description**

Set this property to `true`, to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

**Errors**

- 4300 The object is no longer valid.
- 4301 Invalid address for the return parameter was specified.

**IncludeService**

**See also**

*Property:* `IncludeService` as `Boolean`

**Description**

Set this property to `true`, to include services in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

**Errors**

- 4300 The object is no longer valid.
- 4301 Invalid address for the return parameter was specified.

**IncludeTypes**

**See also**

*Property:* `IncludeTypes` as `Boolean`

**Description**
Set this property to `true`, to include types in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

**Errors**
- 4300  The object is no longer valid.
- 4301  Invalid address for the return parameter was specified.

**MultipleOutputFiles**

See also

*Property:* `MultipleOutputFiles` as Boolean

**Description**
Set this property to `true`, to split the documentation files. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is false.

**Errors**
- 4300  The object is no longer valid.
- 4301  Invalid value has been used to set the property. Invalid address for the return parameter was specified.

**OptionsDialogAction**

See also

*Property:* `OptionsDialogAction` as `SPYDialogAction`

**Description**
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value `spyDialogUserInput(2)`. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use `spyDialogOK(0)`. Default is `spyDialogOK`.

**Errors**
- 4300  The object is no longer valid.
- 4301  Invalid value has been used to set the property. Invalid address for the return parameter was specified.

**OutputFile**

See also

*Property:* `OutputFile` as String

**Description**
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to `Document.GenerateWSDL20Documentation`. 
Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

OutputFileDialogAction
See also

Property: OutputFileDialogAction as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to spyDialogUserInput(2). If the value stored in OutputFile should be taken and no user interaction should occur, use spyDialogOK(0). Default is spyDialogOK.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat
See also

Property: OutputFormat as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is HTML.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent
See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.
SeparateSchemaDocument
See also

**Property:** SeparateSchemaDocument as Boolean

**Description**
Returns/sets a value indicating whether the Schema documentation should be placed in a separate document.

**Errors**
- 3900  The object is no longer valid.
- 3901  Invalid address for the return parameter was specified.

ShowBindingDiagram
See also

**Property:** ShowBindingDiagram as Boolean

**Description**
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

**Errors**
- 4300  The object is no longer valid.
- 4301  Invalid address for the return parameter was specified.

ShowEndpoint
See also

**Property:** ShowEndpoint as Boolean

**Description**
Set this property to true, to show service endpoints in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

**Errors**
- 4300  The object is no longer valid.
- 4301  Invalid address for the return parameter was specified.

ShowExtensibility
See also

**Property:** ShowExtensibility as Boolean
Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

Errors
   4300 The object is no longer valid.
   4301 Invalid address for the return parameter was specified.

ShowFault
See also

Property: ShowFault as Boolean

Description
Set this property to true, to show faults in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

Errors
   4300 The object is no longer valid.
   4301 Invalid address for the return parameter was specified.

ShowInterfaceDiagram
See also

Property: ShowInterfaceDiagram as Boolean

Description
Set this property to true, to show interface diagrams in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

Errors
   4300 The object is no longer valid.
   4301 Invalid address for the return parameter was specified.

ShowOperation
See also

Property: ShowOperation as Boolean

Description
Set this property to true, to show interface and binding operations in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.
Errors
  4300  The object is no longer valid.
  4301  Invalid address for the return parameter was specified.

ShowProgressBar
See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
  4300  The object is no longer valid.
  4301  Invalid address for the return parameter was specified.

ShowResult
See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
  4300  The object is no longer valid.
  4301  Invalid address for the return parameter was specified.

ShowServiceDiagram
See also

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
  4300  The object is no longer valid.
4301 Invalid address for the return parameter was specified.

**ShowSourceCode**

See also

*Property:* `ShowSourceCode` as `Boolean`

**Description**

Set this property to `true`, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

**Errors**

- 4300 The object is no longer valid.
- 4301 Invalid address for the return parameter was specified.

**ShowTypesDiagram**

See also

*Property:* `ShowTypesDiagram` as `Boolean`

**Description**

Set this property to `true`, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

**Errors**

- 4300 The object is no longer valid.
- 4301 Invalid address for the return parameter was specified.

**ShowUsedBy**

See also

*Property:* `ShowUsedBy` as `Boolean`

**Description**

Set this property to `true`, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to `Document.GenerateWSDL20Documentation`. The default for the first run is true.

**Errors**

- 4300 The object is no longer valid.
- 4301 Invalid address for the return parameter was specified.
SPSFile
See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
  4300   The object is no longer valid.
  4301   Invalid address for the return parameter was specified.

UseFixedDesign
See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
  4300   The object is no longer valid.
  4301   Invalid address for the return parameter was specified.
3.2.31 XBRLDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OptionsDialogAction
OutputFile
OutputFileDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
EmbedCSSInHTML
CreateDiagramsFolder

IncludeAll
IncludeOverview
IncludeNamespacePrefixes
IncludeGlobalElements
IncludeDefinitionLinkroles
IncludePresentationLinkroles
IncludeCalculationLinkroles

AllDetails
ShowDiagram
ShowSubstitutiongroup
ShowItemType
ShowBalance
ShowPeriod
ShowAbstract
ShowNillable
ShowLabels
ShowReferences
ShowLinkbaseReferences

ShortQualified Name
ShowImportedElements

Description
This object combines all options for XBRL document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are initialized with the values used during the last generation of XBRL documentation. However, before using the
object you have to set the OutputFile property to a valid file path. Use
OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify
the level of user interaction desired. You can use IncludeAll and AllDetails to set whole
option groups at once or the individual properties to operate on a finer granularity.

AllDetails
See also

*Method:* AllDetails *(i_bDetailsOn as Boolean)*

*Description*
Use this method to turn all details options on or off.

*Errors*
  - 4400 The object is no longer valid.

Application
See also

*Property:* Application as Application (read-only)

*Description*
Access the Authentic Desktop application object.

*Errors*
  - 4400 The object is no longer valid.
  - 4401 Invalid address for the return parameter was specified.

CreateDiagramsFolder
See also

*Property:* CreateDiagramsFolder as Boolean

*Description*
Set this property to true, to create a directory for the created images. Otherwise the diagrams
will be created next to the documentation. This property is only available when the diagrams are
not embedded. The property is initialized with the value used during the last call to
Document.GenerateXBRLDocumentation. The default for the first run is false.

*Errors*
  - 4400 The object is no longer valid.
  - 4401 Invalid address for the return parameter was specified.
DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is PNG.

Errors
- 4400  The object is no longer valid.
- 4401  Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
- 4400  The object is no longer valid.
- 4401  Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
- 4400  The object is no longer valid.
- 4401  Invalid address for the return parameter was specified.
IncludeAll
See also

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4400  The object is no longer valid.

IncludeCalculationLinkroles
See also

Property: IncludeCalculationLinkroles as Boolean

Description
Set this property to true, to include calculation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.

IncludeDefinitionLinkroles
See also

Property: IncludeDefinitionLinkroles as Boolean

Description
Set this property to true, to include definition linkroles in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.

IncludeGlobalElements
See also

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the XBRL documentation. The property is
initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.

IncludeNamespacePrefixes
See also

Property: IncludeNamespacePrefixes as Boolean

Description
Set this property to true, to include namespace prefixes in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.

IncludeOverview
See also

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.

IncludePresentationLinkroles
See also

Property: IncludePresentationLinkroles as Boolean

Description
Set this property to true, to include presentation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.
OptionsDialogAction

See also

Property: OptionsDialogAction as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value spyDialogUserInput(2). If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use spyDialogOK(0). Default is spyDialogOK.

Errors
- 4400 The object is no longer valid.
- 4401 Invalid value has been used to set the property.
  Invalid address for the return parameter was specified.

OutputFile

See also

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to Document.GenerateXBRLDocumentation.

Errors
- 4400 The object is no longer valid.
- 4401 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: OutputFileDialogAction as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to spyDialogUserInput(2). If the value stored in OutputFile should be taken and no user interaction should occur, use spyDialogOK(0). Default is spyDialogOK.

Errors
- 4400 The object is no longer valid.
- 4401 Invalid value has been used to set the property.
  Invalid address for the return parameter was specified.
OutputFormat
See also

Property: OutputFormat as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is HTML.

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent
See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShortQualifiedName
See also

Property: ShortQualifiedName as Boolean

Description
Set this property to true, to use short qualified names in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowAbstract
See also

Property: ShowAbstract as Boolean

Description
Set this property to `true`, to show abstracts in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is `true`.

**Errors**
- **4400**: The object is no longer valid.
- **4401**: Invalid address for the return parameter was specified.

### ShowBalance

**See also**

*Property: ShowBalance as Boolean*

**Description**
Set this property to `true`, to show balances in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is `true`.

**Errors**
- **4400**: The object is no longer valid.
- **4401**: Invalid address for the return parameter was specified.

### ShowDiagram

**See also**

*Property: ShowDiagram as Boolean*

**Description**
Set this property to `true`, to show diagrams in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is `true`.

**Errors**
- **4400**: The object is no longer valid.
- **4401**: Invalid address for the return parameter was specified.

### ShowImportedElements

**See also**

*Property: ShowImportedElements as Boolean*

**Description**
Set this property to `true`, to show imported elements in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is `true`. 
Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowItemtype
See also

Property: ShowItemtype as Boolean

Description
Set this property to true, to show item types in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowLabels
See also

Property: ShowLabels as Boolean

Description
Set this property to true, to show labels in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowLinkbaseReferences
See also

Property: ShowLinkbaseReferences as Boolean

Description
Set this property to true, to show linkbase references in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.
ShowNillable

See also

**Property:** ShowNillable as Boolean

**Description**
Set this property to `true`, to show nillable properties in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is `true`.

**Errors**
- 4400  The object is no longer valid.
- 4401  Invalid address for the return parameter was specified.

ShowPeriod

See also

**Property:** ShowPeriod as Boolean

**Description**
Set this property to `true`, to show periods in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is `true`.

**Errors**
- 4400  The object is no longer valid.
- 4401  Invalid address for the return parameter was specified.

ShowProgressBar

See also

**Property:** ShowProgressBar as Boolean

**Description**
Set this property to `true`, to make the window showing the document generation progress visible. Use `false`, to hide it. Default is `false`.

**Errors**
- 4400  The object is no longer valid.
- 4401  Invalid address for the return parameter was specified.

ShowReferences

See also

**Property:** ShowReferences as Boolean
Description
Set this property to **true**, to show references in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is true.

Errors
- 4400 The object is no longer valid.
- 4401 Invalid address for the return parameter was specified.

ShowResult
See also

*Property: ShowResult as Boolean*

Description
Set this property to **true**, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is true.

Errors
- 4400 The object is no longer valid.
- 4401 Invalid address for the return parameter was specified.

ShowSubstitutiongroup
See also

*Property: ShowSubstitutiongroup as Boolean*

Description
Set this property to **true**, to show substitution groups in the XBRL documentation. The property is initialized with the value used during the last call to `Document.GenerateXBRLDocumentation`. The default for the first run is true.

Errors
- 4400 The object is no longer valid.
- 4401 Invalid address for the return parameter was specified.

SPSFile
See also

*Property: SPSFile as String*

Description
Full path and name of the SPS file that will be used to generate the documentation.
Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
4400  The object is no longer valid.
4401  Invalid address for the return parameter was specified.
3.2.32 XMLData

See also

Properties

- **Kind**
- **Name**
- **TextValue**

- **HasChildren**
- **MayHaveChildren**
- **Parent**

Methods

- **GetFirstChild**
- **GetNextChild**
- **GetCurrentChild**

- **InsertChild**
- **InsertChildAfter**
- **InsertChildBefore**
- **AppendChild**

- **EraseAllChildren**
- **EraseChild**
- **EraseCurrentChild**

- **IsSameNode**

- **CountChildren**
- **CountChildrenKind**

- **GetChild**
- **GetChildAttribute**
- **GetChildElement**
- **GetChildKind**
- **GetNamespacePrefixForURI**

- **HasChildrenKind**
- **SetTextValueXMLEncoded**

Description

The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind `spyXMLDataElement` with the **Name** property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view.
switch.

Note also:

- Setting a new text value for an XML element is possible if the element does not have non-text children. A text value can be set even if the element has attributes.
- When setting a new text value for an XML element which has more than one text child, the latter will be deleted and replaced by one new text child.
- When reading the text value of an XML element which has more than one text child, only the value of the first text child will be returned.

**AppendChild**

See also

*Declaration:* AppendChild (pNewData as XMLData)

*Description*

AppendChild appends pNewData as last child to the XMLData object.

*Errors*

- 1500 The XMLData object is no longer valid.
- 1505 Invalid XMLData kind was specified.
- 1506 Invalid address for the return parameter was specified.
- 1507 Element cannot have Children
- 1512 Cyclic insertion - new data element is already part of document
- 1514 Invalid XMLData kind was specified for this position.
- 1900 Document must not be modified

*Example*

```vbscript
Dim objCurrentParent As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

**CountChildren**

See also

*Declaration:* CountChildren as long

*Description*

CountChildren gets the number of children.

*Available with TypeLibrary version 1.5*

*Errors*
1500 The XMLData object is no longer valid.

**CountChildrenKind**

**See also**

*Declaration:* `CountChildrenKind (nKind as SPYXMLDataKind) as long`

**Description**

`CountChildrenKind` gets the number of children of the specific kind.

Available with TypeLibrary version 1.5

**Errors**

- 1500 The XMLData object is no longer valid.

**EraseAllChildren**

**See also**

*Declaration:* `EraseAllChildren`

**Description**

`EraseAllChildren` deletes all associated children of the XMLData object.

**Errors**

- 1500 The XMLData object is no longer valid.
- 1900 Document must not be modified

**Example**

The sample erases all elements of the active document.

```vba
Dim objCurrentParent As XMLData

objCurrentParent.EraseAllChildren
```

**EraseChild**

*Method:* `EraseChild (Child as XMLData)`

**Description**

Deletes the given child node.

**Errors**

- 1500 Invalid object.
- 1506 Invalid input xml
- 1510 Invalid parameter.
**EraseCurrentChild**

See also

*Declaration:* `EraseCurrentChild`

**Description**

`EraseCurrentChild` deletes the current XMLData child object. Before you call `EraseCurrentChild` you must initialize an internal iterator with `XMLData.GetFirstChild`. After deleting the current child, `EraseCurrentChild` increments the internal iterator of the XMLData element. No error is returned when the last child gets erased and the iterator is moved past the end of the child list. The next call to `EraseCurrentChild` however, will return error 1503.

**Errors**

- **1500** The XMLData object is no longer valid.
- **1503** No iterator is initialized for this XMLData object, or the iterator points past the last child.
- **1900** Document must not be modified

**Examples**

```jscript
// ---------------------------------------
// XMLSpy scripting environment - JScript
// erase all children of XMLData
// ---------------------------------------
// let's get an XMLData element, we assume that the
// cursor selects the parent of a list in grid view

// the following line would be shorter, of course
//objList.EraseAllChildren();

// but we want to demonstrate the usage of EraseCurrentChild
if ((objList != null) && (objList.HasChildren))
{
  try
  {
    objEle = objList.GetFirstChild(-1);
    while (objEle != null)
      objList.EraseCurrentChild();
    // no need to call GetNextChild
  }
  catch (err)
  // 1503 - we reached end of child list
  { if ((err.number & 0xffff) != 1503) throw (err); }
}
```

**GetChild**

See also

*Declaration:* `GetChild(position as long) as XMLData`
Return Value
Returns an XML element as XMLData object.

Description
GetChild() returns a reference to the child at the given index (zero-based).

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

GetChildAttribute

Method: GetChildAttribute (strName as string) child as XMLData object (NULL on error)

Description
Retrieves the attribute having the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

GetChildElement

Method: GetChildElement (strName as string, nIndex as long) child as XMLData object (NULL on error)

Description
Retrieves the Nth child element with the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

GetChildKind

See also

Declaration: GetChildKind (position as long, nKind as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetChildKind() returns a reference to a child of this kind at the given index (zero-based). The
position parameter is relative to the number of children of the specified kind and not to all children of the object.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

GetCurrentChild

See also

Declaration: GetCurrentChild as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetCurrentChild gets the current child. Before you call GetCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

GetFirstChild

See also

Declaration: GetFirstChild(nKind as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetFirstChild initializes a new iterator and returns the first child. Set nKind = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use GetCurrentChild, GetNextChild or EraseCurrentChild.

Errors
1500 The XMLData object is no longer valid.
1501 Invalid XMLData kind was specified.
1504 Element has no children of specified kind.
1510 Invalid address for the return parameter was specified.

Example
See the example at `XMLData.GetNextChild`.

**GetNamespacePrefixForURI**

*Method:* GetNamespacePrefixForURI (strURI as string) strNS as string

*Description*
Returns the namespace prefix of the supplied URI.

*Errors*
- 1500 invalid object.
- 1510 invalid parameter.

**GetNextChild**

*See also*

*Declaration:* GetNextChild as XMLData

*Return Value*
Returns an XML element as XMLData object.

*Description*
GetNextChild steps to the next child of this element. Before you call GetNextChild you must initialize an internal iterator with `XMLData.GetFirstChild`.

Check for the last child of the element as shown in the sample below.

*Errors*
- 1500 The XMLData object is no longer valid.
- 1503 No iterator is initialized for this XMLData object.
- 1510 Invalid address for the return parameter was specified.

*Examples*
```
' ----------------------------------------------
' VBA code snippet - iterate XMLData children
' ----------------------------------------------
On Error Resume Next
Set objParent = objSpy.ActiveDocument.RootElement

