

# Altova Authentic 2024 Desktop



**User & Reference Manual**

# **Altova Authentic 2024 Desktop User & Reference Manual**

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2024

© 2018-2024 Altova GmbH

---

# Table of Contents

<b>1</b>	<b>About Authentic Desktop and This Documentation</b>	<b>10</b>
1.1	Windows File Paths.....	11
1.2	About This Documentation.....	12
<b>2</b>	<b>GUI and Environment</b>	<b>13</b>
2.1	The Graphical User Interface (GUI).....	14
2.1.1	Main Window.....	15
2.1.2	Project Window.....	17
2.1.3	Info Window.....	19
2.1.4	Entry Helpers.....	19
2.1.5	Output Window: Messages.....	19
2.1.6	Menu Bar, Toolbars, Status Bar.....	20
2.2	The Application Environment.....	21
2.2.1	Settings and Customization.....	21
2.2.2	Tutorials, Projects, Examples.....	21
2.2.3	Authentic Desktop Features and Help, and Altova Products.....	22
<b>3</b>	<b>Authentic View Tutorial</b>	<b>23</b>
3.1	Opening an XML Document in Authentic View.....	25
3.2	The Authentic View Interface.....	27
3.3	Node Operations.....	30
3.4	Entering Data in Authentic View.....	33
3.5	Entering Attribute Values.....	36
3.6	Adding Entities.....	37
3.7	Printing the Document.....	38
<b>4</b>	<b>Authentic View Interface</b>	<b>39</b>

---

4.1	Overview of the GUI.....	40
4.2	Authentic View Toolbar Icons.....	42
4.3	Authentic View Main Window.....	45
4.4	Authentic View Entry Helpers.....	48
4.5	Authentic View Context Menus.....	53
<b>5</b>	<b>Editing in Authentic View</b>	<b>55</b>
5.1	Automatic Backup of Files.....	56
5.2	Basic Editing.....	58
5.3	Tables in Authentic View.....	63
5.3.1	SPS Tables.....	63
5.3.2	CALS/HTML Tables.....	64
5.3.3	CALS/HTML Table Editing Icons.....	68
5.4	Editing a DB.....	71
5.4.1	Navigating a DB Table.....	71
5.4.2	DB Queries.....	72
5.4.3	Modifying a DB Table.....	76
5.5	Working with Dates.....	78
5.5.1	Date Picker.....	78
5.5.2	Text Entry.....	79
5.6	Defining Entities.....	81
5.7	XML Signatures.....	83
5.8	Images in Authentic View.....	84
5.9	Keystrokes in Authentic View.....	85
<b>6</b>	<b>Authentic Scripting</b>	<b>86</b>
<b>7</b>	<b>Browser View</b>	<b>88</b>
<b>8</b>	<b>Altova Global Resources</b>	<b>90</b>
8.1	Defining Global Resources.....	91
8.1.1	Files .....	93

---

8.1.2	Folders.....	98
8.1.3	Databases.....	100
8.2	Using Global Resources.....	102
8.2.1	Assigning Files and Folders.....	102
8.2.2	Changing the Active Configuration.....	105
<b>9</b>	<b>Source Control</b> .....	<b>106</b>
9.1	Setting Up Source Control.....	108
9.2	Supported Source Control Systems.....	109
9.3	Local Workspace Folder.....	111
9.4	Application Project.....	112
9.5	Add to Source Control.....	114
9.6	Working with Source Control.....	116
9.6.1	Add to, Remove from Source Control.....	116
9.6.2	Check Out, Check In.....	117
9.6.3	Getting Files as Read-Only.....	119
9.6.4	Copying and Sharing from Source Control.....	121
9.6.5	Changing Source Control.....	124
9.7	Source Control with Git.....	126
9.7.1	Enabling Git Source Control with GIT SCC Plug-in.....	127
9.7.2	Adding a Project to Git Source Control.....	127
9.7.3	Cloning a Project from Git Source Control.....	129
<b>10</b>	<b>Schema Manager</b> .....	<b>131</b>
10.1	Run Schema Manager.....	135
10.2	Status Categories.....	138
10.3	Patch or Install a Schema.....	140
10.4	Uninstall a Schema, Reset.....	142
10.5	Command Line Interface (CLI).....	143
10.5.1	help .....	143
10.5.2	info .....	144
10.5.3	initialize.....	144
10.5.4	install .....	145

---

10.5.5	list .....	145
10.5.6	reset .....	146
10.5.7	uninstall.....	147
10.5.8	update.....	148
10.5.9	upgrade.....	148
<b>11</b>	<b>Authentic Desktop in Visual Studio</b>	<b>149</b>
11.1	Installing the Authentic Desktop Plugin for Visual Studio.....	150
11.2	Differences with Standalone Version.....	151
<b>12</b>	<b>Authentic Desktop in Eclipse</b>	<b>152</b>
12.1	Install the Integration Package for Eclipse.....	153
12.2	Authentic Desktop Perspective in Eclipse.....	155
12.3	Other Authentic Desktop Entry Points in Eclipse.....	158
<b>13</b>	<b>Menu Commands</b>	<b>159</b>
13.1	File Menu.....	160
13.1.1	New .....	160
13.1.2	Open .....	161
13.1.3	Reload.....	166
13.1.4	Encoding.....	166
13.1.5	Close, Close All, Close All But Active.....	167
13.1.6	Save, Save As, Save All.....	167
13.1.7	Send by Mail.....	172
13.1.8	Print .....	173
13.1.9	Print Preview, Print Setup.....	173
13.1.10	Recent Files, Exit.....	174
13.2	Edit Menu.....	175
13.2.1	Undo, Redo.....	175
13.2.2	Cut, Copy, Paste, Delete.....	175
13.2.3	Select All.....	176
13.2.4	Find, Find Next.....	176
13.2.5	Replace.....	177