' get elements of all kinds
Set objCurrentChild = objParent.GetFirstChild(-1)
Do
  ' do something useful with the child
  ' step to next child
  Set objCurrentChild = objParent.GetNextChild
Loop Until (Err.Number - vbObjectError = 1503)
```
// XMLSpy scripting environment - JScript
// iterate through children of XMLData
// ---------------------------------------
try {
    var objXMLData = ... // initialize somehow
    var objChild = objXMLData.GetFirstChild(-1);

    while (true) {
        // do something useful with objChild
        objChild = objXMLData.GetNextChild();
    }
} catch (err) {
    if ((err.number & 0xffff) == 1504) {
        // element has no children
    } else if ((err.number & 0xffff) == 1503) {
        // last child reached
    } else {
        throw (err);
    }
}

GetTextValueXMLDecoded

Method: GetTextValueXMLDecoded () as string

Description
Gets the decoded text value of the XML.

Errors
1500 Invalid object.
1510 Invalid parameter.

HasChildren

See also

Declaration: HasChildren as Boolean

Description
The property is true if the object is the parent of other XMLData objects. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.
HasChildrenKind

See also

Declaration: HasChildrenKind \( (nKind \text{ as SPYXMLDataKind}) \text{ as Boolean} \)

Description
The method returns true if the object is the parent of other XMLData objects of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

InsertChild

See also

Declaration: InsertChild \( (pNewData \text{ as XMLData}) \)

Description
InsertChild inserts the new child before the current child (see also XMLData.GetFirstChild, XMLData.GetNextChild to set the current child).

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

InsertChildAfter

Method: InsertChildBefore \( (Node \text{ as XMLData, NewData as XMLData}) \)

Description
Inserts a new XML node (supplied with the second parameter) after the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position
InsertChildBefore

**Method:** InsertChildBefore (Node as XMLData, NewData as XMLData)

**Description**
Inserts a new XML node (supplied with the second parameter) before the specified node (first parameter).

**Errors**
- 1500 Invalid object.
- 1506 Invalid input xml
- 1507 No children allowed
- 1510 Invalid parameter.
- 1512 Child is already added
- 1514 Invalid kind at position

IsSameNode

**See also**

**Declaration:** IsSameNode (pNodeToCompare as XMLData) as Boolean

**Description**
Returns true if pNodeToCompare references the same node as the object itself.

**Errors**
- 1500 The XMLData object is no longer valid.
- 1506 Invalid address for the return parameter was specified.

Kind

**See also**

**Declaration:** Kind as SPYXMLDataKind

**Description**
Kind of this XMLData object. This property is read-only.

**Errors**
- 1500 The XMLData object is no longer valid.
- 1510 Invalid address for the return parameter was specified.

MayHaveChildren

**See also**
**Declaration:** MayHaveChildren as Boolean

**Description**
Indicates whether it is allowed to add children to this XMLData object. This property is read-only.

**Errors**
- 1500 The XMLData object is no longer valid.
- 1510 Invalid address for the return parameter was specified.

**Name**

**See also**

**Declaration:** Name as String

**Description**
Used to modify and to get the name of the XMLData object.

**Errors**
- 1500 The XMLData object is no longer valid.
- 1510 Invalid address for the return parameter was specified.

**Parent**

**See also**

**Declaration:** Parent as XMLData

**Return value**
Parent as XMLData object. Nothing (or NULL) if there is no parent element.

**Description**
Parent of this element. This property is read-only.

**Errors**
- 1500 The XMLData object is no longer valid.
- 1510 Invalid address for the return parameter was specified.

**SetTextValueXMLEncoded**

**Method:** SetTextValueXMLEncoded (strVal as String)

**Description**
Sets the encoded text value of the XML.

**Errors**
- 1500 Invalid object.
- 1513 Modification not allowed.
**TextValue**

*See also*

*Declaration:* `TextValue` as String

*Description*

Used to modify and to get the text value of this `XMLData` object.

*Errors*

- **1500**  The XMLData object is no longer valid.
- **1510**  Invalid address for the return parameter was specified.
3.3 Enumerations

This is a list of all enumerations used by the Authentic Desktop API. If your scripting environment does not support enumerations use the number-values instead.
3.3.1 ENUMApplicationStatus

Description
Enumeration to specify the current Application status.

Possible values:

- eApplicationRunning = 0
- eApplicationAfterLicenseCheck = 1
- eApplicationBeforeLicenseCheck = 2
- eApplicationConcurrentLicenseCheckFailed = 3
- eApplicationProcessingCommandLine = 4
3.3.2 SPYAttributeTypeDefininition

Description
Attribute type definition that can be selected for generation of Sample XML.
This type is used with the method GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:

- spyMergedGlobal = 0
- spyDistinctGlobal = 1
- spyLocal = 2
3.3.3 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange objects.

Possible values:

spyAuthenticInsertAt = 0
spyAuthenticApply = 1
spyAuthenticClearSurrounding = 2
spyAuthenticAppend = 3
spyAuthenticInsertBefore = 4
spyAuthenticRemove = 5
3.3.4 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange objects.

Possible values:
spyAuthenticDocumentBegin = 0
spyAuthenticDocumentEnd = 1
spyAuthenticRangeBegin = 2
spyAuthenticRangeEnd = 3
3.3.5 SPYAuthenticElementActions

Description
Actions that can be used with the obsolete object GetAllowedElements (superseded by AuthenticRange.CanPerformActionWith).

Possible values:
- k_ActionInsertAt = 0
- k_ActionApply = 1
- k_ActionClearSurr = 2
- k_ActionAppend = 3
- k_ActionInsertBefore = 4
- k_ActionRemove = 5
3.3.6 **SPYAuthenticElementKind**

**Description**
Enumeration of the different kinds of elements used for navigation and selection within the `AuthenticRange` and `AuthenticView` objects.

**Possible values:**
- `spyAuthenticChar` = 0
- `spyAuthenticWord` = 1
- `spyAuthenticLine` = 3
- `spyAuthenticParagraph` = 4
- `spyAuthenticTag` = 6
- `spyAuthenticDocument` = 8
- `spyAuthenticTable` = 9
- `spyAuthenticTableRow` = 10
- `spyAuthenticTableColumn` = 11
3.3.7 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility.

Possible values:
- spyAuthenticMarkupHidden = 0
- spyAuthenticMarkupSmall = 1
- spyAuthenticMarkupLarge = 2
- spyAuthenticMarkupMixed = 3
3.3.8  **SPYAuthenticToolBarButtonState**

**Description**

Authentic toolbar button states are given by the following enumeration:

**Possible values:**

- `authenticToolBarButtonDefault` = 0
- `authenticToolBarButtonEnabled` = 1
- `authenticToolBarButtonDisabled` = 2
3.3.9 **SPYDatabaseKind**

**Description**
Values to select different kinds of databases for import. See
[DatabaseConnection.DatabaseKind](#) for its use.

**Possible values:**

- `spyDB_Access` = 0
- `spyDB_SQLServer` = 1
- `spyDB_Oracle` = 2
- `spyDB_Sybase` = 3
- `spyDB_MySQL` = 4
- `spyDB_DB2` = 5
- `spyDB_Other` = 6
- `spyDB_Unspecified` = 7
- `spyDB_PostgreSQL` = 8
- `spyDB_iSeries` = 9
3.3.10 SPYDialogAction

Description
Values to simulate different interactions on dialogs. See Dialogs for all dialogs available.

Possible values:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spyDialogOK</td>
<td>0</td>
<td>// simulate click on OK button</td>
</tr>
<tr>
<td>spyDialogCancel</td>
<td>1</td>
<td>// simulate click on Cancel button</td>
</tr>
<tr>
<td>spyDialogUserInput</td>
<td>2</td>
<td>// show dialog and allow user interaction</td>
</tr>
</tbody>
</table>
3.3.11 SPYDOMType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

- spyDOMType_mxml4 = 0  Obsolete
- spyDOMType_xerces = 1
- spyDOMType_xerces3 = 2
- spyDOMType_mxml6 = 3

spyDOMType_xerces indicates Xerces 2.x usage; spyDOMType_xerces3 indicates Xerces 3.x usage.
3.3.12 SPYDTDSchemaFormat

**Description**
Enumeration to identify the different schema formats.

**Possible values:**
- spyDTD = 0
- spyW3C = 1
3.3.13 **SPYEncodingByteOrder**

**Description**
Enumeration values to specify encoding byte ordering for text import and export.

**Possible values:**
- `spyNONE` = 0
- `spyLITTLE_ENDIAN` = 1
- `spyBIG_ENDIAN` = 2
3.3.14  SPYExportNamespace

Description
Enumeration type to configure handling of namespace identifiers during export.

Possible values:
spyNoNamespace = 0
spyReplaceColonWithUnderscore = 1
3.3.15 **SPYFindInFilesSearchLocation**

**Description**
The different locations where a search can be performed. This type is used with the `FindInFilesDlg` dialog.

**Possible values:**
- `spyFindInFiles_Documents` = 0
- `spyFindInFiles_Project` = 1
- `spyFindInFiles_Folder` = 2
3.3.16  **SPYFrequentElements**

**Description**
Enumeration value to parameterize schema generation.

**Possible values:**
- `spyGlobalElements = 0`
- `spyGlobalComplexType = 1`
3.3.17  SPYImageKind

Description
Enumeration values to parameterize image type of the generated documentation. These values are used in `SchemaDocumentationDialog.DiagramFormat`.

Possible values:
- `spyImageType_PNG` = 0
- `spyImageType_EMF` = 1
3.3.18 **SPYImportColumnsType**

**Description**
Enumeration to specify different import columns types.

**Possible values:**

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>spylImportColumns_Element</td>
<td>0</td>
</tr>
<tr>
<td>spylImportColumns_Attribute</td>
<td>1</td>
</tr>
</tbody>
</table>
3.3.19 SPYKeyEvent

Description
Enumeration type to identify the different key events. These events correspond with the equally named windows messages.

Possible values:

- spyKeyDown = 0
- spyKeyUp = 1
- spyKeyPressed = 2
3.3.20  **SPYKeyStatus**

**Description**
Enumeration type to identify the key status.

**Possible values:**

- `spyLeftShiftKeyMask` = 1
- `spyRightShiftKeyMask` = 2
- `spyLeftCtrlKeyMask` = 4
- `spyRightCtrlKeyMask` = 8
- `spyLeftAltKeyMask` = 16
- `spyRightAltKeyMask` = 32
3.3.21 SPYLibType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:
- spyLibType_static = 0
- spyLibType_dll = 1
3.3.22 SPYLoading

Description
Enumeration values to define loading behaviour of URL files.

Possible values:

- spyUseCacheProxy = 0
- spyReload = 1
3.3.23  **SPYMouseEvent**

**Description**
Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather than directly comparing with them.

**Examples**

' to check for ctrl-leftbutton-down in VB
If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or XMLSpyLib.spyCtrlKeyDownMask)) Then
    ' react on ctrl-leftbutton-down
End If

' to check for double-click with any button in VBScript
If (((i_eMouseEvent And spyDoubleClickMask) <> 0) Then
    ' react on double-click
End If

**Possible values:**
- spyNoButtonMask = 0
- spyMouseMoveMask = 1
- spyLeftButtonMask = 2
- spyMiddleButtonMask = 4
- spyRightButtonMask = 8
- spyButtonDownMask = 16
- spyButtonDownMask = 32
- spyDoubleClickMask = 64
- spyShiftKeyDownMask = 128
- spyCtrlKeyDownMask = 256
- spyLeftButtonDownMask = 34
- spyMiddleButtonDownMask = 36
- spyRightButtonDownMask = 40
- spyLeftButtonUpMask = 18
- spyMiddleButtonUpMask = 20
- spyRightButtonUpMask = 24
- spyLeftDoubleClickMask = 66
- spyMiddleDoubleClickMask = 68
- spyRightDoubleClickMask = 72
3.3.24 SPYNumberDateTimeFormat

Description
Enumeration value to configure database connections.

Possible values:

spySystemLocale = 0
spySchemaCompatible = 1
3.3.25 SPYProgrammingLanguage

Description
 Enumeration values to select the programming language for code generation from schema definitions.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Possible values:

- spyUndefinedLanguage = -1
- spyJava = 0
- spyCpp = 1
- spyCSharp = 2
3.3.26 SPYProjectItemTypes

**Description**
Enumeration values to identify the different elements in project item lists. See SpyProjectItem.ItemType.

**Possible values:**
- spyUnknownItem = 0
- spyFileItem = 1
- spyFolderItem = 2
- spyURLItem = 3
3.3.27 SPYProjectType

Description
Enumeration values to parameterize generation of C# from schema definitions.

Possible values:

spyVisualStudioProject = 0    Obsolete
spyVisualStudio2003Project  = 1    Obsolete
spyBorlandProject          = 2    Obsolete
spyMonoMakefile            = 3    Obsolete
spyVisualStudio2005Project = 4    For C++ code also
spyVisualStudio2008Project = 5    For C++ code also
spyVisualStudio2010Project = 6    For C++ code also
3.3.28 SpySampleXMLGenerationChoiceMode

Description
This enumeration is used in GenerateSampleXMLDlg.ChoiceMode:

spySampleXMLGen_FirstBranch = 0
spySampleXMLGen_AllBranches  = 1
spySampleXMLGen_ShortestBranch = 2
3.3.29 **SPYSampleXMLGenerationOptimization (Obsolete)**

This enumeration is OBSOLETE since v2014.

**Description**
 Specify the elements that will be generated in the Sample XML. This enumeration is used in `GenerateSampleXMLDlg`.

**Possible values:**

- `spySampleXMLGen_Optimized` = 0
- `spySampleXMLGen_NonMandatoryElements` = 1
- `spySampleXMLGen_Everything` = 2
3.3.30 SpySampleXMLGenerationSampleValueHints

Description
This enumeration is used in GenerateSampleXMLDlg.SampleValueHints

spySampleXMLGen_FirstFit = 0
spySampleXMLGen_RandomFit = 1
spySampleXMLGen_CycleThrough = 2
3.3.31  **SPYSampleXMLGenerationSchemaOrDTDAssignment**

**Description**
Specifies what kind of reference to the schema/DTD should be added to the generated Sample XML.
This enumeration is used in `GenerateSampleXMLDlg`.

**Possible values:**
- `spySampleXMLGen_AssignRelatively` = 0
- `spySampleXMLGen_AssignAbsolutely` = 1
- `spySampleXMLGen_DoNotAssign` = 2
3.3.32  SPYSchemaDefKind

Description
Enumeration type to select schema diagram types.

Possible values:

spyKindElement = 0
spyKindComplexType = 1
spyKindSimpleType = 2
spyKindGroup = 3
spyKindModel = 4
spyKindAny = 5
spyKindAttr = 6
spyKindAttrGroup = 7
spyKindAttrAny = 8
spyKindIdentityUnique = 9
spyKindIdentityKey = 10
spyKindIdentityKeyRef = 11
spyKindIdentitySelector = 12
spyKindIdentityField = 13
spyKindNotation = 14
spyKindInclude = 15
spyKindImport = 16
spyKindRedefine = 17
spyKindFacet = 18
spyKindSchema = 19
spyKindCount = 20
3.3.33  SPYSchemaDocumentationFormat

Description
Enumeration values to parameterize generation of schema documentation. These values are used in `SchemaDocumentationDialog.OutputFormat`.

Possible values:

- `spySchemaDoc_HTML` = 0
- `spySchemaDoc_MSWord` = 1
- `spySchemaDoc_RTF` = 2
- `spySchemaDoc_PDF` = 3


3.3.34  **SPYSchemaExtensionType**

**Description**
Enumeration to specify different Schema Extension types.

**Possible values:**

- `spySchemaExtension_None` = 0
- `spySchemaExtension_SQL_XML` = 1
- `spySchemaExtension_MS_SQL_Server` = 2
- `spySchemaExtension_Oracle` = 3
3.3.35  **SPYSchemaFormat**

**Description**
Enumeration to specify different Schema Format types.

**Possible values:**

<table>
<thead>
<tr>
<th>Enum</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>spySchemaFormat_Hierarchical</td>
<td>0</td>
</tr>
<tr>
<td>spySchemaFormat_Flat</td>
<td>1</td>
</tr>
</tbody>
</table>
3.3.36 SPYTextDelimiters

**Description**
Enumeration values to specify text delimiters for text export.

**Possible values:**
- spyTabulator = 0
- spySemicolon = 1
- spyComma = 2
- spySpace = 3
3.3.37  **SPYTextEnclosing**  

**Description**  
Enumeration value to specify text enclosing characters for text import and export.

**Possible values:**
- `spyNoEnclosing` = 0
- `spySingleQuote` = 1
- `spyDoubleQuote` = 2
3.3.38 SPYTypeDetection

Description
Enumeration to select how type detection works during GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:
- spyBestPossible = 0
- spyNumbersOnly = 1
- spyNoDetection = 2
3.3.39 SPYURLTypes

Description
Enumeration to specify different URL types.

Possible values:
spyURLTypeAuto = -1
spyURLTypeXML = 0
spyURLTypeDTD = 1
3.3.40  SPYValidateXSDVersion

Description
Enumeration values that select what XSD version to use. The XSD version that is selected depends on both (i) the presence/absence—and, if present, the value—of the /xs:schema/@vc:minVersion attribute of the XSD document, and (ii) the value of this enumeration.

spyValidateXSDVersion_1_0 selects XSD 1.0 if vc:minVersion is absent, or is present with any value.
spyValidateXSDVersion_1_1 selects XSD 1.1 if vc:minVersion is absent, or is present with any value.
spyValidateXSDVersion_AutoDetect selects XSD 1.1 if vc:minVersion=1.1. If the vc:minVersion attribute is absent, or is present with a value other than 1.1, then XSD 1.0 is selected.

Possible values
spyValidateXSDVersion_AutoDetect = 0
spyValidateXSDVersion_1_1 = 1
spyValidateXSDVersion_1_0 = 2
3.3.41  **SPYValidateErrorFormat**

**Description**
Enumeration values that select the format of the error message.

**Possible values**

- `spyValidateErrorFormat_Text` = 0
- `spyValidateErrorFormat_ShortXML` = 1
- `spyValidateErrorFormat_LongXML` = 2
3.3.42 SPYViewModes

Description
Enumeration values that define the different view modes for XML documents. The mode `spyViewAuthentic` (4) identifies the mode that was intermediately called DocEdit mode and is now called Authentic mode. The mode `spyViewJsonSchema` identifies a mode which is mapped to the Schema Design View on the GUI but is distinguished internally.

Possible values:

- `spyViewGrid` = 0
- `spyViewText` = 1
- `spyViewBrowser` = 2
- `spyViewSchema` = 3
- `spyViewContent` = 4   // obsolete
- `spyViewAuthentic` = 4
- `spyViewWSDL` = 5
- `spyViewZIP` = 6
- `spyViewEditionInfo` = 7
- `spyViewXBRL` = 8
- `spyViewJsonSchema` = 9
3.3.43 SPYVirtualKeyMask

Description
Enumeration type for the most frequently used key masks that identify the status of the virtual keys. Use these values as bitmasks rather than directly comparing with them. When necessary, you can create further masks by using the 'logical or' operator.

Examples

' VBScript sample: check if ctrl-key is pressed
If ((i_nVirtualKeyStatus And spyCtlKeyMask) <> 0)) Then
    ' ctrl-key is pressed
End If

' VBScript sample: check if ONLY ctrl-key is pressed
If (i_nVirtualKeyStatus == spyCtlKeyMask) Then
    ' exactly ctrl-key is pressed
End If

// JScript sample: check if any of the right virtual keys is pressed
if ((i_nVirtualKeyStatus & (spyRightShiftKeyMask | spyRightCtrlKeyMask | spyRightAltKeyMask)) != 0)
{
    ; ' right virtual key is pressed
}

Possible values:

<table>
<thead>
<tr>
<th>Mask Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>spyNoVirtualKeyMask</td>
<td>0</td>
</tr>
<tr>
<td>spyLeftShiftKeyMask</td>
<td>1</td>
</tr>
<tr>
<td>spyRightShiftKeyMask</td>
<td>2</td>
</tr>
<tr>
<td>spyLeftCtrlKeyMask</td>
<td>4</td>
</tr>
<tr>
<td>spyRightCtrlKeyMask</td>
<td>8</td>
</tr>
<tr>
<td>spyLeftAltKeyMask</td>
<td>16</td>
</tr>
<tr>
<td>spyRightAltKeyMask</td>
<td>32</td>
</tr>
<tr>
<td>spyShiftKeyMask</td>
<td>3</td>
</tr>
<tr>
<td>spyCtrlKeyMask</td>
<td>12</td>
</tr>
<tr>
<td>spyAltKeyMask</td>
<td>48</td>
</tr>
</tbody>
</table>

// spyLeftShiftKeyMask | spyRightShiftKeyMask
// spyLeftCtrlKeyMask | spyRightCtrlKeyMask
// spyLeftAltKeyMask | spyRightAltKeyMask
3.3.44 **SPYXMLDataKind**

**Description**
The different types of XMLData elements available for XML documents.

**Possible values:**

```plaintext
spyXMLDataXMLDocStruct  = 0
spyXMLDataXMLEntityDocStruct = 1
spyXMLDataDTDDocStruct  = 2
spyXMLDataXML          = 3
spyXMLDataElement      = 4
spyXMLDataAttr         = 5
spyXMLDataText         = 6
spyXMLDataCData        = 7
spyXMLDataComment      = 8
spyXMLDataPI           = 9
spyXMLDataDefDoctype   = 10
spyXMLDataDefExternalID = 11
spyXMLDataDefElement   = 12
spyXMLDataDefAttlist   = 13
spyXMLDataDefEntity    = 14
spyXMLDataDefNotation  = 15
spyXMLDataKindsCount   = 16
```
4  **ActiveX Integration**

The Authentic Desktop user interface and the functionality described in this section can be integrated into custom applications that can consume ActiveX controls. ActiveX technology enables a wide variety of languages to be used for integration, such as C++, C#, VB.NET, HTML. (Note that ActiveX components integrated in HTML must be run with Microsoft Internet Explorer versions and platforms that support ActiveX). All components are full OLE Controls. Integration into Java is provided through wrapper classes.

To integrate the ActiveX controls into your custom code, the Authentic Desktop Integration Package must be installed (see https://www.altova.com/components/download). Ensure that you install Authentic Desktop first, and then the Authentic Desktop Integration Package. Other prerequisites apply, depending on language and platform (see Prerequisites).

You can flexibly choose between two different levels of integration: application level and document level.

Integration at application level means embedding the complete interface of Authentic Desktop (including its menus, toolbars, panes, etc) as an ActiveX control into your custom application. For example, in the most simple scenario, your custom application could consist of only one form that embeds the Authentic Desktop graphical user interface. This approach is easier to implement than integration at document level but may not be suitable if you need flexibility to configure the Authentic Desktop graphical user interface according to your custom requirements.

Integration at document level means embedding Authentic Desktop into your own application piece-by-piece. This includes implementing not only the main Authentic Desktop control but also the main document editor window, and, optionally, any additional windows. This approach provides greater flexibility to configure the GUI, but requires advanced interaction with ActiveX controls in your language of choice.

The sections Integration at the Application Level and Integration at Document Level describe the key steps at these respective levels. The ActiveX Integration Examples section provides examples in C#, HTML, and Java. Looking through these examples will help you to make the right decisions quickly. The Object Reference section describes all COM objects that can be used for integration, together with their properties and methods.

For information about using Authentic Desktop as a Visual Studio plug-in, see Authentic Desktop in Visual Studio.
4.1 Prerequisites

To integrate the Authentic Desktop ActiveX control into a custom application, the following must be installed on your computer:

- Authentic Desktop
- The Authentic Desktop Integration Package, available for download at https://www.altova.com/components/download

To integrate the 64-bit ActiveX control, install the 64-bit versions of Authentic Desktop and Authentic Desktop Integration Package. For applications developed under Microsoft .NET platform with Visual Studio, both the 32-bit and 64-bit versions of Authentic Desktop and Authentic Desktop Integration Package must be installed, as explained below.

**Microsoft .NET (C#, VB.NET) with Visual Studio**

To integrate the Authentic Desktop ActiveX control into a 32-bit application developed under Microsoft .NET, the following must be installed on your computer:

- Microsoft .NET Framework 4.0 or later
- Visual Studio 2010/2012/2013/2015/2017
- Authentic Desktop 32-bit and Authentic Desktop Integration Package 32-bit
- The ActiveX controls must be added to the Visual Studio toolbox (see Adding the ActiveX Controls to the Toolbox).

If you want to integrate the 64-bit ActiveX control, the following prerequisites apply in addition to the ones above:

- Authentic Desktop 32-bit and Authentic Desktop Integration Package 32-bit must still be installed (this is required to provide the 32-bit ActiveX control to the Visual Studio designer, since Visual Studio runs on 32-bit)
- Authentic Desktop 64-bit and Authentic Desktop Integration Package 64-bit must be installed (provides the actual 64-bit ActiveX control to your custom application at runtime)
- In Visual Studio, create a 64-bit build configuration and build your application using this configuration. For an example, see Running the Sample C# Solution.

**Java**

To integrate the Authentic Desktop ActiveX control into Java application using the Eclipse development environment, the following must be installed on your computer:

- Java Runtime Environment (JRE) or Java Development Kit (JDK) 7 or later
- Eclipse
- Authentic Desktop and Authentic Desktop Integration Package

**Note:** To run the 64-bit version of the Authentic Desktop ActiveX control, use a 64-bit version of Eclipse, as well as the 64-bit version of Authentic Desktop and the Authentic Desktop Integration Package.

**Authentic Desktop integration and deployment on client computers**

If you create a .NET application and intend to distribute it to other clients, you will need to install the following on the client computer(s):
- Authentic Desktop
- The Authentic Desktop Integration Package
- The custom integration code or application.
4.2 Adding the ActiveX Controls to the Toolbox

To use the Authentic Desktop ActiveX controls in an application developed with Visual Studio, the controls must first be added to the Visual Studio Toolbox, as follows:

1. On the Tools menu of Visual Studio, click Choose Toolbox Items.
2. On the COM Components tab, select the check boxes next to the Authentic DesktopControl, Authentic DesktopControl Document, and Authentic DesktopControl Placeholder.

In case the controls above are not available, follow the steps below:

1. On the COM Components tab, click Browse, and select the AuthenticControl.ocx file from the Authentic Desktop installation folder. Remember that the Authentic Desktop Integration Package must be installed; otherwise, this file is not available, see Prerequisites.
2. If prompted to restart Visual Studio with elevated permissions, click Restart under different credentials.

If the steps above were successful, the Authentic Desktop ActiveX controls become available in the Visual Studio Toolbox.
**Note:** For an application-level integration, only the **AuthenticDesktopControl** ActiveX control is used (see [Integration at Application Level](#)). The **AuthenticDesktopControl Document** and **AuthenticDesktopControl Placeholder** controls are used for document-level integration (see [Integration at Document Level](#)).
4.3 Integration at Application Level

Integration at application level allows you to embed the complete interface of Authentic Desktop into a window of your application. With this type of integration, you get the whole user interface of Authentic Desktop, including all menus, toolbars, the status bar, document windows, and helper windows. Customization of the application's user interface is restricted to what Authentic Desktop provides. This includes rearrangement and resizing of helper windows and customization of menus and toolbars.

The only ActiveX control you need to integrate is `AuthenticDesktopControl`. Do not instantiate or access `AuthenticDesktopControlDocument` or `AuthenticDesktopControlPlaceholder` ActiveX controls when integrating at application-level.

If you have any initialization to do or if you want to automate some behaviour of Authentic Desktop, use the properties, methods, and events described for `AuthenticDesktopControl`. Consider using `AuthenticDesktopControl.Application` for more complex access to Authentic Desktop functionality.

For an example that shows how the Authentic Desktop application can be embedded in an HTML page, see HTML Integration at Application Level.

In C# or VB.NET with Visual Studio, the steps to create a basic, one-form application which integrates the Authentic Desktop ActiveX controls at application level are as follows:

1. Check that all prerequisites are met (see Prerequisites).
2. Create a new Visual Studio Windows Forms project with a new empty form.
3. If you have not done that already, add the ActiveX controls to the toolbox (see Adding the ActiveX Controls to the Toolbox).
4. Drag the `AuthenticDesktopControl` from the toolbox onto your new form.
5. Select the `AuthenticDesktopControl` on the form, and, in the Properties window, set the `IntegrationLevel` property to `ICActiveXIntegrationOnApplicationLevel`. 
6. Create a build platform configuration that matches the platform under which you want to build (x86, x64). Here is how you can create the build configuration:

   a. Right-click the solution in Visual Studio, and select **Configuration Manager**.
   b. Under **Active solution platform**, select **New...** and then select the x86 or x64 configuration (in this example, **x86**).
You are now ready to build and run the solution in Visual Studio. Remember to build using the configuration that matches your target platform (x86, x64).
4.4 Integration at Document Level

Compared to integration at application level, integration at document level is a more complex, yet more flexible way to embed Authentic Desktop functionality into your application by means of ActiveX controls. With this approach, your code can access selectively the following parts of the Authentic Desktop user interface:

- Document editing window
- Project window
- Info window
- Messages window
- Entry helper windows (Elements, Attributes, Entities)
- Output window

As mentioned in Integration at Application Level, for an ActiveX integration at application level, only one control is required, namely the AuthenticDesktopControl. However, for an ActiveX integration at document level, functionality Authentic Desktop is provided by the following ActiveX controls:

1. AuthenticDesktopControl
2. AuthenticDesktopControl Document
3. AuthenticDesktopControl Placeholder

These controls are supplied by the AuthenticControl.ocx file available in the application installation folder of Authentic Desktop. When you develop the ActiveX integration with Visual Studio, you will need to add these controls to the Visual Studio toolbox (see Adding the ActiveX Controls to the Toolbox).

The basic steps to integrate the ActiveX controls at document level into your application are as follows:

1. First, instantiate AuthenticDesktopControl in your application. Instantiating this control is mandatory; it enables support for the AuthenticDesktopControl Document and AuthenticDesktopControl Placeholder controls mentioned above. It is important to set the IntegrationLevel property to ICActiveXIntegrationOnDocumentLevel (or “1”). To hide the control from the user, set its Visible property to False.

Note: When integrating at document level, do not use the Open method of the AuthenticDesktopControl; this might lead to unexpected results. Use the corresponding open methods of AuthenticDesktopControl Document and AuthenticDesktopControl Placeholder instead.

2. Create at least one instance of AuthenticDesktopControl Document in your application. This control supplies the document editing window of Authentic Desktop to your application and can be instantiated multiple times if necessary.

Use the method Open to load any existing file. To access document-related functionality, use the Path and Save or methods and properties accessible via the property Document.

Note: The control does not support a read-only mode. The value of the property ReadOnly is ignored.
3. Optionally, add to your application the `AuthenticDesktopControl Placeholder` control for each additional window (other than the document window) that must be available to your application.

Instances of `AuthenticDesktopControl Placeholder` allow you to selectively embed additional windows of Authentic Desktop into your application. The window kind (for example, Project window) is defined by the property `PlaceholderWindowID`. Therefore, to set the window kind, set the property `PlaceholderWindowID`. For valid window identifiers, see `AuthenticDesktopControlPlaceholderWindow`.

**Note:** Use only one `AuthenticDesktopControl Placeholder` for each window identifier.

For placeholder controls that select the Authentic Desktop project window, additional methods are available. Use `OpenProject` to load a Authentic Desktop project. Use the property `Project` and the methods and properties from the Authentic Desktop automation interface to perform any other project related operations.

For example, in C# or VB.NET with Visual Studio, the steps to create a basic, one-form application which integrates the Authentic Desktop ActiveX controls at document level could be similar to those listed below. Note that your application may be more complex if necessary; however, the instructions below are important to understand the minimum requirements for an ActiveX integration at document level.

1. Create a new Visual Studio Windows Forms project with a new empty form.
2. If you have not done that already, add the ActiveX controls to the toolbox (see `Adding the ActiveX Controls to the Toolbox`).
3. Drag the `AuthenticDesktopControl` from the toolbox onto your new form.
4. Set the `IntegrationLevel` property of the `AuthenticDesktopControl` to `ICActiveXIntegrationOnDocumentLevel`, and the `Visible` property to `False`. You can do this either from code or from the `Properties` window.
5. Drag the `AuthenticDesktopControl Document` from the toolbox onto the form. This control provides the main document window of Authentic Desktop to your application, so you may need to resize it to a reasonable size for a document.
6. Optionally, add one or more `AuthenticDesktopControl Placeholder` controls to the form (one for each additional window type that your application needs, for example, the `Project` window). You will typically want to place such additional placeholder controls either below or to the right or left of the main document control, for example:
7. Set the `PlaceholderWindowID` property of each `AuthenticDesktopControl Placeholder` control to a valid window identifier. For the list of valid values, see `AuthenticDesktopControlPlaceholderWindow`.

8. Add commands to your application (at minimum, you will need to open, save and close documents), as shown below.

**Querying Authentic Desktop Commands**

When you integrate at document level, no Authentic Desktop menu or toolbar is available to your application. Instead, you can retrieve the required commands, view their status, and execute them programmatically, as follows:

- To retrieve all available commands, use the `CommandsList` property of the `AuthenticDesktopControl`.
- To retrieve commands organized according to their menu structure, use the `MainMenu` property.
- To retrieve commands organized by the toolbar in which they appear, use the `Toolbars` property.
- To send commands to Authentic Desktop, use the `Exec` method.
- To query if a command is currently enabled or disabled, use the `QueryStatus` method.

This enables you to flexibly integrate Authentic Desktop commands into your application's menus and toolbars.

Your installation of Authentic Desktop also provides you with command label images used within Authentic Desktop. See the folder `<ApplicationFolder>\Examples\ActiveX\Images` of your Authentic Desktop installation for icons in GIF format. The file names correspond to the command names as they are listed in the Command Reference section.
General considerations
To automate the behaviour of Authentic Desktop, use the properties, methods, and events
described for the AuthenticDesktopControl, AuthenticDesktopControl Document, and
AuthenticDesktopControl Placeholder.

For more complex access to Authentic Desktop functionality, consider using the following
properties:

- AuthenticDesktopControl.Application
- AuthenticDesktopControlDocument.Document
- AuthenticDesktopControlPlaceHolder.Project

These properties give you access to the Authentic Desktop automation interface
(AuthenticDesktopAPI)

**Note:** To open a document, always use AuthenticDesktopControlDocument.Open or
AuthenticDesktopControlDocument.New on the appropriate document control. To open
a project, always use AuthenticDesktopControlPlaceHolder.OpenProject on a
placeholder control embedding a Authentic Desktop project window.

For examples that show how to instantiate and access the necessary controls in different
programming environments, see ActiveX Integration Examples.
4.5 ActiveX Integration Examples

This section contains examples of Authentic Desktop document-level integration using different container environments and programming languages. (The HTML section additionally contains examples of integration at application level.) Source code for all examples is available in the folder `<ApplicationFolder>\Examples\ActiveX` of your Authentic Desktop installation.
4.5.1 C#

A basic ActiveX integration example solution for C# and Visual Studio is available in the folder `<ApplicationFolder>\Examples\ActiveX\C#`. Before you compile the source code and run the sample, make sure that all prerequisites are met (see Running the Sample C# Solution).

Running the Sample C# Solution

The sample Visual Studio solution available in the folder `<ApplicationFolder>\Examples\ActiveX\C#` illustrates how to consume the Authentic Desktop ActiveX controls. Before attempting to build and run this solution, note the following steps:

Step 1: Check the prerequisites

Visual Studio 2010 or later is required to open the sample solution. For the complete list of prerequisites, see Prerequisites.

Step 2: Copy the sample to a directory where you have write permissions

To avoid running Visual Studio as an Administrator, copy the source code to a directory where you have write permissions, instead of running it from the default location.

Step 3: Check and set all required control properties

The sample application contains one instance of AuthenticDesktopControlDocument and several instances of AuthenticDesktopControlPlaceHolder controls. Double-check that the following properties of these controls are set as shown in the table below:

<table>
<thead>
<tr>
<th>Control name</th>
<th>Property</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>axAuthenticDesktopControl</td>
<td>IntegrationLevel</td>
<td>ICAActiveXIntegrationOnDocumentLevel</td>
</tr>
<tr>
<td>axAuthenticDesktopControlHelperWndEntities</td>
<td>PlaceholderWndID</td>
<td>2</td>
</tr>
<tr>
<td>axAuthenticDesktopControlHelperWndAttributes</td>
<td>PlaceholderWndID</td>
<td>1</td>
</tr>
<tr>
<td>axAuthenticDesktopControlHelperWndElements</td>
<td>PlaceholderWndID</td>
<td>0</td>
</tr>
<tr>
<td>axAuthenticDesktopControlHelperWndInfo</td>
<td>PlaceholderWndID</td>
<td>18</td>
</tr>
<tr>
<td>axAuthenticDesktopControlHelperWndProject</td>
<td>PlaceholderWndID</td>
<td>4</td>
</tr>
</tbody>
</table>

Here is how you can view or set the properties of an ActiveX control:

1. Open the MDIMain.cs form in the designer window.
Note: On 64-bit Windows, it may be necessary to change the build configuration of the Visual Studio solution to "x86" before opening the designer window. If you need to build the sample as a 64-bit application, see Prerequisites.


3. Click an ActiveX control in the Document Outline window, and edit its required property in the Properties window, for example:
Step 4: Set the build platform

- Create a build platform configuration that matches the platform under which you want to build (x86, x64). Here is how you can create the build configuration:

  a. Right-click the solution in Visual Studio, and select **Configuration Manager**.
  b. Under **Active solution platform**, select **New...** and then select the x86 or x64 configuration (in this example, **x86**). 
You are now ready to build and run the solution in Visual Studio. Remember to build using the configuration that matches your target platform (x86, x64); otherwise, runtime errors might occur.
4.5.2 HTML

The code listings in this section show how to integrate the AuthenticDesktopControl at application level and document level. Source code for all examples is available in the folder <ApplicationFolder>\Examples\ActiveX\HTML of your Authentic Desktop installation.

Note: ActiveX controls in an HTML page are supported only by Internet Explorer when it runs as a 32-bit application. When Internet Explorer 10 or 11 runs in 64-bit mode, it does not load ActiveX controls. The default browser security settings will normally block ActiveX, so you will need to explicitly allow blocked content to run on the page when prompted by Internet Explorer.

HTML Integration at Application Level

This example shows a simple integration of the Authentic Desktop control at application-level into a HTML page. The integration is described in the following sections:

- Instantiate a AuthenticDesktopControl in HTML code.
- Implement buttons to load documents and automate code-generation tasks.
- Define actions for some application events.

The code for this example is available at the following location in your Authentic Desktop installation:
<ApplicationFolder>\Examples\ActiveX\HTML\AuthenticActiveX_ApplicationLevel.htm

Instantiate the Control

The HTML Object tag is used to create an instance of the AuthenticDesktopControl. The Classid is that of AuthenticDesktopControl. Width and height specify the window size. No additional parameters are necessary, since application-level is the default.

```html
<OBJECT id="objAuthenticDesktopControl"
    Classid="clsid:4A3E7996-89B1-4d7b-9D87-615CC7C4CB47"
    width="1000"
    height="700"
    VIEWASTEXT>
</OBJECT>
```

Add Button to Open Default Document

As a simple example of how to automate some tasks, we add a button to the page:

```html
<input type="button" value="Open" onclick="BtnOpen()"
```

When clicked, a predefined document will be opened in the AuthenticDesktopControl. The MakeAbsolutePath method creates an absolute path using the location of the script as a base path.