---

13.3	Project Menu.....	178
13.3.1	New Project.....	181
13.3.2	Open Project.....	181
13.3.3	Reload Project.....	181
13.3.4	Close Project.....	181
13.3.5	Save Project, Save Project As.....	182
13.3.6	Source Control.....	182
13.3.7	Add Files to Project.....	196
13.3.8	Add Global Resource to Project.....	196
13.3.9	Add URL to Project.....	196
13.3.10	Add Active File to Project.....	197
13.3.11	Add Active And Related Files to Project.....	197
13.3.12	Add Project Folder to Project.....	197
13.3.13	Add External Folder to Project.....	197
13.3.14	Add External Web Folder to Project.....	200
13.3.15	Script Settings.....	203
13.3.16	Properties.....	204
13.3.17	Most Recently Used Projects.....	207
13.4	XML Menu.....	208
13.4.1	Check Well-Formedness.....	208
13.4.2	Validate XML.....	208
13.4.3	Validate on Edit.....	209
13.5	XSL/XQuery Menu.....	210
13.5.1	XSL Transformation.....	210
13.5.2	XSL-FO Transformation.....	211
13.5.3	XSL Parameters / XQuery Variables.....	212
13.6	Authentic Menu.....	217
13.6.1	New Document.....	218
13.6.2	Edit Database Data.....	218
13.6.3	Edit StyleVision Stylesheet.....	219
13.6.4	Select New Row with XML Data for Editing.....	219
13.6.5	XML Signature.....	220
13.6.6	Define XML Entities.....	222
13.6.7	View Markup.....	224
13.6.8	RichEdit.....	224

---

13.6.9	Append/Insert/Duplicate/Delete Row.....	224
13.6.10	Collapse/Expand Markup.....	225
13.6.11	Move Row Up/Down.....	225
13.6.12	Generate HTML, RTF, PDF, Word 2007+ Document.....	225
13.6.13	Trusted Locations.....	226
13.7	View Menu.....	227
13.7.1	Authentic View.....	227
13.7.2	Browser View.....	227
13.8	Browser Menu.....	228
13.9	Tools Menu.....	229
13.9.1	Spelling.....	229
13.9.2	Spelling Options.....	232
13.9.3	Scripting Editor.....	234
13.9.4	Macros.....	235
13.9.5	User-Defined Tools.....	235
13.9.6	Global Resources.....	235
13.9.7	Active Configuration.....	236
13.9.8	XML Schema Manager.....	237
13.9.9	Customize.....	237
13.9.10	Restore Toolbars and Windows.....	253
13.9.11	Options.....	253
13.10	Window Menu.....	272
13.11	Help Menu.....	274
13.11.1	Help .....	274
13.11.2	Keyboard Map.....	274
13.11.3	Activation, Order Form, Registration, Updates.....	275
13.11.4	Other Commands.....	279
13.12	Command Line.....	280

## **14 Programmers' Reference 281**

14.1	Scripting Editor.....	283
14.1.1	Creating a Scripting Project.....	284
14.1.2	Built-in Commands.....	297
14.1.3	Enabling Scripts and Macros.....	307



---

14.2	IDE Plugins.....	310
14.2.1	Registration of IDE Plugins.....	310
14.2.2	ActiveX Controls.....	311
14.2.3	Configuration XML.....	311
14.2.4	ATL Sample Files.....	314
14.2.5	IXMLSpyPlugIn.....	320
14.3	Application API.....	325
14.3.1	Overview.....	326
14.3.2	Interfaces.....	355
14.3.3	Enumerations.....	588
14.4	ActiveX Integration.....	603
14.4.1	Prerequisites.....	603
14.4.2	Adding the ActiveX Controls to the Toolbox.....	604
14.4.3	Integration at Application Level.....	606
14.4.4	Integration at Document Level.....	608
14.4.5	ActiveX Integration Examples.....	611
14.4.6	Command Reference.....	624
14.4.7	Object Reference.....	632

## **15 Appendices 653**

15.1	Technical Data.....	654
15.1.1	OS and Memory Requirements.....	654
15.1.2	Altova Engines.....	654
15.1.3	Unicode Support.....	655
15.1.4	Internet Usage.....	655
15.2	License Information.....	656
15.2.1	Electronic Software Distribution.....	656
15.2.2	Software Activation and License Metering.....	657
15.2.3	Altova End-User License Agreement for Authentic.....	658

## **Index 659**

# 1 About Authentic Desktop and This Documentation

[Altova Authentic 2024 Desktop](#) 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 10, Windows 11, and Windows Server 2016 or newer. Authentic Desktop Enterprise Edition is available for 64-bit and 32-bit machines.



*Last updated: 8 April 2024*

## 1.1 Windows File Paths

### File paths in Windows

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 7/8/10/11	C:\Users\ <username>\Documents</username>
-------------------	---

- *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 7/8/10/11	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 2016 or newer.

## 1.2 About This Documentation

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](#)<sup>13</sup> that describes the GUI and the Authentic Desktop environment.
- A [tutorial](#)<sup>23</sup> to get you started using Authentic Desktop.
- A description of [Authentic View](#)<sup>39</sup>, 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](#)<sup>88</sup>, in which the XML document is transformed on the fly and presented in a browser window.
- An explanation of Altova's [Global Resources](#)<sup>90</sup> feature, which enables resources to be quickly switched from one to the other.
- Explanations of how Authentic Desktop can be used in [Visual Studio](#)<sup>149</sup> and [Eclipse](#)<sup>152</sup>.
- A [menu command reference](#)<sup>159</sup> that contains a description of windows and menu commands available in Authentic Desktop.