```javascript
function BtnOpen()
{
    if (strPath.value.length > 0)
Connect to Custom Events

The example implements two event callbacks for AuthenticDesktopControl custom events to show the principle:

<!-- ----------------------------------------------------------- -->
<!--  custom event 'OnDocumentOpened' of AuthenticDesktopControl object  -->
<SCRIPT FOR="objAuthenticDesktopControl" event="OnDocumentOpened( objDocument )" LANGUAGE="javascript">
  // alert("Document '" + objDocument.Name + '" opened!");
</SCRIPT>
<!-- ----------------------------------------------------------- -->
<!--  custom event 'OnDocumentClosed' of AuthenticDesktopControl object  -->
<SCRIPT FOR="objAuthenticDesktopControl" event="OnDocumentClosed( objDocument )" LANGUAGE="javascript">
  // alert("Document '" + objDocument.Name + '" closed!");
</SCRIPT>

HTML Integration at Document Level

This example shows an integration of the Authentic Desktop control at document-level into a HTML page. The following topics are covered:

- Instantiate a AuthenticDesktopControlActiveX control object in HTML code
- Instantiate a AuthenticDesktopControlDocument ActiveX control to allow editing a Authentic Desktop file
- Instantiate one AuthenticDesktopControlPlaceholder for a AuthenticDesktopControl project window
- Instantiate one AuthenticDesktopControlPlaceholder to alternatively host one of the Authentic Desktop helper windows

This example is available in its entirety in the file AuthenticDesktopActiveX_ApplicationLevel.htm within the <ApplicationFolder>\Examples\ActiveX\HTML\ folder of your Authentic Desktop installation.
Instantiate the AuthenticDesktopControl

AuthenticDesktopControlThe HTML OBJECT tag is used to create an instance of the AuthenticDesktopControl. The Classid is that of AuthenticDesktopControl. Width and height are set to 0 since we use this control as manager control without use for its user interface. The integration level is specified as a parameter within the OBJECT tag.

```html
<object id="objAuthenticDesktopControl" classid="clsid:4A3E7996-89B1-4d7b-9D87-615CC7C4CB47" width="0" height="0" VIEWASTEXT>
  <param name="IntegrationLevel" value="1">
</object>
```

Create Editor Window

The HTML OBJECT tag is used to embed an editing window. The additional custom parameter specifies that the control is to be initialized with a new empty document.

```html
<object id="objDoc1" classid="clsid:EBD19852-F38E-4274-B48E-47A8EA8EF450" width="800" height="500" VIEWASTEXT>
  <param name="NewDocument">
</object>
```

Create Project Window

The HTML OBJECT tag is used to create a AuthenticDesktopControlPlaceHolder window. The parameter defines the placeholder to show the Authentic Desktop project window.

```html
<!-- create Project window placeholder control.          -->
<!-- the editor with focus will automatically direct its -->
<!-- output to the appropriate helper window.            -->
<object id="objProjectWindow" classid="clsid:B80F1F5A-B739-4de8-BC76-DDAE2E848C42" width="200" height="250" VIEWASTEXT>
  <param name="PlaceholderWindowID" value="-1">
</object>
```

Create Placeholder for Helper Windows

The AuthenticDesktopControlPlaceHolder control is required to host an application helper window, see also Integration at Document Level. For example, in the code listing below, the HTML object tag is used to instantiate a control that will host the Message window.

```html
<!-- create Message window placeholder control.          -->
<!-- the editor with focus will automatically direct its -->
<!-- output to the appropriate helper window.            -->
<object id="objMessageWindow" classid="clsid:B80F1F5A-B739-4de8-BC76-DDAE2E848C42" width="800" height="200" VIEWASTEXT>
  <param name="PlaceholderWindowID" value="-1">
</object>
```
Notice that the `PlaceholderWindowID` parameter is set to -1, which means that, initially, no helper window is shown. Whenever a file is open, the method `DisplayHelperWindowsContent()` is invoked. This method assigns the appropriate `PlaceholderWindowID` to each control. For the list of possible values of `PlaceholderWindowID`, see `AuthenticDesktopControlPlaceholderWindow`.
4.5.3 Java

Authentic Desktop ActiveX components can be accessed from Java code. Java integration is provided by the libraries listed below. These libraries are available in the folder <ApplicationFolder>\Examples\JavaAPI of your Authentic Desktop installation, after you have installed both Authentic Desktop and the Authentic Desktop Integration Package (see also Prerequisites).

- AltovaAutomation.dll: a JNI wrapper for Altova automation servers (in case of the 32-bit installation of Authentic Desktop)
- AltovaAutomation_x64.dll: a JNI wrapper for Altova automation servers (in case of the 64-bit installation of Authentic Desktop)
- AltovaAutomation.jar: Java classes to access Altova automation servers
- AuthenticActiveX.jar: Java classes that wrap the Authentic ActiveX interface
- AuthenticActiveX_JavaDoc.zip: a Javadoc file containing help documentation for the Java interface

**Note:** In order to use the Java ActiveX integration, the .dll and .jar files must be included in the Java class search path.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details, see Example Java Project.

Rules for mapping the ActiveX Control names to Java

For the documentation of ActiveX controls, see Object Reference. Note that the object naming conventions are slightly different in Java compared to other languages. Namely, the rules for mapping between the ActiveX controls and the Java wrapper are as follows:

- **Classes and class names**
  For every component of the Authentic Desktop ActiveX interface a Java class exists with the name of the component.

- **Method names**
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with get and set can be used. If a property does not support write-access, no setter method is available. Example: For the IntegrationLevel property of the AuthenticDesktopControl, the Java methods getIntegrationLevel and setIntegrationLevel are available.

- **Enumerations**
  For every enumeration defined in the ActiveX interface, a Java enumeration is defined with the same name and values.

- **Events and event handlers**
  For every interface in the automation interface that supports events, a Java interface with the same name plus 'Event' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus 'DefaultHandler'. For example: AuthenticDesktopControl: Java class to access the application
Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Java name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticDesktopControlDocument, method New</td>
<td>newDocument</td>
</tr>
<tr>
<td>Document, method SetEncoding</td>
<td>setFileEncoding</td>
</tr>
<tr>
<td>AuthenticView, method Goto</td>
<td>gotoElement</td>
</tr>
<tr>
<td>AuthenticRange, method Goto</td>
<td>gotoElement</td>
</tr>
<tr>
<td>AuthenticRange, method Clone</td>
<td>cloneRange</td>
</tr>
</tbody>
</table>

This section

This section shows how some basic Authentic Desktop ActiveX functionality can be accessed from Java code. It is organized into the following sub-sections:

- Example Java Project
- Creating the ActiveX Controls
- Loading Data in the Controls
- Basic Event Handling
- Menus
- UI Update Event Handling
- Creating an XML Tree

Example Java Project

The Authentic Desktop installation package contains an example Java project, located in the ActiveX Examples folder of the application folder: `<ApplicationFolder>`\Examples\ActiveX\Java\.

The Java example shows how to integrate the AuthenticDesktopControl in a common desktop application created with Java. You can test it directly from the command line using the batch file BuildAndRun.bat, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

File list

The Java examples folder contains all the files required to run the example project. These files are listed below:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.classpath</td>
<td>Eclipse project helper file</td>
</tr>
<tr>
<td>.project</td>
<td>Eclipse project file</td>
</tr>
<tr>
<td>Directory/ file name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AltovaAutomation.dll</td>
<td>Java-COM bridge: DLL part (for the 32-bit installation)</td>
</tr>
<tr>
<td>AltovaAutomation_x64.dll</td>
<td>Java-COM bridge: DLL part (for the 64-bit installation)</td>
</tr>
<tr>
<td>AltovaAutomation.jar</td>
<td>Java-COM bridge: Java library part</td>
</tr>
<tr>
<td>AuthenticActiveX.jar</td>
<td>Java classes of the Authentic Desktop ActiveX control</td>
</tr>
<tr>
<td>AuthenticActiveX_JavaDoc.zip</td>
<td>Javadoc file containing help documentation for the Java API</td>
</tr>
<tr>
<td>AuthenticDesktopContainer.java</td>
<td>Java example source code</td>
</tr>
<tr>
<td>AuthenticDesktopContainerEventHandler.java</td>
<td>Java example source code</td>
</tr>
<tr>
<td>BuildAndRun.bat</td>
<td>Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.</td>
</tr>
<tr>
<td>XMLTreeDialog.java</td>
<td>Java example source code</td>
</tr>
</tbody>
</table>

What the example does
The example places one AuthenticDesktop document editor window, the Project window, the Info window and an Authentic entry helper in an AWT frame window. It reads out the File menu defined for Authentic and creates an AWT menu with the same structure. You can use this menu or the project window to open and work with files in the document editor.

You can modify the example in any way you like.

The following specific features are described in code listings:

- **Creating the ActiveX Controls**: Starts Authentic Desktop, which is registered as an automation server, or activates Authentic Desktop if it is already running.
- **Loading Data in the Controls**: Locates one of the example documents installed with Authentic Desktop and opens it.
- **Basic Event Handling**: Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- **Menus**: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- **UI Update Event Handling**: Shows how to handle Authentic Desktop events.
- **Creating an XML Tree**: Shows how to create an XML tree and prepare it for modal activation.

Updating the path to the Examples folder
Before running the provided sample, you may need to edit the AuthenticDesktopContainer.java file. Namely, check that the following path refers to the actual folder where the Authentic Desktop
example files are stored on your operating system:

```java
final String strExamplesFolder = System.getenv("USERPROFILE") + "\Documents \Altova\Authentic2019\AuthenticExamples\";
```

### Running the example from the command line

To run the example from the command line:

1. Check that all prerequisites are met (see Prerequisites).
2. Open a command prompt window, change the current directory to the sample Java project folder, and type:

   ```cmd
   buildAndRun.bat "<Path-to-the-Java-bin-folder>"
   ```

3. Press Enter.

The Java source in AuthenticDesktopContainer.java will be compiled and then executed.

### Compiling and running the example in Eclipse

To import the sample Java project into Eclipse:

1. Check that all prerequisites are met (see Prerequisites).
2. On the File menu, click Import.
3. Select Existing Projects into Workspace, and browse for the Eclipse project file located at `<ApplicationFolder>\Examples\ActiveX\Java\`. Since you may not have write-access in this folder, it is recommended to select the Copy projects into workspace check box on the Import dialog box.

To run the example application, right-click the project in Package Explorer and select the command Run as | Java Application.

Help for Java API classes is available through comments in code as well as the Javadoc view of Eclipse. To enable the Javadoc view in Eclipse, select the menu command Window | Show View | JavaDoc.

### Creating the ActiveX Controls

The code listing below show how ActiveX controls can be created. The constructors will create the Java wrapper objects. Adding these Canvas-derived objects to a panel or to a frame will trigger the creation of the wrapped ActiveX object.

```java
/**
 * Authentic Desktop manager control - always needed
 */
public static AuthenticDesktopControl authenticDesktopControl = null;
/**
 * Authentic Desktop document editing control
 */
```
08  */
09 public static AuthenticDesktopControlDocument authenticDesktopDocument = null;
10 */
11    */
12    * Tool windows - Authentic Desktop place-holder controls
13    */
14 private static AuthenticDesktopControlPlaceholder
authenticDesktopInfoToolWindow = null;
15 private static AuthenticDesktopControlPlaceholder
authenticDesktopEHElementToolWindow = null;
16 private static AuthenticDesktopControlPlaceholder
authenticDesktopProjectToolWindow = null;
17 // Create the Authentic Desktop ActiveX control, The parameter determines
18 // that we want
19 // to place document controls and place-holder controls individually.
20 // It gives us full control over the menu, as well.
21 authenticDesktopControl = new AuthenticDesktopControl(
    ICActiveXIntegrationLevel.ICActiveXIntegrationOnDocumentLevel.getValue());
22 authenticDesktopDocument = new AuthenticDesktopControlDocument();
23 authenticDesktopDocument.setPreferredSize( new Dimension( 640, 480 ) );
24 frame.add( authenticDesktopDocument, BorderLayout.CENTER );
25
26 // Create a project window and open the sample project in it
27 authenticDesktopProjectToolWindow = new
AuthenticDesktopControlPlaceHolder(
    XMLSpyControlPlaceholderWindow.XMLSpyControlProjectWindowToolWnd.getValue() );
28 authenticDesktopProjectToolWindow.setPreferredSize( new Dimension( 200, 200 ) );

Loading Data in the Controls
The code listing below show how data can be loaded in the ActiveX controls.

1    // Locate samples installed with the product.
2    final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\Documents\Altova\Authentic2019\AuthenticExamples\";
3    authenticDesktopControlPlaceholder = new AuthenticDesktopControlPlaceHolder( XMLSpyControlPlaceholderWindow.XMLSpyControlProjectWindowToolWnd.getValue() );

Basic Event Handling
The code listing below shows how basic events can be handled. When calling the
AuthenticDesktopControl’s open method, or when trying to open a file via the menu or Project
tree, the onOpenedOrFocused event is sent to the attached event handler. The basic handling for
this event is opening the file by calling the Authentic DesktopDocumentControl’s open method.

01 // Open the PXF file when button is pressed
btnOpenPxf.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            authenticDesktopControl.open(strExamplesFolder + "OrgChart.pxf");
        } catch (AutomationException e1) {
            e1.printStackTrace();
        }
    }
});

public void onOpenedOrFocused(String i_strFileName, boolean i_bOpenWithThisControl, boolean i_bFileAlreadyOpened) throws AutomationException {
    // Handle the New/Open events coming from the Project tree or from the menus
    if (!i_bFileAlreadyOpened) {
        // This is basically an SDI interface, so open the file in the already existing document control
        try {
            AuthenticDesktopContainer.authenticDesktopDocument.open(i_strFileName);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

Menus

The code listing below shows how menu items can be created. Each AuthenticDesktopCommand object gets a corresponding MenuItem object, with the ActionCommand set to the ID of the command. The actions generated by all menu items are handled by the same function, which can perform specific handlings (like reinterpreting the closing mechanism) or can delegate the execution to the AuthenticDesktopControl object by calling its exec method. The menuMap object that is filled during menu creation is used later (see section UI Update Event Handling).

// Load the file menu when the button is pressed
btnMenu.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            // Create the menubar that will be attached to the frame
            MenuBar mb = new MenuBar();
            // Load the main menu's first item - the File menu
            XMLSpyCommand xmlSpyMenu = xmlSpyControl.getMainMenu().getSubCommands().getItem(0);
            // Create Java menu items from the Commands objects
            Menu fileMenu = new Menu();
            handlerObject.fillMenu(fileMenu, xmlSpyMenu.getSubCommands());
            fileMenu.setLabel(xmlSpyMenu.getLabel().replace("\", ""));
        }
    }
});
mb.add( fileMenu );
frame.setMenuBar( mb );
frame.validate();
}
catch (AutomationException e1) {
e1.printStackTrace();
}
// Disable the button when the action has been performed
((AbstractButton) e.getSource()).setEnabled( false );

/** * Populates a menu with the commands and submenus contained in an * 
XMLSpyCommands object */
public void fillMenu( Menu newMenu, XMLSpyCommands xmlSpyMenu ) throws 
AutomationException
{
// For each command/submenu in the xmlSpyMenu
for ( int i = 0 ; i < xmlSpyMenu.getCount() ; ++i )
{
XMLSpyCommand xmlSpyCommand = xmlSpyMenu.getItem( i );
if ( xmlSpyCommand.getIsSeparator() )
newMenu.addSeparator();
else
{
XMLSpyCommands subCommands = xmlSpyCommand.getSubCommands();
// Is it a command (leaf), or a submenu?
if ( subCommands.isNull() || subCommands.getCount() == 0 )
{
// Command -> add it to the menu, set its ActionCommand to its ID 
and store it in the menuMap
MenuItem mi = new MenuItem( xmlSpyCommand.getLabel().replace( "&", 
"" ) );
mi.setActionCommand( "" + xmlSpyCommand.getID() );
mi.addActionListener( this );
newMenu.add( mi );
menuMap.put( xmlSpyCommand.getID(), mi );
}
else
{
// Submenu -> create submenu and repeat recursively
Menu newSubMenu = new Menu();
fillMenu( newSubMenu, subCommands );
newSubMenu.setLabel( xmlSpyCommand.getLabel().replace( "&", "" ) );
newMenu.add( newSubMenu );
}
}
/** * Action handler for the menu items * Called when the user selects a menu item; the item's action command 
corresponds to the command table for XMLSpy */
public void actionPerformed( ActionEvent e )
{
try
{
int iCmd = Integer.parseInt( e.getActionCommand() );
// Handle explicitly the Close commands
switch ( iCmd )
UI Update Event Handling
The code listing below shows how a UI-Update event handler can be created.

```java
/**
 * Call-back from the XMLSpyControl.
 * Called to enable/disable commands
 */
@Override
public void onUpdateCmdUI() throws AutomationException {
    // A command should be enabled if the result of queryStatus contains the
    // Supported (1) and Enabled (2) flags
    for (java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet())
        pair.getValue().setEnabled(AuthenticDesktopContainer.authenticDesktopControl.queryStatus(pair.getKey()) > 2);
}

/**
 * Call-back from the XMLSpyControl.
 * Usually called while enabling/disabling commands due to UI updates
 */
@Override
public boolean onIsActiveEditor(String i_strFilePath) throws AutomationException {
    try {
        return AuthenticDesktopContainer.authenticDesktopDocument.getDocument().getFullName().equalsIgnoreCase(i_strFilePath);
    } catch (Exception e) {
        return false;
    }
}
```

Creating an XML Tree
The listing below loads an XML data object as nodes in a tree.
// access required XMLSpy Java-COM classes
import com.altova.automation.XMLSpy.XMLData;

// access AWT and Swing components
import java.awt.*;
import javax.swing.*;
import javax.swing.tree.*;

/**
  * A simple example of a tree control loading the structure from an XMLData object.
  * The class receives an XMLData object, loads its nodes in a JTree, and prepares
  * for modal activation.
  * Feel free to modify and extend this sample.
  *
  * @author Altova GmbH
  */
class XMLTreeDialog extends JDialog {

  /**
    * The tree control
    */
  private JTree myTree;

  /**
    * Root node of the tree control
    */
  private DefaultMutableTreeNode top;

  /**
    * Constructor that prepares the modal dialog containing the filled tree control
    * @param xml The data to be displayed in the tree
    * @param parent Parent frame
    */
  public XMLTreeDialog( XMLData xml, Frame parent ) {
    super( parent, "XML tree", true );
    // Arrange controls in the dialog
top = new DefaultMutableTreeNode("root");
myTree = new JTree(top);
setContentPane( new JScrollPane( myTree ) );
    // Build up the tree
fillTree( top, xml );
myTree.expandRow( 0 );
  }

  /**
    * Loads the nodes of an XML element under a given tree node
    * @param node Target tree node
    * @param elem Source XML element
    */
  private void fillTree( DefaultMutableTreeNode node, XMLData elem) {
  }
}
try
{
    // There are several ways to iterate through child elements: either
    // using the getFirstChild/getNextChild,
    // or by incrementing an index up to countChildren and calling getChild
    // [as shown below].
    // If you only want to get children of one kind, you should use
    countChildrenKind/getChildKind,
    // or provide a kind to the getFirstChild before iterating with the
    getNextChild.
    int nSize = elem.countChildren();
    for ( int i = 0 ; i < nSize ; ++i)
    {
        // Create a new tree node for each child element, and continue
        recursively
        XMLData newElem = elem.getChild(i) ;
        DefaultMutableTreeNode newNode = new
        DefaultMutableTreeNode( newElem.getName() ) ;
        node.add( newNode ) ;
        fillTree( newNode, newElem ) ;
    }
}

catch (Exception e)
{
    e.printStackTrace();
}
4.6 Command Reference

This section lists the names and identifiers of all menu commands that are available within Authentic Desktop. Every sub-section lists the commands from the corresponding top-level menu of Authentic Desktop. The command tables are organized as follows:

- The "Menu Item" column shows the command's menu text as it appears in Authentic Desktop, to make it easier for you to identify the functionality behind the command.
- The "Command Name" column specifies the string that can be used to get an icon with the same name from ActiveXImages folder of the Authentic Desktop installation directory.
- The "ID" column shows the numeric identifier of the column that must be supplied as argument to methods which execute or query this command.

To execute a command, use the AuthenticDesktopControl Exec or the AuthenticDesktopControlDocument.Exec methods. To query the status of a command, use the AuthenticDesktopControl.QueryStatus or AuthenticDesktopControlDocument.QueryStatus methods.

Depending on the edition of Authentic Desktop you have installed, some of these commands might not be supported.
4.6.1 "File" Menu

The "File" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>New...</td>
<td>ID_FILE_NEW</td>
<td>57600</td>
</tr>
<tr>
<td>Open...</td>
<td>ID_FILE_OPEN</td>
<td>57601</td>
</tr>
<tr>
<td>Reload</td>
<td>IDC_FILE_RELOAD</td>
<td>34065</td>
</tr>
<tr>
<td>Encoding...</td>
<td>IDC.Encoding</td>
<td>34061</td>
</tr>
<tr>
<td>Close</td>
<td>ID_FILE_CLOSE</td>
<td>57602</td>
</tr>
<tr>
<td>Close All</td>
<td>IDC_CLOSE_ALL</td>
<td>34050</td>
</tr>
<tr>
<td>Close All But Active</td>
<td>IDC_CLOSE_OTHERS</td>
<td>34271</td>
</tr>
<tr>
<td>Save</td>
<td>ID_FILE_SAVE</td>
<td>57603</td>
</tr>
<tr>
<td>Save As...</td>
<td>ID_FILE_SAVE_AS</td>
<td>57604</td>
</tr>
<tr>
<td>Save All</td>
<td>ID_FILE_SAVE_ALL</td>
<td>34208</td>
</tr>
<tr>
<td>Send by Mail...</td>
<td>ID_FILE_SEND_MAIL</td>
<td>57612</td>
</tr>
<tr>
<td>Print...</td>
<td>ID_FILE_PRINT</td>
<td>57607</td>
</tr>
<tr>
<td>Print Preview</td>
<td>IDC_PRINT_PREVIEW</td>
<td>34104</td>
</tr>
<tr>
<td>Print Setup...</td>
<td>ID_FILE_PRINT_SETTP</td>
<td>57606</td>
</tr>
<tr>
<td>Recent File</td>
<td>ID_FILE_MRU_FILE1</td>
<td>57616</td>
</tr>
<tr>
<td>Exit</td>
<td>ID_APP_EXIT</td>
<td>57665</td>
</tr>
</tbody>
</table>
4.6.2 "Edit" Menu

The "Edit" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>ID_EDIT_UNDO</td>
<td>57643</td>
</tr>
<tr>
<td>Redo</td>
<td>ID_EDIT_REDO</td>
<td>57644</td>
</tr>
<tr>
<td>Cut</td>
<td>ID_EDIT_CUT</td>
<td>57635</td>
</tr>
<tr>
<td>Copy</td>
<td>ID_EDIT_COPY</td>
<td>57634</td>
</tr>
<tr>
<td>Paste</td>
<td>ID_EDIT_PASTE</td>
<td>57637</td>
</tr>
<tr>
<td>Delete</td>
<td>ID_EDIT_CLEAR</td>
<td>57632</td>
</tr>
<tr>
<td>Select All</td>
<td>ID_EDIT_SELECT_ALL</td>
<td>57642</td>
</tr>
<tr>
<td>Find…</td>
<td>ID_EDIT_FIND</td>
<td>57636</td>
</tr>
<tr>
<td>Find Next</td>
<td>ID_EDIT_REPEAT</td>
<td>57640</td>
</tr>
<tr>
<td>Replace…</td>
<td>ID_EDIT_REPLACE</td>
<td>57641</td>
</tr>
</tbody>
</table>
### 4.6.3 "Project" Menu

The "Project" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Project</td>
<td>IDC_ICPROJECTGUI_NEW</td>
<td>37200</td>
</tr>
<tr>
<td>Open Project...</td>
<td>IDC_ICPROJECTGUI_OPEN</td>
<td>37201</td>
</tr>
<tr>
<td>Reload Project</td>
<td>IDC_ICPROJECTGUI_RELOAD</td>
<td>37202</td>
</tr>
<tr>
<td>Close Project</td>
<td>IDC_ICPROJECTGUI_CLOSE</td>
<td>37203</td>
</tr>
<tr>
<td>Save Project</td>
<td>IDC_ICPROJECTGUI_SAVE</td>
<td>37204</td>
</tr>
<tr>
<td>Save Project As...</td>
<td>IDC_ICPROJECTGUI_SAVE_AS</td>
<td>37207</td>
</tr>
<tr>
<td>Enable Source Control</td>
<td>ID_SCC_ENABLE</td>
<td>38602</td>
</tr>
<tr>
<td>Add Files to Project...</td>
<td>IDC_ICPROJECTGUI_ADD_FILES_TO_PROJECT</td>
<td>37205</td>
</tr>
<tr>
<td>Add Global Resource to Project...</td>
<td>IDC_ICPROJECTGUI_ADD_GLOBALRESOURCE_TO_PROJECT</td>
<td>37239</td>
</tr>
<tr>
<td>Add URL to Project...</td>
<td>IDC_ICPROJECTGUI_ADD_URL_TO_PROJECT</td>
<td>37206</td>
</tr>
<tr>
<td>Add Active File to Project</td>
<td>IDC_ICPROJECTGUI_ADD_ACTIVE_FILE_TO_PROJECT</td>
<td>37208</td>
</tr>
<tr>
<td>Add Active and Related Files to Project</td>
<td>IDC_ICPROJECTGUI_ADD_ACTIVEANDRELATEDFILES_TO_PROJECT</td>
<td>37209</td>
</tr>
<tr>
<td>Add Project Folder to Project...</td>
<td>IDC_ICPROJECTGUI_ADD_FOLDER_TO_PROJECT</td>
<td>37210</td>
</tr>
<tr>
<td>Add External Folder to Project...</td>
<td>IDC_ICPROJECTGUI_ADD_EXT_FOLDER_TO_PROJECT</td>
<td>37211</td>
</tr>
<tr>
<td>Add External Web Folder to Project...</td>
<td>IDC_ICPROJECTGUI_ADD_EXT_URL_FOLDER_TO_PROJECT</td>
<td>37212</td>
</tr>
<tr>
<td>Script settings...</td>
<td>IDC_PROJECT_SCRIPT_SETTINGS</td>
<td>34136</td>
</tr>
<tr>
<td>Properties...</td>
<td>IDC_ICPROJECTGUI_PROJECT_PROPERTIES</td>
<td>37223</td>
</tr>
<tr>
<td>Recent Project</td>
<td>IDC_ICPROJECTGUI_RECENT</td>
<td>37224</td>
</tr>
</tbody>
</table>
### 4.6.4 "XML" Menu

The "XML" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Well-Formedness</td>
<td>IDC_CHECK_WELL_FORM</td>
<td>34049</td>
</tr>
<tr>
<td>Validate XML</td>
<td>IDC_VALIDATE</td>
<td>32954</td>
</tr>
</tbody>
</table>
4.6.5 "XSL/XQuery" Menu

The "XSL/XQuery" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSL Transformation</td>
<td>IDC_TRANSFORM_XSL</td>
<td>33006</td>
</tr>
<tr>
<td>XSL-FO Transformation</td>
<td>IDC_TRANSFORM_XSLFO</td>
<td>33007</td>
</tr>
<tr>
<td>XSL Parameters / XQuery Variables…</td>
<td>IDC_TRANSFORM_XSL_PARAMS</td>
<td>33008</td>
</tr>
</tbody>
</table>
4.6.6 "Authentic" Menu

The "Authentic" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Document...</td>
<td>IDC_AUTHENTIC_NEW_FILE</td>
<td>34036</td>
</tr>
<tr>
<td>Edit Database Data...</td>
<td>IDC_AUTHENTIC_EDIT_DB</td>
<td>34035</td>
</tr>
<tr>
<td>Edit StyleVision Stylesheet</td>
<td>IDC_EDIT_SPS</td>
<td>34060</td>
</tr>
<tr>
<td>Select New Row with XML Data for Editing...</td>
<td>IDC_CHANGE_WORKING_DB_XML_CELL</td>
<td>32861</td>
</tr>
<tr>
<td>XML Signature...</td>
<td>IDC_AUTHENTICGUI_XMLSIGNATURE</td>
<td>32862</td>
</tr>
<tr>
<td>Define XML Entities...</td>
<td>IDC_DEFINE_ENTITIES</td>
<td>32805</td>
</tr>
<tr>
<td>Hide Markup</td>
<td>IDC_MARKUP_HIDE</td>
<td>32855</td>
</tr>
<tr>
<td>Show Small Markup</td>
<td>IDC_MARKUP_SMALL</td>
<td>32858</td>
</tr>
<tr>
<td>Show Large Markup</td>
<td>IDC_MARKUP_LARGE</td>
<td>32856</td>
</tr>
<tr>
<td>Show Mixed Markup</td>
<td>IDC_MARKUP_MIXED</td>
<td>32857</td>
</tr>
<tr>
<td>Toggle Bold</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_TOGGLE_BOLD</td>
<td>32813</td>
</tr>
<tr>
<td>Toggle Italic</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_TOGGLE_ITALIC</td>
<td>32814</td>
</tr>
<tr>
<td>Toggle Underline</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_TOGGLE_UNDERLINE</td>
<td>32815</td>
</tr>
<tr>
<td>Toggle Strikethrough</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_TOGGLE_STRIKETHROUGH</td>
<td>32816</td>
</tr>
<tr>
<td>Foreground Color</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_COLOR_FOREGROUND</td>
<td>32824</td>
</tr>
<tr>
<td>Background Color</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_COLOR_BACKGROUND</td>
<td>32830</td>
</tr>
<tr>
<td>Align Left</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_ALIGN_LEFT</td>
<td>32818</td>
</tr>
<tr>
<td>Center</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_ALIGN_CENTER</td>
<td>32819</td>
</tr>
<tr>
<td>Align Right</td>
<td>IDC_AUTHENTICGUI_RICHEDIT_ALIGN_RIGHT</td>
<td>32820</td>
</tr>
<tr>
<td>Append Row</td>
<td>IDC_ROW_APPEND</td>
<td>32806</td>
</tr>
<tr>
<td>Insert Row</td>
<td>IDC_ROW_INSERT</td>
<td>32809</td>
</tr>
<tr>
<td>Menu item</td>
<td>Command name</td>
<td>ID</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Duplicate Row</td>
<td>IDC_ROW_DUPLICATE</td>
<td>32808</td>
</tr>
<tr>
<td>Move Row Up</td>
<td>IDC_ROW_MOVE_UP</td>
<td>32811</td>
</tr>
<tr>
<td>Move Row Down</td>
<td>IDC_ROW_MOVE_DOWN</td>
<td>32810</td>
</tr>
<tr>
<td>Delete Row</td>
<td>IDC_ROW_DELETE</td>
<td>32807</td>
</tr>
<tr>
<td>Generate an HTML document</td>
<td>IDC_PXF_GENERATE_HTML</td>
<td>34283</td>
</tr>
<tr>
<td>Generate an RTF document</td>
<td>IDC_PXF_GENERATE_RTF</td>
<td>34284</td>
</tr>
<tr>
<td>Generate a PDF document</td>
<td>IDC_PXF_GENERATE_PDF</td>
<td>34285</td>
</tr>
<tr>
<td>Generate a Word 2007+ document</td>
<td>IDC_PXF_GENERATE_DOCX</td>
<td>34286</td>
</tr>
<tr>
<td>Trusted Locations...</td>
<td>IDC_TRUSTED_LOCATIONS</td>
<td>34288</td>
</tr>
</tbody>
</table>
4.6.7 "View" Menu

The "View" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentic View</td>
<td>IDC_VIEWCONTENT</td>
<td>34177</td>
</tr>
<tr>
<td>Browser View</td>
<td>IDC_VIEW_BROWSER</td>
<td>34176</td>
</tr>
<tr>
<td>Text View Settings</td>
<td>IDC_TEXTVIEW_SETTINGS</td>
<td>34119</td>
</tr>
</tbody>
</table>
### "Browser" Menu

The "Browser" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back</td>
<td>IDC_STEP_BACK</td>
<td>32958</td>
</tr>
<tr>
<td>Forward</td>
<td>IDC_STEP_FORWARD</td>
<td>32957</td>
</tr>
<tr>
<td>Stop</td>
<td>IDC_BROWSER_STOP</td>
<td>34047</td>
</tr>
<tr>
<td>Refresh</td>
<td>IDC_BROWSER_REFRESH</td>
<td>34046</td>
</tr>
<tr>
<td>Largest</td>
<td>IDC_BROWSER_FONT_LARGEST</td>
<td>34041</td>
</tr>
<tr>
<td>Larger</td>
<td>IDC_BROWSER_FONT_LARGE</td>
<td>34040</td>
</tr>
<tr>
<td>Medium</td>
<td>IDC_BROWSER_FONT_MEDIUM</td>
<td>34042</td>
</tr>
<tr>
<td>Smaller</td>
<td>IDC_BROWSER_FONT_SMALL</td>
<td>34043</td>
</tr>
<tr>
<td>Smallest</td>
<td>IDC_BROWSER_FONT_SMALLEST</td>
<td>34044</td>
</tr>
</tbody>
</table>
### 4.6.9 "Tools" Menu

The "Tools" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spelling…</td>
<td>IDC_SPELL_CHECK</td>
<td>34154</td>
</tr>
<tr>
<td>Spelling Options…</td>
<td>IDC_SPELL_OPTIONS</td>
<td>34155</td>
</tr>
<tr>
<td>Scripting Editor…</td>
<td>IDC_SCRIPTFORMEDITOR_EDIT_PROJECT</td>
<td>39666</td>
</tr>
<tr>
<td>none</td>
<td>ID_SCRIPTFORMEDITOR_EXECUTE_MACRO_MENU_UPDATE</td>
<td>39600</td>
</tr>
<tr>
<td>Global Resources</td>
<td>IDC_GLOBALRESOURCES</td>
<td>37401</td>
</tr>
<tr>
<td>Customize…</td>
<td>IDC_APP_TOOLS_CUSTOMIZE</td>
<td>32959</td>
</tr>
<tr>
<td>Options…</td>
<td>IDC_SETTINGS</td>
<td>34133</td>
</tr>
<tr>
<td></td>
<td>ID_SCRIPTING_MACROITEMS</td>
<td>34249</td>
</tr>
</tbody>
</table>
### 4.6.10 "Window" Menu

The "Window" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascade</td>
<td>ID_WINDOWCASCADE</td>
<td>57650</td>
</tr>
<tr>
<td>Tile horizontally</td>
<td>ID_WINDOWTILEHORZ</td>
<td>57651</td>
</tr>
<tr>
<td>Tile vertically</td>
<td>ID_WINDOWTILEVERT</td>
<td>57652</td>
</tr>
<tr>
<td>Project window</td>
<td>IDC_PROJECTWINDOW</td>
<td>34128</td>
</tr>
<tr>
<td>Info window</td>
<td>IDC_INFOWINDOW</td>
<td>34085</td>
</tr>
<tr>
<td>Entry Helpers</td>
<td>IDC_ENTRYHELPERS</td>
<td>34062</td>
</tr>
<tr>
<td>Output windows</td>
<td>IDC_OUTPUTDIALOGBARS</td>
<td>34004</td>
</tr>
<tr>
<td>Project and Entry Helpers</td>
<td>IDC_PROJECTENTRYHELPERS</td>
<td>34006</td>
</tr>
<tr>
<td>All on/off</td>
<td>IDC_ALLBARS</td>
<td>34031</td>
</tr>
</tbody>
</table>
### 4.6.11 "Help" Menu

The "Help" menu has the following commands:

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Command name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents...</td>
<td>IDC_HELP_CONTENTS</td>
<td>32966</td>
</tr>
<tr>
<td>Index...</td>
<td>IDC_HELP_INDEX</td>
<td>32967</td>
</tr>
<tr>
<td>Search...</td>
<td>IDC_HELP_SEARCH</td>
<td>32969</td>
</tr>
<tr>
<td>Keyboard Map...</td>
<td>IDC_HELP_KEYMAPDLG</td>
<td>32968</td>
</tr>
<tr>
<td>Software Activation...</td>
<td>IDC_ACTIVATION</td>
<td>32970</td>
</tr>
<tr>
<td>Order Form...</td>
<td>IDC_OPEN_ORDER_PAGE</td>
<td>32971</td>
</tr>
<tr>
<td>Registration...</td>
<td>IDC_REGISTRATION</td>
<td>32972</td>
</tr>
<tr>
<td>Check for Updates...</td>
<td>IDC_CHECK_FOR_UPDATES</td>
<td>32973</td>
</tr>
<tr>
<td>XMLSpy Product Comparison...</td>
<td>IDC_PRODUCT_COMPARISON</td>
<td>32955</td>
</tr>
<tr>
<td>Support Center...</td>
<td>IDC_OPEN_SUPPORT_PAGE</td>
<td>32961</td>
</tr>
<tr>
<td>FAQ on the Web...</td>
<td>IDC_OPEN_FAQ_PAGE</td>
<td>32962</td>
</tr>
<tr>
<td>Download Components and Free Tools...</td>
<td>IDC_OPEN_COMPONENTS_PAGE</td>
<td>32963</td>
</tr>
<tr>
<td>Authentic on the Internet...</td>
<td>IDC_OPEN_HOME_PAGE</td>
<td>32964</td>
</tr>
<tr>
<td>Authentic Training...</td>
<td>IDC_OPEN_TRAINING_PAGE</td>
<td>32965</td>
</tr>
<tr>
<td>About Authentic...</td>
<td>ID_APP_ABOUT</td>
<td>57664</td>
</tr>
</tbody>
</table>
4.7 Object Reference

Objects:
Authentic DesktopCommand
Authentic DesktopCommands
AuthenticDesktopControl
AuthenticDesktopControlDocument
AuthenticDesktopControlPlaceHolder

To give access to standard Authentic Desktop functionality, objects of the Authentic Desktop automation interface can be accessed as well. See AuthenticDesktopControl.Application, AuthenticDesktopControlDocument.Document and AuthenticDesktopControlPlaceHolder.Project for more information.
4.7.1 Authentic DesktopCommand

Properties:
- ID
- Label
- Name
- IsSeparator
- ToolTip
- StatusText
- SubCommands

Description:
A command object can be one of the following: an executable command, a command container (for example, a menu, submenu, or toolbar), or a menu separator. To determine what kind of information is stored in the current command object, query its ID, IsSeparator, and SubCommands properties, as follows.