## 2 GUI and Environment

This section describes:

- [The application GUI](#)<sup>14</sup>, and
- [The application environment](#)<sup>21</sup>.

The [GUI section](#)<sup>14</sup> 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.

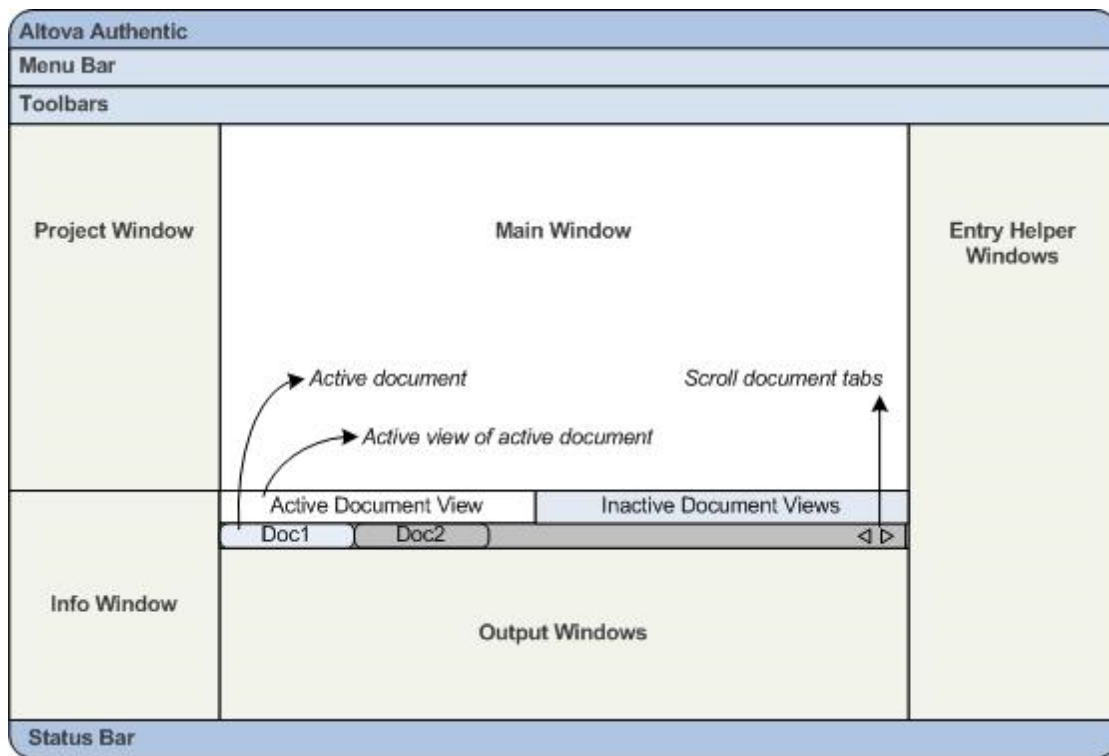
The [Application Environment section](#)<sup>21</sup> points out the various settings that control how files are displayed and can be edited. It also explains how and where you can customize your application. In this section, you will learn where important example and tutorial files have been installed 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.

## 2.1 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.



The GUI also contains a menu bar, status bar, and toolbars, all of which are described in a subsection 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.

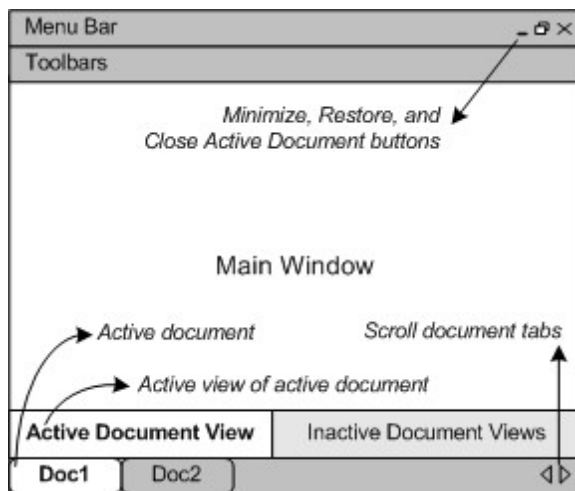
## Auto-hiding sidebars

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**).

### 2.1.1 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](#)<sup>272</sup> 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](#)<sup>272</sup> 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](#)<sup>227</sup> menu.

The available views are either editing or browser views:

- [Authentic View](#)<sup>39</sup>: For editing XML documents that are based on StyleVision Power Stylesheets in a graphical interface.
- [Browser View](#)<sup>88</sup>: An integrated browser view that supports both CSS and XSL stylesheets.

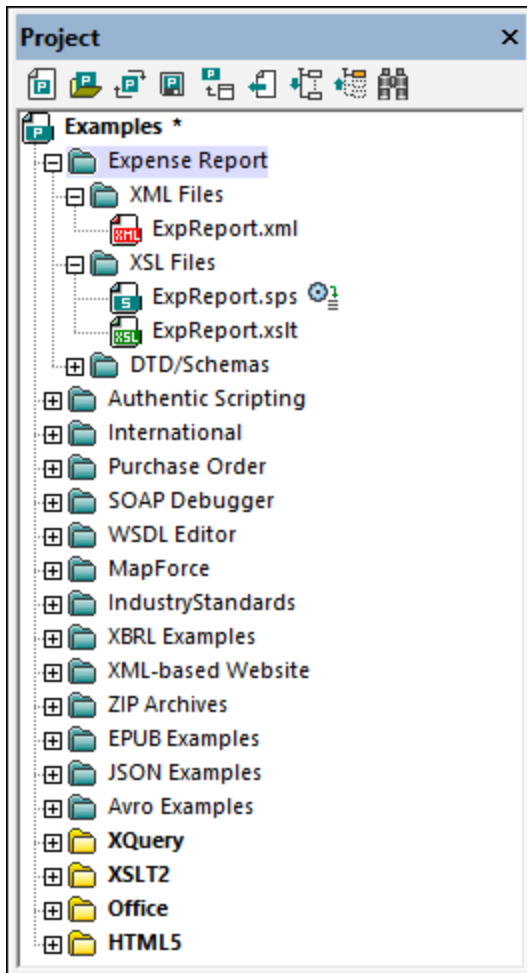
**Note:** The default view for individual file extensions can be customized in the [Tools | Options](#)<sup>253</sup> dialog: in the Default View pane of the File Types tab.