<table>
<thead>
<tr>
<th>The Command object is...</th>
<th>When...</th>
</tr>
</thead>
</table>
| An executable command   | • ID is greater than zero  
                          | • IsSeparator is false  
                          | • SubCommands is empty |
| A command container     | • ID is zero  
                          | • IsSeparator is true  
                          | • SubCommands contains a collection of Command objects. |
| Separator               | • ID is zero  
                          | • IsSeparator is true |

Accelerator

Property: Accelerator as string

Description:
Returns the accelerator key defined for the command. If the command has no accelerator key assigned, this property returns the empty string. The string representation of the accelerator key has the following format:

```
[ALT+] [CTRL+] [SHIFT+] key
```

Where key is converted using the Windows Platform SDK function GetKeyNameText.

ID

Property: ID as long

Description:
This property gets the unique identifier of the command. A command's ID is required to execute the command (using Exec) or query its status (using QueryStatus). If the command is a container for other commands (for example, a top-level menu), or a separator, the ID is 0.
IsSeparator

Property: IsSeparator as boolean

Description:
The property returns true if the command object is a menu separator; false otherwise. See also Command.

Label

Property: Label as string

Description:
This property gets the text of the command as it is displayed in the graphical user interface of Authentic Desktop. If the command is a separator, "Label" is an empty string. This property may also return an empty string for some toolbar commands that do not have any GUI text associated with them.

Name

Property: Name as string

Description:
This property gets the unique name of the command. This value can be used to get the icon file of the command, where it is available. The available icon files can be found in the folder <ApplicationFolder>\Examples\ActiveX\Images of your Authentic Desktop installation.

StatusText

Property: Label as string

Description:
The status text is the text shown in the status bar of Authentic Desktop when the command is selected. It applies only to command objects that are not separators or containers of other commands; otherwise, the property is an empty string.

SubCommands

Property: SubCommands as Commands

Description:
The SubCommands property gets the collection of Command objects that are sub-commands of the current command. The property is applicable only to commands that are containers for other commands (menus, submenus, or toolbars). Such container commands have the ID set to 0, and the IsSeparator property set to false.

ToolTip

Property: ToolTip as string

Description:
This property gets the text that is shown as a tool-tip for each command. If the command does
not have a tooltip text, the property returns an empty string.
4.7.2 **Authentic DesktopCommands**

Properties:

- Count
- Item

Description:
Collection of Command objects to get access to command labels and IDs of the AuthenticDesktopControl. Those commands can be executed with the Exec method and their status can be queried with QueryStatus.

**Count**

*Property:* Count as long

*Description:*
Number of Command objects on this level of the collection.

**Item**

*Property:* Item (n as long) as Command

*Description:*
Gets the command with the index n in this collection. Index is 1-based.
4.7.3 **AuthenticDesktopControl**

**Properties:**
- IntegrationLevel
- Appearance
- Application
- BorderStyle
- CommandsList
- CommandsStructure (deprecated)
- EnableUserPrompts
- MainMenu
- Toolbars

**Methods:**
- Open
- Exec
- QueryStatus

**Events:**
- OnUpdateCmdUI
- OnOpenedOrFocused
- OnCloseEditingWindow
- OnFileChangedAlert
- OnContextChanged
- OnDocumentOpened
- OnValidationWindowUpdated

This object is a complete ActiveX control and should only be visible if the Authentic Desktop library is used in the Application Level mode.

**Properties**

The following properties are defined:
- IntegrationLevel
- EnableUserPrompts
- Appearance
- BorderStyle

Command related properties:
- CommandsList
- MainMenu
- Toolbars
- CommandsStructure (deprecated)

Access to AuthenticDesktopAPI:
- Application

**Appearance**

**Property:** Appearance as **short**

**Dispatch Id:** -520
**Description:**
A value not equal to 0 displays a client edge around the control. Default value is 0.

**Application**

*Property:* Application as `Application`

*Dispatch Id:* 1

**Description:**
The `Application` property gives access to the `Application` object of the complete Authentic Desktop automation server API. The property is read-only.

**BorderStyle**

*Property:* BorderStyle as `short`

*Dispatch Id:* -504

**Description:**
A value of 1 displays the control with a thin border. Default value is 0.

**CommandsList**

*Property:* CommandList as `Commands` (read-only)

*Dispatch Id:* 1004

**Description:**
This property returns a flat list of all commands defined available with AuthenticDesktopControl. To get commands organized according to their menu structure, use `MainMenu`. To get toolbar commands, use `Toolbars`.

```csharp
public void GetAllAuthenticCommands()
{
    // Get all commands from the Authentic ActiveX control assigned to the current form
    AuthenticControlLib.XMLSpyCommands commands =
    this.axAuthenticDesktopControl1.CommandList;
    // Loop through all commands
    for (int i = 0; i < commands.Count; i++)
    {
        // Get each command by index and output it to the console
        AuthenticControlLib.XMLSpyCommand cmd =
        axAuthenticDesktopControl1.CommandList[i];
        Console.WriteLine("(0) {1} {2}".cmd.ID, cmd.Name, 
        cmd.Label.Replace("&", "");
    }

```
C# example

EnableUserPrompts

Property: EnableUserPrompts as boolean

Dispatch Id: 1006

Description:
Setting this property to false, disables user prompts in the control. The default value is true.

IntegrationLevel

Property: IntegrationLevel as ICAActiveXIntegrationLevel

Dispatch Id: 1000

Description:
The IntegrationLevel property determines the operation mode of the control. See also Integration at Application Level and Integration at Document Level for more information.

Note: It is important to set this property immediately after the creation of the AuthenticDesktopControl object.

MainMenu

Property: MainMenu as Command (read-only)

Dispatch Id: 1003

Description:
This property provides information about the structure and commands available in the AuthenticDesktopControl main menu, as a Command object. The Command object contains all available submenus of Authentic Desktop (for example "File", "Edit", "View" etc.). To access the submenu objects, use the SubCommands property of the MainMenu property. Each submenu is also a Command object. For each submenu, you can then further iterate through their SubCommands property in order to get their corresponding child commands and separators (this technique may be used, for example, to create the application menu programmatically). Note that some menu commands act as containers ("parents") for other menu commands, in which case they also have a SubCommands property. To get the structure of all menu commands programmatically, you will likely need to create a recursive function.

```csharp
public void GetAuthenticMenus()
{
}
```
Programmers' Reference

ActiveX Integration

755

// Get the main menu from the Authentic ActiveX control assigned to the
current form
AuthenticControlLib.XMLSpyCommand mainMenu =
this.axAuthenticDesktopControl1.MainMenu;
// Loop through entries of the main menu (e.g. File, Edit, etc.)
for (int i = 0; i < mainMenu.SubCommands.Count; i++)
{
AuthenticControlLib.XMLSpyCommand menu = mainMenu.SubCommands[i];
Console.WriteLine("{0} menu has {1} children items (including
separators)", menu.Label.Replace("&", ""), menu.SubCommands.Count);
}
}
C# example

Toolbars
Property: Toolbars as Commands (read-only)
Dispatch Id: 1005
Description:
This property provides information about the structure of AuthenticDesktopControl toolbars, as a
Command object. The Command object contains all available toolbars of Authentic Desktop. To
access the toolbars, use the SubCommands property of the Toolbars property. Each toolbar is
also a Command object. For each toolbar, you can then further iterate through their SubCommands
property in order to get their commands (this technique may be used, for example, to create the
application's toolbars programmatically).

public void GetAuthenticToolbars()
{
// Get the application toolbars from the Authentic ActiveX control
assigned to the current form
AuthenticControlLib.XMLSpyCommands toolbars =
this.axAuthenticDesktopControl1.Toolbars;
// Iterate through all toolbars
for (int i = 0; i < toolbars.Count; i++)
{
AuthenticControlLib.XMLSpyCommand toolbar = toolbars[i];
Console.WriteLine();
Console.WriteLine("The toolbar \"{0}\" has the following commands:",
toolbar.Label);
// Iterate through all commands of this toolbar
for (int j = 0; j < toolbar.SubCommands.Count; j++)
{
AuthenticControlLib.XMLSpyCommand cmd = toolbar.SubCommands[j];
// Output only command objects that are not separators
if ( ! cmd.IsSeparator)
{
Console.WriteLine("{0}, {1}, {2}", cmd.ID, cmd.Name,

© 2019 Altova Gmb H

Altova Authentic 2019 Desktop Enterprise Edition


C# example

Methods

The following methods are defined:

Open
Exec
QueryStatus

Exec

Method: Exec (nCmdID as long) as boolean

Dispatch Id: 6

Description:
This method calls the Authentic Desktop command with the ID nCmdID. If the command can be executed, the method returns true. To get a list of all available commands, use CommandsList. To retrieve the status of any command, use QueryStatus.

Open

Method: Open (strFilePath as string) as boolean

Dispatch Id: 5

Description:
The result of the method depends on the extension passed in the argument strFilePath. If the file extension is .sps, a new document is opened. If the file extension is .svp, the corresponding project is opened. If a different file extension is passed into the method, the control tries to load the file as a new component into the active document.

Do not use this method to load documents or projects when using the control in document-level integration mode. Instead, use AuthenticDesktopControlDocument.Open and AuthenticDesktopControlPlaceHolder.OpenProject.

QueryStatus

Method: QueryStatus (nCmdID as long) as long

Dispatch Id: 7

Description:
QueryStatus returns the enabled/disabled and checked/unchecked status of the command specified by nCmdID. The status is returned as a bit mask.
Bit   Value  Name          Meaning
-----------------------------------------------
0     1      Supported     Set if the command is supported.
1     2      Enabled       Set if the command is enabled (can be executed).
2     4      Checked       Set if the command is checked.

This means that if `QueryStatus` returns 0 the command ID is not recognized as a valid Authentic Desktop command. If `QueryStatus` returns a value of 1 or 5, the command is disabled.

Events
The AuthenticDesktopControl ActiveX control provides the following connection point events:

- `OnUpdateCmdUI`
- `OnOpenedOrFocused`
- `OnCloseEditingWindow`
- `OnFileChangedAlert`
- `OnContextChanged`
- `OnDocumentOpened`
- `OnValidationWindowUpdated`

**OnCloseEditingWindow**

*Event:* `OnCloseEditingWindow (i_strFilePath as String) as boolean`

*Dispatch Id:* 1002

*Description:* This event is triggered when Authentic Desktop needs to close an already open document. As an answer to this event, clients should close the editor window associated with `i_strFilePath`. Returning true from this event indicates that the client has closed the document. Clients can return false if no specific handling is required and AuthenticDesktopControl should try to close the editor and destroy the associated document control.

**OnContextChanged**

*Event:* `OnContextChanged (i_strContextName as String, i_bActive as bool) as bool`

*Dispatch Id:* 1004

*Description:* This event is not used in Authentic Desktop.

**OnDocumentOpened**

*Event:* `OnDocumentOpened (objDocument as Document)`

*Dispatch Id:* 1
Description:
This event is triggered whenever a document is opened. The argument `objDocument` is a `Document` object from the Authentic Desktop automation interface and can be used to query for more details about the document, or perform additional operations. When integrating on document-level, it is often better to use the event `AuthenticDesktopControlDocument.OnDocumentOpened` instead.

**OnFileChangedAlert**

**Event**: `OnFileChangedAlert (i_strFilePath as String) as bool`

**Dispatch Id**: 1001

**Description**:
This event is triggered when a file loaded with AuthenticDesktopControl is changed on the hard disk by another application. Clients should return true, if they handled the event, or false, if Authentic Desktop should handle it in its customary way, i.e. prompting the user for reload.

**OnLicenseProblem**

**Event**: `OnLicenseProblem (i_strLicenseProblemText as String)`

**Dispatch Id**: 1005

**Description**:
This event is triggered when AuthenticDesktopControl detects that no valid license is available for this control. In case of restricted user licenses this can happen some time after the control has been initialized. Integrators should use this event to disable access to this control's functionality. After returning from this event, the control will block access to its functionality (e.g. show empty windows in its controls and return errors on requests).

**OnOpenedOrFocused**

**Event**: `OnOpenedOrFocused (i_strFilePath as String, i_bOpenWithThisControl as bool)`

**Dispatch Id**: 1000

**Description**:
When integrating at application level, this event informs clients that a document has been opened, or made active by Authentic Desktop.

When integrating at document level, this event instructs the client to open the file `i_strFilePath` in a document window. If the file is already open, the corresponding document window should be made the active window.

If `i_bOpenWithThisControl` is true, the document must be opened with AuthenticDesktopControl, since internal access is required. Otherwise, the file can be opened with different editors.
OnToolWindowUpdated

Event: OnToolWindowUpdated(pToolWnd as long)

Dispatch Id: 1006

Description:
This event is triggered when the tool window is updated.

OnUpdateCmdUI

Event: OnUpdateCmdUI()

Dispatch Id: 1003

Description:
Called frequently to give integrators a good opportunity to check status of Authentic Desktop commands using AuthenticDesktopControl.QueryStatus. Do not perform long operations in this callback.

OnValidationWindowUpdated

Event: OnValidationWindowUpdated()

Dispatch Id: 3

Description:
This event is triggered whenever the validation output window is updated with new information.
4.7.4 **AuthenticDesktopControlDocument**

**Properties:**
- Appearance
- BorderStyle
- Document
- IsModified
- Path
- ReadOnly

**Methods:**
- Exec
- New
- Open
- QueryStatus
- Reload
- Save
- SaveAs

**Events:**
- OnDocumentOpened
- OnDocumentClosed
- OnModifiedFlagChanged
- OnContextChanged
- OnFileChangedAlert
- OnActivate

If the AuthenticDesktopControl is integrated in the Document Level mode each document is displayed in an own object of type **AuthenticDesktopControlDocument**. The **AuthenticDesktopControlDocument** contains only one document at the time but can be reused to display different files one after another.

This object is a complete ActiveX control.

**Properties**
The following properties are defined:
- ReadOnly
- IsModified
- Path
- Appearance
- BorderStyle

Access to AuthenticDesktopAPI:
- Document

**Appearance**

*Property:* Appearance as short

*Dispatch Id:* -520

*Description:*
A value not equal to 0 displays a client edge around the document control. Default value is 0.

**BorderStyle**

*Property:* `BorderStyle` as `short`

*Dispatch Id:* -504

*Description:* A value of 1 displays the control with a thin border. Default value is 0.

**Document**

*Property:* `Document` as `Document`

*Dispatch Id:* 1

*Description:* The `Document` property gives access to the `Document` object of the Authentic Desktop automation server API. This interface provides additional functionality which can be used with the document loaded in the control. The property is read-only.

**IsModified**

*Property:* `IsModified` as `boolean` (read-only)

*Dispatch Id:* 1006

*Description:* `IsModified` is `true` if the document content has changed since the last open, reload or save operation. It is `false`, otherwise.

**Path**

*Property:* `Path` as `string`

*Dispatch Id:* 1005

*Description:* Sets or gets the full path name of the document loaded into the control.

**ReadOnly**

*Property:* `ReadOnly` as `boolean`

*Dispatch Id:* 1007
Description:
Using this property you can turn on and off the read-only mode of the document. If `ReadOnly` is `true` it is not possible to do any modifications.

Methods

The following methods are defined:

Document handling:
**New**
**Open**
**Reload**
**Save**
**SaveAs**

Command Handling:
**Exec**
**QueryStatus**

**Exec**

*Method:* `Exec (nCmdID as long) as boolean`

*Dispatch Id:* 8

*Description:* `Exec` calls the Authentic Desktop command with the ID `nCmdID`. If the command can be executed, the method returns `true`. This method should be called only if there is currently an active document available in the application.

To get commands organized according to their menu structure, use the `MainMenu` property of `AuthenticDesktopControl`. To get toolbar commands, use the `Toolbars` property of the `AuthenticDesktopControl`.

**New**

*Method:* `New () as boolean`

*Dispatch Id:* 1000

*Description:* This method initializes a new document inside the control.

**Open**

*Method:* `Open (strFileName as string) as boolean`

*Dispatch Id:* 1001

*Description:*
Open loads the file \texttt{strFileName} as the new document into the control.

\textbf{QueryStatus}

\textit{Method:} QueryStatus (\texttt{nCmdID as long}) as \texttt{long}

\textit{Dispatch Id:} 9

\textit{Description:}

\texttt{QueryStatus} returns the enabled/disabled and checked/unchecked status of the command specified by \texttt{nCmdID}. The status is returned as a bit mask.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Supported</td>
<td>Set if the command is supported.</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Enabled</td>
<td>Set if the command is enabled (can be executed).</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Checked</td>
<td>Set if the command is checked.</td>
</tr>
</tbody>
</table>

This means that if \texttt{QueryStatus} returns 0 the command ID is not recognized as a valid Authentic Desktop command. If \texttt{QueryStatus} returns a value of 1 or 5 the command is disabled. The client should call the \texttt{QueryStatus} method of the document control if there is currently an active document available in the application.

\textbf{Reload}

\textit{Method:} Reload () as \texttt{boolean}

\textit{Dispatch Id:} 1002

\textit{Description:}

\texttt{Reload} updates the document content from the file system.

\textbf{Save}

\textit{Method:} Save () as \texttt{boolean}

\textit{Dispatch Id:} 1003

\textit{Description:}

\texttt{Save} saves the current document at the location \texttt{Path}.

\textbf{SaveAs}

\textit{Method:} SaveAs (\texttt{strFileName as string}) as \texttt{boolean}

\textit{Dispatch Id:} 1004

\textit{Description:}

\texttt{SaveAs} sets \texttt{Path} to \texttt{strFileName} and then saves the document to this location.
Events
The AuthenticDesktopControlDocument ActiveX control provides following connection point events:

- **OnDocumentOpened**
- **OnDocumentClosed**
- **OnModifiedFlagChanged**
- **OnContextChanged**
- **OnFileChangedAlert**
- **OnActivate**
- **OnSetEditorTitle**

**OnActivate**
*Event:* OnActivate ()

*Dispatch Id:* 1005

*Description:* This event is triggered when the document control is activated, has the focus, and is ready for user input.

**OnContextChanged**
*Event:* OnContextChanged (i_strContextName as String, i_bActive as bool) as bool

*Dispatch Id:* 1004

*Description:* None

**OnDocumentClosed**
*Event:* OnDocumentClosed (objDocument as Document)

*Dispatch Id:* 1001

*Description:* This event is triggered whenever the document loaded into this control is closed. The argument objDocument is a Document object from the Authentic Desktop automation interface and should be used with care.

**OnDocumentOpened**
*Event:* OnDocumentOpened (objDocument as Document)

*Dispatch Id:* 1000

*Description:* This event is triggered whenever a document is opened in this control. The argument objDocument is a Document object from the Authentic Desktop automation interface, and can
be used to query for more details about the document, or perform additional operations.

**OnDocumentSaveAs**

**Event:** OnContextDocumentSaveAs (i_strFileName as String)

**Dispatch Id:** 1007

**Description:**
This event is triggered when this document gets internally saved under a new name.

**OnFileChangedAlert**

**Event:** OnFileChangedAlert () as bool

**Dispatch Id:** 1003

**Description:**
This event is triggered when the file loaded into this document control is changed on the hard disk by another application. Clients should return true, if they handled the event, or false, if Authentic Desktop should handle it in its customary way, i.e. prompting the user for reload.

**OnModifiedFlagChanged**

**Event:** OnModifiedFlagChanged (i_bIsModified as boolean)

**Dispatch Id:** 1002

**Description:**
This event gets triggered whenever the document changes between modified and unmodified state. The parameter i_bIsModified is true if the document contents differs from the original content, and false, otherwise.