## 2.1.2 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 shown in the screenshot, for instance, the `Expense Report` folder is further organized into sub-folders for XML, XSL, and Schema files.

**Note:** The Project Window of Authentic Desktop will initially contain the application's default `Examples` project. To load the default `Examples` project, go to the application's `Examples` folder in the [\(My\) Documents folder](#)<sup>11</sup>, and double-click the file `Examples.spp`.



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). A subset of **Project** menu commands, because they are frequently used, are also available in the toolbar of the Project Window (*screenshot below*).



The toolbar commands are, from left: *New Project*, *Open Project*, *Reload Project*, *Save Project*, *Add Active File to Project*, *Select Active File*, *Expand All*, *Collapse All*, *Find*. The names of these commands are self-explanatory and are explained in the [Project menu](#)<sup>178</sup>.

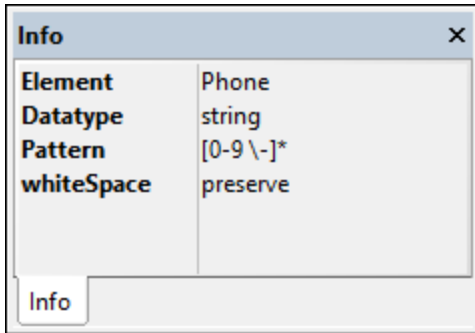
The key operations related to the Project Window are listed below.

- 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.
- When you hover over an image file that has been placed in a project folder, a preview of the image is displayed (.png, .jpeg, .gif, .bmp, .tiff, and .ico formats). Double-click the image to open it in the system's default image viewer/editor program.
- 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.

### 2.1.3 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.

### 2.1.4 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:** You can turn the display of entry helpers on or off with the menu option **Window | Entry Helpers**.

### 2.1.5 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](#) <sup>208</sup> 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](#) <sup>208</sup> or **F7**) in the Project window.

## 2.1.6 Menu Bar, Toolbars, Status Bar

### Menu Bar

The menu bar ([see illustration](#)<sup>14</sup>) 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 **F**ile 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 (**F**ile | **N**ew), 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](#)<sup>242</sup> in the Keyboard tab of the Customize dialog (**T**ools | **C**ustomize).

### Toolbars

The toolbars ([see illustration](#)<sup>14</sup>) 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](#)<sup>239</sup> 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](#)<sup>239</sup>. 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](#)<sup>14</sup>) 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.

## 2.2 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](#)<sup>21</sup>: Describes how and where important settings and customization options can be defined.
- [Tutorials, Projects, Examples](#)<sup>21</sup>: Notes the location of the various non-program files included in the application package.
- [Product features and documentation, and Altova products](#)<sup>22</sup>: Provides links to the [Altova website](#), where you can find information about product features, additional Help formats, and other Altova products.

### 2.2.1 Settings and Customization

This section provides a brief overview of aspects that allow you to personalize Authentic Desktop.

#### 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](#)<sup>237</sup>). The various customization options are described in the [Menu Reference](#)<sup>159</sup> section.

### 2.2.2 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:\Users\\Documents\Altova\Authentic2024\AuthenticExamples\
```

The `My Documents\Altova\Authentic2024` 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.

## Location of tutorial, project, and examples files

All tutorial, project, and example files are located in the `AuthenticExamples` folder.

## 2.2.3 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](#).

### 3 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: <input type="text" value="US"/>			
Street:	119 Oakstreet, Suite 4876	Phone:	+1 (321) 555 5155 0
City:	Vereno	Fax:	+1 (321) 555 5155 4
State & Zip:	<input type="text" value="DC"/> <input type="text" value="29213"/>	E-mail:	<a href="mailto:office@nanonull.com">office@nanonull.com</a>
<b><u>Vereno Office Summary:</u> 4 departments, 15 employees.</b>			
<p>The company was established in <b>Vereno in 1995</b> 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 <i>NanoSoft Development Suite</i> 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.</p>			

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](#)<sup>39</sup> section.

#### Tutorial requirements

All **the files** you need for the tutorial are in the `AuthenticExamples` 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)

**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.



## 3.1 Opening an XML Document in Authentic View

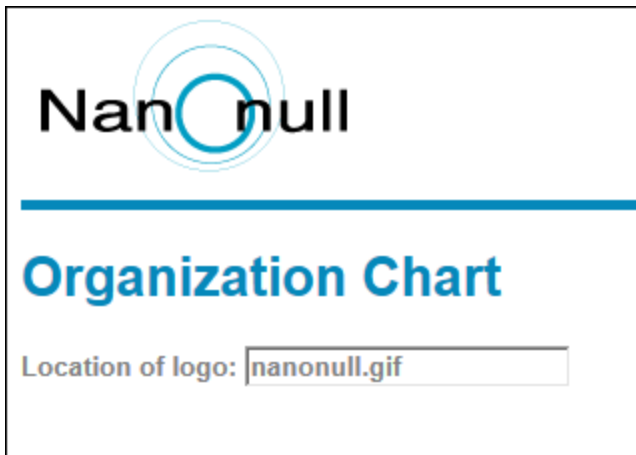
In Authentic View, you can edit an existing XML document or create and edit a new XML document. In this tutorial, you will open an existing XML document in Authentic View (described in this section) and learn how you can edit it (subsequent sections). Additionally, in this section is a description of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



**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**.
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.

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.


## 3.2 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:

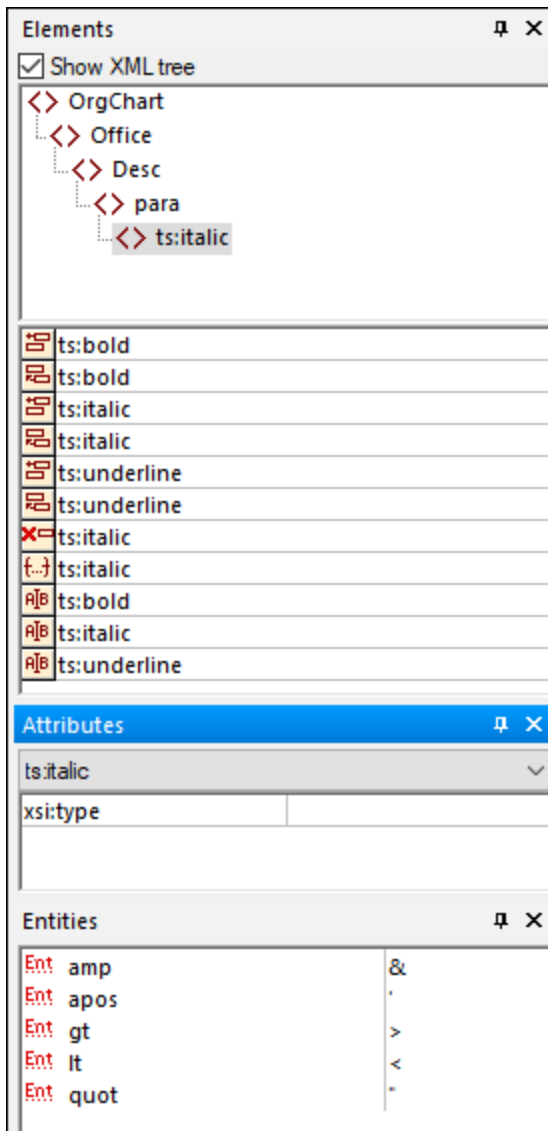
```
<DocumentRoot>
  <Person id="ABC001">
    <Name>Alpha Beta</Name>
    <Address>Some Address</Address>
    <Tel>1234567</Tel>
  </Person>
</DocumentRoot>
```

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](#)<sup>39</sup>.

### 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](#)<sup>39</sup>

## 3.3 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](#)<sup>27</sup>. 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](#)<sup>27</sup>.




**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](#)<sup>29</sup> or by clicking the node operation entry in the [Elements entry helper](#)<sup>27</sup>. In some cases, an element or attribute can be added by clicking the [Add Node link](#)<sup>30</sup> in the Authentic View of the document. In the special cases of elements defined as paragraphs or list items, pressing the [Enter key](#)<sup>31</sup> 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](#)<sup>31</sup>, [Remove Node](#)<sup>31</sup>, and [Clear Element](#)<sup>32</sup> 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 `NanonullOrg.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 `add @href` link appears within the `CompanyLogo` element that was edited

(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 `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.

**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  icon. In the `NanonullOrg.xml` document, select text inside the mixed content `para` element and experiment with applying the `bold` and `italic` elements.



The stylesheet designer might also have created a toolbar icon to apply an element. In the `NanonullOrg.xml` document, the `bold` and `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](#)<sup>63</sup> section.




## 3.4 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](#)<sup>36</sup>. 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 <b>USA</b> , Inc.	
Location:	US ▼
Street:	119 Oakstreet, Suite 4876
City:	Vereno
State & Zip:	DC ▼ 29213

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](#)<sup>37</sup> 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](#)<sup>81</sup>, which appears in the Entities entry helper. The [entity is inserted](#)<sup>37</sup> 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 markup on](#) <sup>27</sup>.

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.

Data-Entry Device	Data in XML File
Input Field (Text Box)	Text entered by user
Multiline Input Field	Text entered by user
Combo box	User selection mapped to value
Check box	User selection mapped to value
Radio button	User selection mapped to value
Button	User selection mapped to value

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.

The screenshot shows a form for 'Nanonull, Inc.' with the following elements:

- Location:** A dropdown menu with 'US' selected.
- Street:** A text input field containing 'kstreet, Suite 4876'.
- City:** A text input field.
- State & Zip:** A dropdown menu with 'DC' selected and a text input field containing '29213'.
- Vereno Office Summary:** A text input field containing 'tments, 15 employees.'

A vertical list of state abbreviations (AK, AL, AR, AZ, CA, CO, CT, DC, DE, FL, GA, GU) is shown next to the State & Zip dropdown, with 'DC' highlighted.

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

```
<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.

## 3.5 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

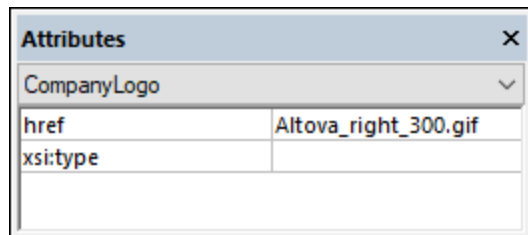
### Attribute values in the main window

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](#)<sup>33</sup>. 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 `AuthenticExamples` folder).