**OnSetEditorTitle**

**Event:** OnSetEditorTitle ()

**Dispatch Id:** 1006

**Description:**
This event is being raised when the contained document is being internally renamed.
4.7.5 **AuthenticDesktopControlPlaceHolder**

**Properties available for all kinds of placeholder windows:**

- **PlaceholderWindowID**

**Properties for project placeholder window:**

- **Project**

**Methods for project placeholder window:**

- **OpenProject**
- **CloseProject**

The **AuthenticDesktopControlPlaceHolder** control is used to show the additional Authentic Desktop windows like Overview, Library or Project window. It is used like any other ActiveX control and can be placed anywhere in the client application.

**Properties**

The following properties are defined:

- **PlaceholderWindowID**

**Access to AuthenticDesktopAPI:**

- **Project**

**Label**

**Property:** Label as String (read-only)

**Dispatch Id:** 1001

**Description:**
This property gives access to the title of the placeholder. The property is read-only.

**PlaceholderWindowID**

**Property:** PlaceholderWindowID as **AuthenticDesktopControlPlaceHolderWindow**

**Dispatch Id:** 1

**Description:**
This property specifies which Authentic Desktop window should be displayed in the client area of the control. The PlaceholderWindowID can be set at any time to any valid value of the **AuthenticDesktopControlPlaceHolderWindow** enumeration. The control changes its state immediately and shows the new Authentic Desktop window.
**Project**

**Property:** Project as Project (read-only)

**Dispatch Id:** 2

**Description:**
The `Project` property gives access to the `Project` object of the Authentic Desktop automation server API. This interface provides additional functionality which can be used with the project loaded into the control. The property will return a valid project interface only if the placeholder window has `PlaceholderWindowID` with a value of `Authentic DesktopXProjectWindow (=3)`. The property is read-only.

**Methods**
The following method is defined:

*OpenProject*

*CloseProject*

**OpenProject**

**Method:** OpenProject (strFileName as string) as boolean

**Dispatch Id:** 3

**Description:**
`OpenProject` loads the file `strFileName` as the new project into the control. The method will fail if the placeholder window has a `PlaceholderWindowID` different to `XMLSpyXProjectWindow (=3)`.

**CloseProject**

**Method:** CloseProject ()

**Dispatch Id:** 4

**Description:**
`CloseProject` closes the project loaded the control. The method will fail if the placeholder window has a `PlaceholderWindowID` different to `Authentic DesktopXProjectWindow (=3)`.

**Events**
The `AuthenticDesktopControlPlaceholder` ActiveX control provides following connection point events:

*OnModifiedFlagChanged*
OnModifiedFlagChanged

**Event:** OnModifiedFlagChanged(i_bIsModified as boolean)

**Dispatch Id:** 1

**Description:**
This event gets triggered only for placeholder controls with a `PlaceholderWindowID` of `Authentic DesktopXProjectWindow (=3)`. The event is fired whenever the project content changes between modified and unmodified state. The parameter `i_bIsModified` is `true` if the project contents differs from the original content, and `false`, otherwise.

OnSetLabel

**Event:** OnSetLabel(i_strNewLabel as string)

**Dispatch Id:** 1000

**Description:**
Raised when the title of the placeholder window is changed.
4.7.6 Enumerations

The following enumerations are defined:

ICActiveXIntegrationLevel
AuthenticDesktopControlPlaceholderWindow

ICActiveXIntegrationLevel
Possible values for the IntegrationLevel property of the AuthenticDesktopControl.

ICActiveXIntegrationOnApplicationLevel = 0
ICActiveXIntegrationOnDocumentLevel = 1

AuthenticDesktopControlPlaceholderWindow
This enumeration contains the list of the supported additional Authentic Desktop windows.

AuthenticDesktopControlNoToolWnd = -1
AuthenticDesktopControlEntryHelperTopToolWnd = 0
AuthenticDesktopControlEntryHelperMiddleToolWnd = 1
AuthenticDesktopControlEntryHelperBottomToolWnd = 2
AuthenticDesktopControlValidatorOutputToolWnd = 3
AuthenticDesktopControlProjectWindowToolWnd = 4
AuthenticDesktopControlInfoToolWnd = 18
Chapter 4

Appendices
Appendices

These appendices contain technical information about Authentic Desktop and important licensing information. Each appendix contains sub-sections as given below:

**Technical Data**
- OS and memory requirements
- Altova XML Parser
- Altova XSLT and XQuery Engines
- Unicode support
- Internet usage

**License Information**
- Electronic software distribution
- Intellectual property rights and copyright
- End User License Agreement
1 Technical Data

This section contains information on some technical aspects of your software. This information is organized into the following sections:

- OS and Memory Requirements
- Altova Engines
- Unicode Support
- Internet Usage
1.1 OS and Memory Requirements

Operating System
Altova software applications are available for the following platforms:

- Windows 7 SP1 with Platform Update, Windows 8, Windows 10
- Windows Server 2008 R2 SP1 with Platform Update or newer

Memory
Since the software is written in C++ it does not require the overhead of a Java Runtime Environment and typically requires less memory than comparable Java-based applications. However, each document is loaded fully into memory so as to parse it completely and to improve viewing and editing speed. As a result, the memory requirement increases with the size of the document.

Memory requirements are also influenced by the unlimited Undo history. When repeatedly cutting and pasting large selections in large documents, available memory can rapidly be depleted.
1.2 Altova Engines

XML Validator
When opening an XML document, the application uses its built-in XML validator to check for well-formedness, to validate the document against a schema (if specified), and to build trees and infosets. The XML validator is also used to provide intelligent editing help while you edit documents and to dynamically display any validation error that may occur.

The built-in XML validator implements the Final Recommendation of the W3C’s XML Schema 1.0 and 1.1 specifications. New developments recommended by the W3C’s XML Schema Working Group are continuously being incorporated in the XML validator, so that Altova products give you a state-of-the-art development environment.

XSLT and XQuery Engines
Altova products use the Altova XSLT 1.0, 2.0, and 3.0 Engines and the Altova XQuery 1.0 and 3.1 Engines. If one of these engines is included in the product, then documentation about implementation-specific behavior for each engine is given in the appendices of the documentation.

Note: Altova MapForce generates code using the XSLT 1.0, 2.0 and XQuery 1.0 engines.
1.3 **Unicode Support**

Altova’s XML products provide full Unicode support. To edit an XML document, you will also need a font that supports the Unicode characters being used by that document.

Please note that most fonts only contain a very specific subset of the entire Unicode range and are therefore typically targeted at the corresponding writing system. If some text appears garbled, the reason could be that the font you have selected does not contain the required glyphs. So it is useful to have a font that covers the entire Unicode range, especially when editing XML documents in different languages or writing systems. A typical Unicode font found on Windows PCs is Arial Unicode MS.

In the `/Examples` folder of your application folder you will find an XHTML file called `UnicodeUTF-8.html` that contains the following sentence in a number of different languages and writing systems:

- *When the world wants to talk, it speaks Unicode*
- *Wenn die Welt miteinander spricht, spricht sie Unicode*
- *世界的に話すなら、Unicodeです。*

Opening this XHTML file will give you a quick impression of Unicode’s possibilities and also indicate what writing systems are supported by the fonts available on your PC.
1.4 Internet Usage

Altova applications will initiate Internet connections on your behalf in the following situations:

- If you click the "Request evaluation key-code" in the Registration dialog (Help | Software Activation), the three fields in the registration dialog box are transferred to our web server by means of a regular http (port 80) connection and the free evaluation key-code is sent back to the customer via regular SMTP e-mail.

- In some Altova products, you can open a file over the Internet (File | Open | Switch to URL). In this case, the document is retrieved using one of the following protocol methods and connections: HTTP (normally port 80), FTP (normally port 20/21), HTTPS (normally port 443). You could also run an HTTP server on port 8080. (In the URL dialog, specify the port after the server name and a colon.)

- If you open an XML document that refers to an XML Schema or DTD and the document is specified through a URL, the referenced schema document is also retrieved through a HTTP connection (port 80) or another protocol specified in the URL (see Point 2 above). A schema document will also be retrieved when an XML file is validated. Note that validation might happen automatically upon opening a document if you have instructed the application to do this (in the File tab of the Options dialog (Tools | Options)).

- In Altova applications using WSDL and SOAP, web service connections are defined by the WSDL documents.

- If you are using the Send by Mail command (File | Send by Mail) in XMLSpy, the current selection or file is sent by means of any MAPI-compliant mail program installed on the user's PC.

- As part of Software Activation and LiveUpdate as further described in the Altova Software License Agreement.
2 License Information

This section contains information about:

- the distribution of this software product
- the license agreement governing the use of this product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

To view the terms of any Altova license, go to the [Altova Legal Information page](https://www.altova.com/legal) at the [Altova website](https://www.altova.com).
2.1 Electronic Software Distribution

This product is available through electronic software distribution, a distribution method that provides the following unique benefits:

- You can evaluate the software free-of-charge for 30 days before making a purchasing decision. *(Note: Altova MobileTogether Designer is licensed free of charge.)*
- Once you decide to buy the software, you can place your order online at the Altova website and get a fully licensed product within minutes.
- When you place an online order, you always get the latest version of our software.
- The product package includes an onscreen help system that can be accessed from within the application interface. The latest version of the user manual is available at www.altova.com in (i) HTML format for online browsing, and (ii) PDF format for download (and to print if you prefer to have the documentation on paper).

30-day evaluation period

After downloading this product, you can evaluate it for a period of up to 30 days free of charge. About 20 days into the evaluation period, the software will start to remind you that it has not yet been licensed. The reminder message will be displayed once each time you start the application. If you would like to continue using the program after the 30-day evaluation period, you must purchase a product license, which is delivered in the form of a license file containing a key code. Unlock the product by entering this key code in the Software Activation dialog of your product.

You can purchase product licenses at the Altova online shop.

Helping Others within Your Organization to Evaluate the Software

If you wish to distribute the evaluation version within your company network, or if you plan to use it on a PC that is not connected to the Internet, you may distribute only the installer file, provided that this file is not modified in any way. Any person who accesses the software installer that you have provided must request their own 30-day evaluation license key code and after expiration of their evaluation period, must also purchase a license in order to be able to continue using the product.
2.2 Altova End-User License Agreement for Authentic

- The Altova End-User License Agreement for Authentic is available here: https://www.altova.com/legal/authentic-eula
- Altova's Privacy Policy is available here: https://www.altova.com/privacy
Index

...
Index

**Authentic Desktop integration,**
example of, 720, 721

**Authentic DesktopCommand,**
in AuthenticDesktopControl, 748

**Authentic DesktopCommands,**
in AuthenticDesktopControl, 751

**Authentic Integration Package,** 137, 140

**Authentic menu,** 233
dynamic table editing, 40
markup display, 40

**Authentic perspective in Eclipse,** 147

**Authentic Plugin for Eclipse,**
installing, 140

**Authentic Plugin for VS .NET,**
installing, 137

**Authentic Scripting,**
security settings, 247
trusted locations, 247

**Authentic View,** 63
adding nodes, 29
applying elements, 29
CDATA sections in, 32
clearing elements, 29
context menu, 26
context menus, 51
data entry devices in, 32
displaying markup tags, 26
document display, 43
ingoing data in an XML DB, 237
editing DB data in, 235
entering attribute values, 34
entering data in, 32
entities in, 32
entry helpers, 26
entry helpers in, 46
formatting text in, 40
generating output documents from PXF file, 246
inserting entities in, 35
inserting nodes, 29
main window in, 43
markup display in, 40, 43
opening an XML document in, 24
opening new XML file in, 234
overview of GUI, 38
paste as XML/Text, 51
printing an XML document from, 36
removing nodes, 29
special characters in, 32

**SPS Tables,** 61
switching to, 249
tables (SPS and XML), 60
tables in, 29
toolbar icons, 40
usage of important features, 54
usage of XML tables, 63
XML table icons, 68
XML tables, 63

**Authentic View Events,** 345

**Authentic View template,** 24

**AuthenticDataTransfer,**
dropEffect, 444
dataGet, 444
ownDrag, 445
type, 445

**AuthenticDesktopControl,** 752
documentation of, 703
element of integration at application level, 720, 721
examples of integration at document level, 715
integration using C#, 716
integration using HTML, 721
object reference, 747

**AuthenticDesktopControlDocument,** 760

**AuthenticDesktopControlPlaceHolder,** 766

**AuthenticRange,**
AppendRow, 450
Application, 451
Can PerformAction, 451
Can PerformActionWith, 451
Close, 452
Collapse To Begin, 452
Collapse To End, 452
Copy, 453
Cut, 453
Delete, 453
DeleteRow, 454
DuplicateRow, 454
ExpandTo, 455
First Text Position, 455
First XML Data, 456
First XML Data Offset, 457
Get Element Attribute Names, 458
Get Element Attribute Value, 459
Get Element Hierarchy, 459
Get Entity Names, 459
Goto, 460
Goto Next, 460
AuthenticRange, 462
GotoNextCursorPosition, 461
GotoPrevious, 462
GotoPreviousCursorPosition, 462
HasElementAttribute, 463
InsertEntity, 463
InsertRow, 464
IsCopyEnabled, 464
IsCutEnabled, 464
IsDeleteEnabled, 465
IsEmpty, 465
IsEqual, 465
IsFirstRow, 465
IsInDynamicTable, 466
IsLastRow, 466
IsPasteEnabled, 466
IsTextStateApplied, 467
LastCursorPosition, 467
LastXMLData, 468
LastXMLDataOffset, 469
MoveBegin, 470
MoveEnd, 470
MoveRowDown, 471
MoveRowUp, 471
Parent, 471
Paste, 471
PerformAction, 472
Select, 473
SelectNext, 473
SelectPrevious, 474
SetElementAttributeValue, 475
SetFromRange, 476
Text, 476
AuthenticView, 493
Application, 486
AsXMLString, 486
DocumentBegin, 488
DocumentEnd, 488
Event, 489
Goto, 490
IsRedoEnabled, 490
IsUndoEnabled, 491
MarkupVisibility, 491
OnBeforeCopy, 478
OnBeforeCut, 479
OnBeforeDelete, 479
OnBeforeDrop, 480
OnBeforePaste, 480
OnDragOver, 481
OnKeyboardEvent, 482
OnMouseEvent, 483
OnSelectionChanged, 483
Parent, 491
Print, 491
Redo, 492
Selection, 492
Undo, 493
WholeDocument, 494
XMLDataRoot, 494
Auto-hiding windows, 8
Auto-Macro setting, 350
Automatic validation, 288

B
Background Information, 773
Big-endian, 290
Bookmark, 311
Browser, 291
View, 250
Browser menu, 251
Browser View, 251
back, 252
font size, 256
forward, 253
refresh content, 255
separate window, 257
stop loading page, 254

C
C#, 716
integration of Authentic Desktop, 716
Carriage return key, see Enter key, 88
Cascade, Window, 300
Catalog, Oasis XML, 223
CDATA sections, inserting in Authentic View, 55
Changing view,
Changing view,
to Authentic View, 40

Chapters, 311
Character-Set,
encoding, 290
Check,
spelling checker, 259
Class ID,
in Authentic Desktop integration, 720

CodeGeneratorDlg,
Application, 495
CPPSettings_DOMType, 496
CPPSettings_LibraryType, 497
CPPSettings_UseMFC, 497
CSharpSettings_ProjectType, 497
OutputPath, 498
OutputPathDialogAction, 498
OutputResultDialogAction, 498
Parent, 499
ProgrammingLanguage, 499
PropertySheetDialogAction, 499
TemplateName, 500

COM-API,
documentation, 390
Command, 277
add to toolbar/menu, 270
context menu, 277
delete from menu, 277
reset menu, 277
Command line, 318
Commands,
listing in key map, 312
Configurations,
of a global resource, 93
Configurations in global resources, 109
Configure,
XMLSPY UI, 372
Context menu,
commands, 277
for customization, 281
Context menus,
in Authentic View, 51
Copy command, 176
Copyright information, 778
CR&LF, 286
Custom dictionary, 259
CustomCatalog, 223
Customization, 19

Customize, 277
customize context menu, 277
Customize context menu, 281
macros, 278
menu, 277
toolbar/menu commands, 270
Cut command, 176

D

DatabaseConnection,
ADOConnection, 501
AsAttributes, 502
CreateMissingTables, 502
CreateNew, 503
DatabaseKind, 503
ExcludeKeys, 503
File, 504
IncludeEmptyElements, 504
NumberDateTimeFormat, 505
ODBCConnection, 505
SQLSelect, 506
TextFieldLen, 506

Databases,
editing in Authentic View, 235
see also DB, 71

Date Picker,
using in Authentic View, 81

Dates,
changing manually, 82

DB, 71, 73
creating queries, 73
editing in Authentic View, 71, 78
filtering display in Authentic View, 73
navigating tables in Authentic View, 72
parameters in DB queries, 73
queries in Authentic View, 71

Debugging macros, 355
Default,
encoding, 290
menu, 277

Default editor, 288

Default view,
setting in Main Window, 288

Delete, 270
Application.URLDelete, 440
Index

Delete, 270
  command from context menu, 277
  command from toolbar, 270
  icon from toolbar, 270
  row (in Authentic View), 244
  shortcut, 274
  toolbar, 272
Delete command, 176
Dialogs,
  Application, 508
  CodeGeneratorDlg, 508
  DTDSchemaGeneratorDlg, 510
  FileSelectionDlg, 509
  GenerateSampleXMLDlg, 509
  Parent, 509
  SchemaDocumentationDlg, 509
Dictionary, 259
  adding custom, 259
  modifying existing, 259
  spelling checker, 259
directories,
  creating with Application.URLMakeDirectory, 440
Distribution,
  of Altova's software products, 778, 779
Dockable window, 303, 304
Docking windows, 8
Document, 524
  Application, 517
  AssignDTD, 517
  AssignSchema, 517
  AssignXSL, 517
  AssignXSLFO, 518
  AuthenticView, 518
  Close, 519
  ConvertDTDDOrSchema, 519
  CreateChild, 521
  CreateSchemaDiagram, 522
  CurrentViewMode, 522
  DataRoot, 522
  DocEditView, 523
  Encoding, 523
  EndChanges, 524
  ExecuteXQuery, 524
  ExportToDatabase, 524
  ExportToText, 525
  FullName, 527
  GenerateDTDDOrSchema, 527, 528
  GenerateProgramCode, 528
  GenerateSampleXML, 528
  GenerateSchemaDocumentation, 529
  GetExportElementList, 531
  GetPathName, 531
  GridView, 531
  IsModified, 532
  IsValid, 532
  IsWellFormed, 533
  Name, 533
  OnBeforeCloseDocument, 514
  OnBeforeSaveDocument, 514
  OnBeforeValidate, 515
  OnCloseDocument, 516
  OnViewActivation, 516
  Path, 534
 RootElement, 534
  Save, 535
  SaveAs, 535
  Saved, 535
  SaveInString, 535
  SaveToURL, 536
  SetActiveDocument, 536
  SetEncoding, 536
  SetExternallsValid, 538
  SetPathName, 538
  Spelling checker, 259
  StartChanges, 538
  SwitchViewMode, 539
  Title, 539
  TransformXSL, 540
  TransformXSLFO, 540
  UpdateViews, 541
  UpdateXMLData, 541
  XQuery, 524
Document Events, 345
Document-level,
  examples of integration of <%SPY-GEN%>, 715
Documents,
  Count, 542
  Item, 543
  NewAuthenticFile, 543
  NewFile, 543
  NewFileFromText, 544
  OpenAuthenticFile, 544
  OpenFile, 544
  OpenURL, 545
  OpenURLDialog, 546
Documents in Main Window, 10