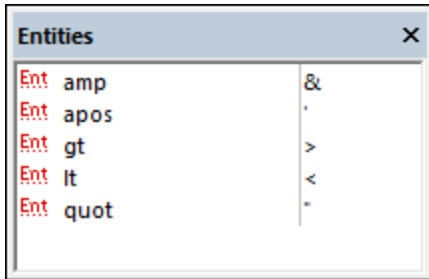


This causes the Nanonull logo to be replaced by the Altova logo.

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

## 3.6 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*).

Marketing ( 2 )		
First	Last	Title
Joe	Martin	Marketing Manager Europe &
Susi	Sanna	Art Director
Employees: 2 (13% of Office, 5% of Company)		
Non-Shareholders: None.		

**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](#)<sup>81</sup>.

## 3.7 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.

Page 1 of 5

# ALTOVA

[www.altova.com](http://www.altova.com)

---

## Organization Chart

Location of logo:

---

### Nanonull, Inc.

Location:

<b>Street:</b>	119 Oakstreet, Suite 4876	<b>Phone:</b>	+1 (321) 555 5155 0
<b>City:</b>	Vereno	<b>Fax:</b>	+1 (321) 555 5155 4
<b>State &amp; Zip:</b>	<input type="text" value="DC"/> <input type="text" value="29213"/>	<b>E-mail:</b>	<a href="mailto:office@nanonull.com">office@nanonull.com</a>

**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.

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.

## 4 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

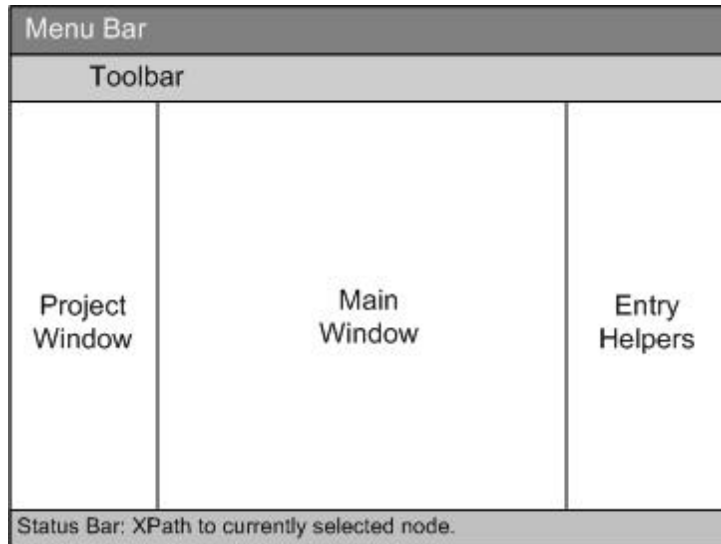
Additional sources of Authentic View information are:

- 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](#)

## 4.1 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](#)<sup>42</sup>.

### 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.

### Info window

This window provides information about the node that is currently selected in Authentic View.

### 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](#)<sup>45</sup>.



## 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](#)<sup>48</sup> for details.

## Status Bar

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

## Context menus

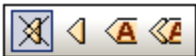
These are the menus that appear when you right-click in the Main Window. The available commands are context-sensitive editing commands, i.e. they allow you to manipulate structure and content relevant to the selected node. Such manipulations include inserting, appending, or deleting a node, adding entities,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





## 4.2 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

**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](#) <sup>63</sup> 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](#) <sup>68</sup>.



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](#) <sup>63</sup>.


## 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; Go to Previous Record; Open the *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.

## 4.3 Authentic View Main Window

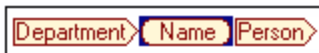
There are four viewing modes in Authentic View: Large Markup; Small Markup; Mixed Markup; and Hide All Markup. These modes enable you to view the document with varying levels of markup information. To switch between modes, use the commands in the **Authentic** menu or the icons in the toolbar (see the previous section, [Authentic View toolbar icons](#)<sup>42</sup>).

### Large markup

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



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.



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



### Small markup

This shows the start and end tags of elements/attributes without names:



⌘ Nanonull, Inc. ⌘

Location: ⌘ US ⌘

<p>⌘</p> <p><b>Street:</b> ⌘ 119 Oakstreet, Suite 4876 ⌘</p> <p><b>City:</b> ⌘ Vereno ⌘</p> <p><b>State &amp; Zip:</b> ⌘ DC ⌘</p> <p>29213 ⌘</p>	<p><b>Phone:</b> ⌘ +1 (321) 555 5155 0 ⌘</p> <p><b>Fax:</b> ⌘ +1 (321) 555 5155 4 ⌘</p> <p><b>E-mail:</b> ⌘ <a href="mailto:office@nanonull.com">office@nanonull.com</a> ⌘</p>
--	--

⌘

⌘ ⌘ [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.

⌘ ⌘

Notice that start tags have a symbol inside them while end tags are empty. Also, element tags have an angular-brackets symbol while attribute tags have an equals sign as their symbol (see screenshot below).

⌘ ⌘ 2006-04-01 ⌘ ⌘ Boston ⌘, ⌘ USA ⌘ ⌘ ⌘

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.

⌘ ⌘ ⌘ **Office Summary: 4 departments, 15 employees.** ⌘ ⌘

## 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.



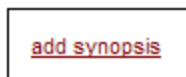
- Data-entry devices. The display contains either an input field (text box), a multiline input field, combo box, check box, or radio button. In the case of input fields and multiline input fields, the text you enter in the field becomes the XML content of the element or the value of the attribute.