© 2019 Altova GmbH
Empty elements, 288

Encoding,
default, 290
of files, 161

End User License Agreement, 778

Enter key,
effects of using, 88

Entities,
defining in Authentic View, 55, 83
inserting in Authentic View, 35, 55

Entities entry helper,
in Authentic View, 46

Entry helpers, 15
toggling display on and off, 307

Entry-Helper, 305, 308

Enumerations,
in AuthenticDesktopControl, 769
SPYAttributeTypeDefinition, 660
SPYAuthenticActions, 661
SPYAuthenticDocumentPosition, 662
SpyAuthenticElementActions, 663
SPYAuthenticElementKind, 664
SPYAuthenticMarkupVisibility, 665
SPYDatabaseKind, 667
SPYDialogAction, 668
SPYDOMType, 669
SPYDTDSchemaFormat, 670
SPYEncodingByteOrder, 671
SPYExportNamespace, 672
SPYFrequentElements, 674
SPYKeyEvent, 677
SPYLibType, 679
SPYLoading, 680
SPYMouseEvent, 681
SPYNumberDateTimeFormat, 682
SPYProgrammingLanguage, 683
SPYProjectItemTypes, 684
SPYProjectType, 685
SPYSampleXMLGenerationOptimization, 687
SPYSampleXMLGenerationSchemaOrDTDAssignment, 689
SPYSchemaDefKind, 690
SPYSchemaDocumentationFormat, 691
SPYTextDelimiters, 694
SPYTextEnclosing, 695
SPYTypeDetection, 696
SPYURLTapes, 697
SPYViewModes, 700

Eclipse platform,
and Authentic Desktop, 139
and Authentic Integration Package, 140
Authentic perspective in, 147

Edit,
macro button, 281

Edit menu, 174

Edited with XMLSpy, 286

ElementList,
Count, 551
Item, 551
RemoveElement, 551

ElementListItem,
ElementKind, 553
FieldCount, 553
Name, 553
RecordCount, 553

Elements entry helper,
in Authentic View, 46

E-mail,
sending files with, 169

Duplicate,
row (in Authentic View), 244

Dynamic (SPS) tables in Authentic View,
usage of, 61

Dynamic tables,
editing, 40

DTDs, 286, 288

DTDSchemaGeneratorDlg,
Application, 547
AttributeTypeDefinition, 547
DTDSchemaFormat, 548
FrequentElements, 548
GlobalAttributes, 548
MaxEnumLength, 548
MergeAllEqualNamed, 549
OnlyStringEnums, 549
OutputPath, 549
OutputPathDialogAction, 549
Parent, 550
ResolveEntities, 550
TypeDetection, 550
ValueList, 550

Duplicate,
row (in Authentic View), 244

Dynamic (SPS) tables in Authentic View,
usage of, 61

Dynamic tables,
editing, 40

Eclipse platform,
and Authentic Desktop, 139
and Authentic Integration Package, 140
Authentic perspective in, 147

Edit,
macro button, 281

Edit menu, 174

Edited with XMLSpy, 286

ElementList,
Count, 551
Item, 551
RemoveElement, 551

ElementListItem,
ElementKind, 553
FieldCount, 553
Name, 553
RecordCount, 553

Elements entry helper,
in Authentic View, 46

E-mail,
sending files with, 169

Empty elements, 288

Encoding,
default, 290
of files, 161

End User License Agreement, 778

Enter key,
effects of using, 88

Entities,
defining in Authentic View, 55, 83
inserting in Authentic View, 35, 55

Entities entry helper,
in Authentic View, 46

Entry helpers, 15
toggling display on and off, 307

Entry-Helper, 305, 308

Enumerations,
in AuthenticDesktopControl, 769
SPYAttributeTypeDefinition, 660
SPYAuthenticActions, 661
SPYAuthenticDocumentPosition, 662
SpyAuthenticElementActions, 663
SPYAuthenticElementKind, 664
SPYAuthenticMarkupVisibility, 665
SPYDatabaseKind, 667
SPYDialogAction, 668
SPYDOMType, 669
SPYDTDSchemaFormat, 670
SPYEncodingByteOrder, 671
SPYExportNamespace, 672
SPYFrequentElements, 674
SPYKeyEvent, 677
SPYLibType, 679
SPYLoading, 680
SPYMouseEvent, 681
SPYNumberDateTimeFormat, 682
SPYProgrammingLanguage, 683
SPYProjectItemTypes, 684
SPYProjectType, 685
SPYSampleXMLGenerationOptimization, 687
SPYSampleXMLGenerationSchemaOrDTDAssignment, 689
SPYSchemaDefKind, 690
SPYSchemaDocumentationFormat, 691
SPYTextDelimiters, 694
SPYTextEnclosing, 695
SPYTypeDetection, 696
SPYURLTapes, 697
SPYViewModes, 700

© 2019 Altova GmbH
Enumerations,
- SPYVirtualKeyMask, 701
- SPYXMLDataKind, 702

Evaluation key,
- for your Altova software, 313

Evaluation period,
- of Altova's software products, 778, 779

Event, 423, 424, 425, 478, 479, 480, 481, 482, 483, 514, 515, 516, 576, 577, 578

Event handlers,
- in Scripting Project, 345
- overview, 331

Events, 345, 398
- and event handlers, 333

Explorer, 288

ExportSettings,
- CreateKeys, 555
- ElementList, 555
- EntitiesToText, 555
- ExportAllElements, 556
- FromAttributes, 556
- FromSingleSubElements, 556
- FromTextValues, 556
- IndependentPrimaryKey, 557
- Namespace, 557
- SubLevelLimit, 557

External applications,
- opening files in, 273

External parsed entities, 288

External XSL processor, 292

customizing, 223

File menu, 151

File types, 288

Files,
- adding to source control, 194
- most recently used, 173

FileSelectionDlg,
- Application, 558
- DialogAction, 558
- FullName, 559
- Parent, 559

Find,
- and replace text in document, 179
- text in document, 178

Floating windows, 8

Font size,
- in Browser View, 256

Form Object Palette, 327

Form Object properties, 340

Forms,
- and built-in commands, 358
- and event handling, 343
- and Form Objects, 340
- creating new, 338
- in Scripting Projects, 337
- invocation of, 333
- naming, 338
- overview, 331
- properties of, 338
- setting tab sequence of objects, 340

F

Favorites, 311

File, 286
- closing, 162
- creating new, 152
- default encoding, 290
- encoding, 161
- opening, 154
- opening options, 286
- printing options, 171
- saving, 163
- sending by e-mail, 169
- tab, 286

File extensions,
Global resources, 92
active configuration for, 269
changing configurations, 109
defining, 93, 268
defining database-type, 103
defining file-type, 96
defining folder-type, 101
toolbar activation, 272
using, 105, 109
using file-type and folder-type, 106
Global Resources XML File, 93
Global scripting project,
of Authentic Desktop, 325
Grammar, 288
Graphics formats,
in Authentic View, 87
Grid View Events, 345
GridView,
CurrentFocus, 578
Deselect, 579
isVisible, 579
OnBeforeDrag, 576
OnBeforeDrop, 576
OnBeforeStartEditing, 577
OnEditingFinished, 577
OnFocusChanged, 578
Select, 579
SetFocus, 579
GUI description, 8

Help,
contents, 311
key map, 312
Help menu, 310
Help system, 311
Hide, 303, 304, 305, 308
Hide markup, 40, 43
Hide markup (in Authentic View), 242
Hotkey, 274
HTML,
integration of Authentic Desktop, 721
HTML example,
of AuthenticDesktopControl integration, 720, 721
HTML output,
generating in Authentic View from PXF file, 246

Icon,
add to toolbar/menu, 270
show large, 281
Image formats,
in Authentic View, 87
Info Window, 14, 304, 308
Insert,
row (in Authentic View), 244
Integrating,
Authentic Desktop in applications, 703
Internet, 317
Internet usage,
in Altova products, 777

Java, 724
Java settings, 294
Java virtual machine,
path setting, 294
JRE,
for Authentic Plugin for Eclipse, 140

Key map, 312
Keyboard shortcut, 274
Key-codes,
for your Altova software, 313

Language,
scripting language - changing, 333
Large markup (in Authentic View), 242
Legal information, 778
License,
License,
  information about, 778
Licenses,
  for your Altova software, 313
Line-breaks, 286
Links,
  following in Authentic View, 55
Little-endian, 290
loading, 545

M
Macro,
  add to menu/toolbar, 278
  edit button, 281
Macros,
  creating with Scripting Editor, 350
  debugging, 355
  editing with Scripting Editor, 350
  execution of, 333
  functions for, in Global Declarations, 335
  how to use in Scripting Project, 349
  overview, 331
  running, 352
  running application macros, 266
  setting as Auto-Macro in Scripting Editor, 350
Main Window, 10
MainCatalog, 223
Markup,
  in Authentic View, 40, 43
Markup (in Authentic View),
  hide, 242
  show small/large/mixed, 242
Maximum cell width, 291
Memory requirements, 774
Menu, 277
  add macro to, 278
  add/delete command, 270
  Authentic, 233
  customize, 277
  Default/XMLSpy, 277
  delete commands from, 277
  Edit, 174
  Help, 310
  Project, 180
  Tools, 258

View, 248
Window, 299
XML, 220
XSL/XQuery, 225
Menu Bar, 17
Menu Browser, 251
Messages Window, 16
Microsoft® SharePoint® Server, 211
MIME, 288
Mixed markup (in Authentic View), 242
Mostly recently used files,
  list of, 173
Move up/down,
  row (Authentic View, 245
MSXML, 292
Multi-user, 286

N
New file,
  creating, 152
Non-XML files, 288

O
OASIS,
  XML catalog, 223
Open,
  file, 154
Opening options,
  file, 286
Optimal Widths, 291
Ordering Altova software, 313
OS,
  for Altova products, 774
Output formatting, 286
Output windows,
  toggling display on and off, 306
Overview,
  of XMLSpy API, 392
Parameters, in DB queries, 73
    passing to stylesheet via interface, 229
Parent, 534
Parser, built into Altova products, 775
    XSLT, 292
Paste, as Text, 55
    as XML, 55
Paste As, Text, 51
    XML, 51
Paste command, 176
PDF output, generating in Authentic View from PXF file, 246
Platforms, for Altova products, 774
Plug-in, ATL sample files, 375
    registration, 370
    User interface configuration, 372
    XMLSPY, 369
Presentation, 291
Print setup, 172
Printing, from Authentic View, 36
Printing options, 171
Program settings, 286
Programmers' Reference, 320
Programming points, in Scripting Project, 356
Project, properties, 216
Project menu, 180
Project Window, 12, 303, 308
    toggling display on and off, 307
Projects, 206
    adding active files to, 205, 206
    adding external folders to, 208
    adding external Web folders to, 211
    adding files to, 202
    adding folders to, 207
    adding global resources to, 203
    adding related files to, 206
    adding to source control, 194
    adding URL to, 204
    closing, 186
    creating new, 183
    most recently used, 219
    opening, 184
    overview, 180
    reloading, 185
    saving, 187
Properties and Events pane, 327
PUBLIC, identifier - catalog, 223
PXF file, generating output documents from Authentic View, 246

Queries, for DB display in Authentic View, 73

Redo command, 175
Register, plug-in, 370
Registering your Altova software, 313
Registry, settings, 286
Regular expressions, in search string, 178
Reload, 286
Reloading, changed files, 160
Replace, text, 178
    text in document, 179
Reset, menu commands, 277
    shortcut, 274
    toolbar & menu commands, 272
Return key, see Enter key, 88
RichEdit, 243
Row,
  append (in Authentic View), 244
delete (in Authentic View), 244
duplicate (in Authentic View), 244
insert (in Authentic View), 244
move up/down, 245
RTF output,
  generating in Authentic View from PXF file, 246

ShowSourceCode, 592
ShowType, 592
ShowUsedBy, 592
Script language, 295
Scripting, 295
Scripting Editor,
  GUI description, 327
  Main Window, 327
  starting, 265
Scripting Environment, 322
  usage overview, 324
Scripting language, 333
Scripting Project,
  and Events, 345
  application event handlers, 345
  Event Handlers, 331
  Forms, 331
  Forms in, 337
  Global Declarations, 331
  Global Declarations in, 335
  Macros, 331
  Macros in, 349
  programming points, 356
  steps for creating, 333
Scripting Project Tree pane, 327
Scripting Projects,
  for Authentic Desktop, 325
  for Authentic Desktop Project, 325
Search,
  see Find, 179
Select All command, 177
Settings, 19, 286
  scripting, 295
SharePoint® Server, 211
Shortcut, 274
  assigning/deleting, 274
  show in tooltip, 281
Show, 303, 304, 305, 308
Show large markup, 40, 43
Show mixed markup, 40, 43
Show small markup, 43
Show small markup, 40
Side-by-side, 291
Small markup (in Authentic View), 242
Source control, 295
  add to source control, 194
  changing provider, 200
  checking out, 191
Source control, 295
  enabling, disabling, 189
  get latest version, 190
  getting files, 190
  installing a source-control plug-in, 110
  open project, 188
  properties, 199
  refresh status, 200
  removing from, 195
  sharing from, 195
  show differences, 198
  show history, 197
  supported providers, 188
  undo check out, 193

Source control manager, 200

Spelling checker, 259
  custom dictionary, 259

Spelling options, 262

Splash screen, 291

SPP file locations, 180

SPS,
  assigning to new XML file, 152

SPS tables,
  editing dynamic tables, 40

SPS tables in Authentic View,
  usage of, 61

SpyProject,
  CloseProject, 594
  ProjectFile, 594
  RootItems, 594
  SaveProject, 595
  SaveProjectAs, 595

SpyProjectItem,
  ChildItems, 596
  FileExtensions, 596
  ItemType, 596
  Name, 597
  Open, 597
  ParentItem, 597
  Path, 597
  ValidateWith, 597
  XMLForXSLTransformation, 598
  XSLForXMLETransformation, 598
  XSLTransformationFileExtension, 598
  XSLTransformationFolder, 598

SpyProjectItems,
  AddFile, 599
  AddFolder, 599
  AddURL, 599
  Count, 600
  Item, 600
  RemoveItem, 600

Start group,
  add (context menu), 281

Static (SPS) tables in Authentic View,
  usage of, 61

Status Bar, 17

Support Center, 317

Support options, 21

Syntax-coloring, 288, 291

T

Tab characters, 286

Table,
  build automatically, 288

Table of contents, 311

Tables,
  editing dynamic (SPS) tables, 40
  in Authentic View, 29

Tables in Authentic View,
  icons for editing XML tables, 68
  usage of, 60
  using SPS (static and dynamic) tables, 61
  using XML tables, 63

Technical Information, 773

Technical Support, 317

Template files,
  for new documents, 152

Template XML File,
  in Authentic View, 24

Templates,
  of XML documents in Authentic View, 234

terminate, 437

Text,
  editing in Authentic View, 55
  find and replace, 179
  finding in document, 178
  formatting in Authentic View, 55

Text View Events, 345

TextImportExportSettings,
  DestinationFolder, 601
  EnclosingCharacter, 601
  Encoding, 602

© 2019 Altova GmbH
TextImportExportSettings,
   EncodingByteOrder, 602
   FieldDelimiter, 602
   FileExtension, 602
   HeaderRow, 602
   ImportFile, 603
Tile,
   horizontally, 301
   vertically, 302
Toggle, 303, 304, 305, 308
Toolbar, 17, 272
   activate/deactivate, 272
   add command to, 270
   add macro to, 278
   create new, 272
   reset toolbar & menu commands, 272
   show large icons, 281
Tools,
   see also External applications, 273
Tools menu, 258
Tooltip, 281
   show, 281
   show shortcuts in, 281
Topic,
   view on TOC, 311
Transformation,
   see XSLT transformation, 227
Trusted locations for Authentic scripts, 247
Turn off automatic validation, 288

V
Validation, 19, 223
Validation messages, 16
Validator,
   in Altova products, 775
View,
   Browser view, 250
View menu, 248
Visual Studio,
   adding the Authentic Desktop ActiveX Controls to the
toolbox, 706
Visual Studio .Net,
   and Authentic Desktop, 136
   and Authentic Desktop differences, 138
VS.NET,
   and Authentic Integration Package, 137

W
Watch for changes, 286
Web Server, 317
Well-formedness check, 221
Window, 308
   Cascade, 300
   Entry-Helper, 305, 308
   Info, 304, 308
   Open, 309
   Project, 303, 308
   Tile horizontally, 301
   Tile vertically, 302
Window menu, 299
Windows,
   auto-hiding, 8
   floating, docking, tabbing, 8
   managing display of, 8
   support for Altova products, 774
Word 2007+ output,
   generating in Authentic View from PXF file, 246

U
UCS-2, 290
Undo command, 175
Unicode support,
   in Altova products, 776
URL, 440, 536, 545, 546
   sending by e-mail, 169
User interface,
   configure using plug-in, 372
User interface description, 8
User manual, 3
User Manual. Authentic Desktop, 6
User Reference, 150
UTF-16, 290
XML

Oasis catalog, 223
spelling checker, 259

XML DB,
loading new data row into Authentic View, 237
loading new XML data row, 72

XML document,
opening in Authentic View, 24

XML menu, 220

XML Parser,
about, 775

XML Signature, 238

XML signatures, 85

XML tables in Authentic View,
icons for editing, 68
usage of, 63

XML-Conformance, 288

XMLData,
AppendChild, 647
EraseAllChildren, 648
EraseCurrentChild, 649
GetChild, 649
GetChildKind, 650
GetCurrentChild, 651
GetFirstChild, 651
GetNextChild, 652
HasChildren, 653
HasChildrenKind, 654
InsertChild, 654
IsSameNode, 655
Kind, 655
MayHaveChildren, 655
Name, 656
Parent, 656
TextValue, 657

XMLSpy, 150

features, 21
help, 21
plug-in registration, 370

XMLSpy API,
documentation, 390
overview, 392

XMLSpy command table, 734

XMLSpy plug-in, 369
XMLSpyLib, 390, 393

Application, 422
AuthenticDataTransfer, 444
AuthenticRange, 449
AuthenticView, 478
CodeGenGeneratorDlg, 495
DatabaseConnection, 501
Dialogs, 508
Document, 512
Documents, 542
DTDSchemaGeneratorDlg, 547
ElementList, 551
ElementListIterator, 553
ExportSettings, 555
FileSelectionDlg, 558
GenerateSampleXMLDlg, 571
GridView, 576
ProjectItem, 596
SchemaDocumentationDlg, 580
SpyProject, 594
SpyProjectItems, 599
TextImportExportSettings, 601
XMLData, 646

XPath to selected node, 38

XQuery,
passing variables to the XQuery document, 229

XSL/XQuery menu, 225

XSLT,
processor, 292

XSLT parameters,
passing to stylesheet via interface, 229

XSLT transformation, 226, 227
to FO, 227
to PDF, 227