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

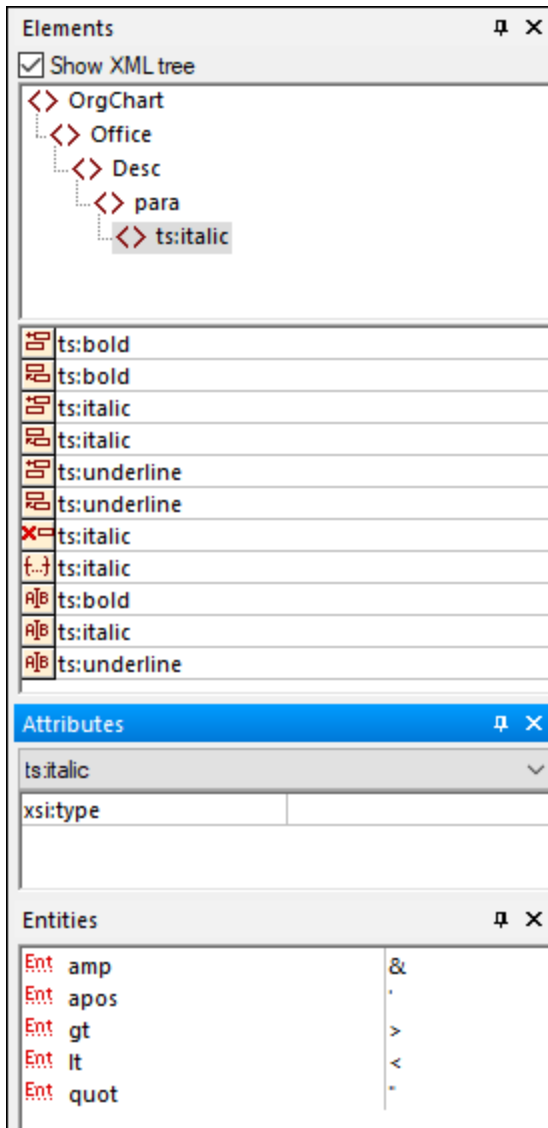
When an element or attribute is **optional** (according to the referenced schema), a prompt of type `add [element/attribute]` is displayed:



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.

## 4.4 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; removed; 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*

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.

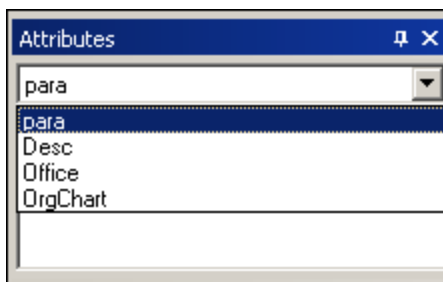


#### *Clear Element (when insertion point selected)*

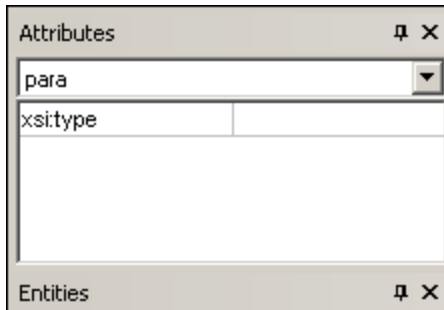
This icon appears when the cursor is placed within an element that is a child of a mixed-content element. Clicking the icon clears the inline element.

## 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`.



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`.)



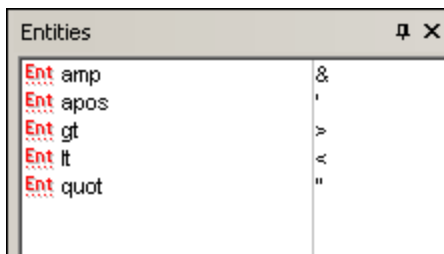
To enter a value for an attribute, click in the value field of the attribute and enter the value. This creates the attribute and its value in the XML document.

Note the following:

- 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 either (i) selecting a value from the dropdown list that appears, or (ii) entering a value. Values displayed in the dropdown list 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](#) <sup>81</sup> in the Editing in Authentic View section.

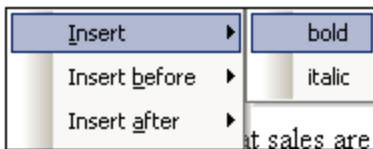


## 4.5 Authentic View Context Menus

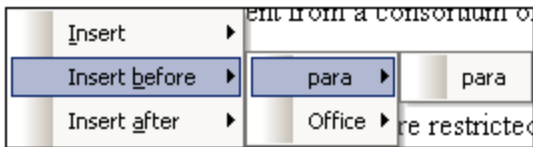
Right-clicking on some selected document content or node pops up a context menu with commands relevant to the selection or cursor location.

### 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](#)<sup>48</sup> 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](#)<sup>81</sup> 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](#)<sup>48</sup>.

## 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).

**Note:** 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 node or by clicking either the start or end tag of the node.

## 5 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:

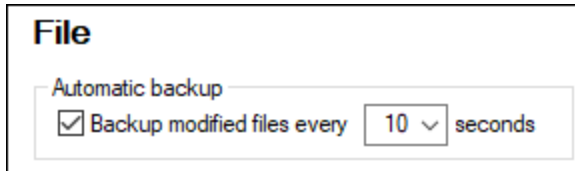
- There are three distinct types of tables used in Authentic View. The section [Using tables in Authentic View](#)<sup>63</sup> explains the three types of tables (static SPS, dynamic SPS, and XML), and when and how to use them. It starts with the broad, conceptual picture and moves to the details of usage.
- The Date Picker is a graphical calendar that enters dates in the correct XML format when you click a date. See [Date Picker](#)<sup>78</sup>.
- 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](#)<sup>81</sup> for details.
- In the Enterprise and Professional editions of Altova products, Authentic View users can sign XML documents with [digital XML signatures](#)<sup>83</sup> and verify these signatures.
- What [image formats](#)<sup>84</sup> can be displayed in Authentic View.

Altova website: [XML content editing](#), [XML authoring](#)

## 5.1 Automatic Backup of Files

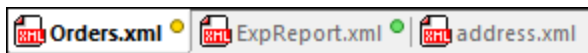
Files that are modified in Authentic Desktop are automatically backed up at regular intervals. In the *File* tab of the Options dialog ([Tools | Options | File](#) <sup>254</sup>) shown in the screenshot below, you can:

- Switch on/off automatic backups
- Specify the frequency of backups (5 seconds to 300 seconds)



### Indicators

The file tabs at the bottom of the Main Window contain symbols to the right of the file name which indicate the saved/unsaved state and backup state of the file (*screenshot below*).



### Saved / Unsaved

A colored circle symbol is present if a file has been modified. If no such symbol is present, it means that the file has not been modified since either being opened or being last saved. In the screenshot above, for example, `address.xml`.

### Backup state

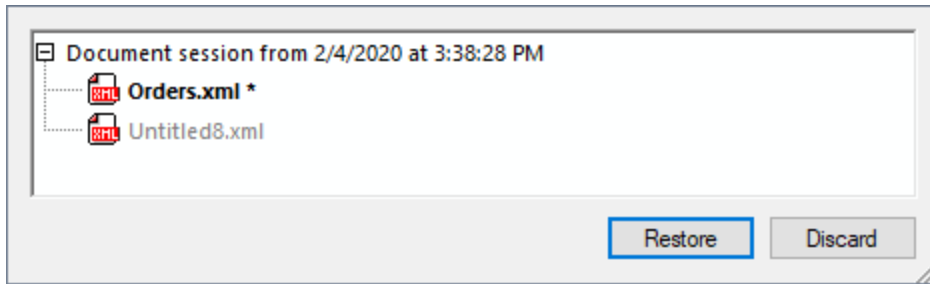
The colors of the circle symbols indicate the backup state of the file.

- **Yellow:** The file has been modified, but the last modification has not been backed up (or saved).
- **Green:** The file has been backed up, and it has not been modified since being backed up. However, the file has not been saved. (If it had been saved, there would be no circle symbol.)
- **Red:** Backup is not supported for this file or a backup has failed.
- **Gray:** The automatic backup function has been disabled (via the [Options dialog](#) <sup>253</sup>; see above). The presence of the symbol, however, indicates that the file has not been saved since last modification. (If it had been saved, there would be no circle symbol.)

### Restoring from backups

If Authentic Desktop terminates unexpectedly, then, at the next application start, a Restore Document dialog is displayed which contains a list of all documents that were open at the time of the application being terminated (*screenshot below*). You can hover over each file to see its path. In the case of temporary files that have not yet been saved, the filepath will be the current default path were a Save As dialog opened for that file.





For each file in the list, its font style and the presence/absence of asterisks provide the following information:

- A bold style and asterisk indicates that the file contains unsaved changes. Such files will be restored in their last backed-up state.
- A normal style indicates that the file has been saved and there are no unsaved changes. Such files will be restored in their saved state.
- A grayed out style indicates that the file has neither been saved nor been backed up (for example, because it is a new file that was not edited). Such files will not be restored.

You can now do one of the following:

- Click **Restore** to restore the files in the GUI from their last backed-up state.
- Click **Discard** to not open any of the listed files and to discard any available backups.

## 5.2 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*).

### Vereno Office Summary: 4 departments, 16 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.

An editable Authentic View document with no XML markup.

Address ipo:city Vereno ipo:city Address Office Summary:  
4 departments, 16 employees. Desc para

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.

para para

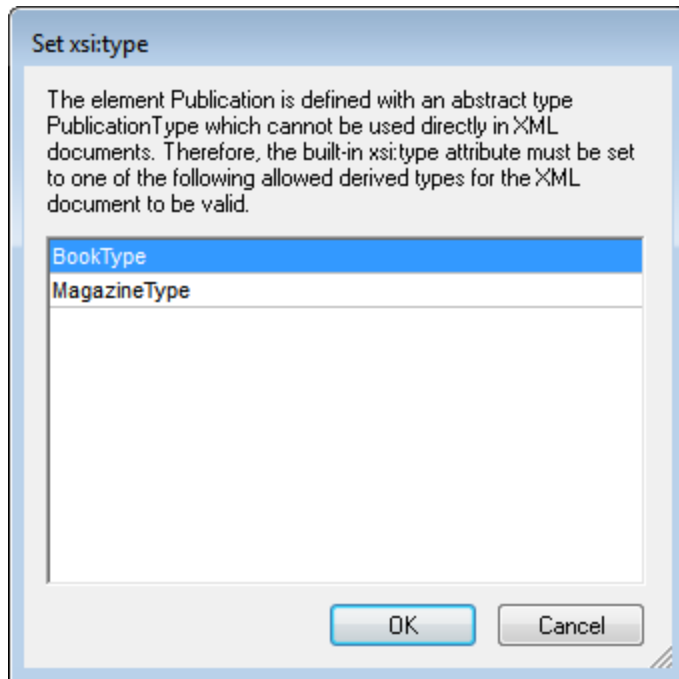
An editable Authentic View document with XML markup tags.

### 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:

```
<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](#) <sup>53</sup>.

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](#) <sup>48</sup> section of the documentation.

## Using RichEdit in Authentic View

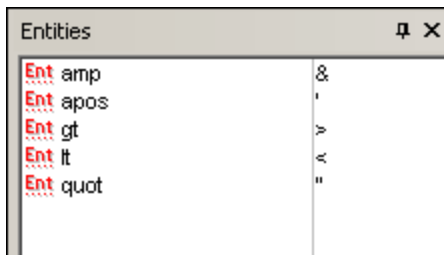
In Authentic View, when the cursor is placed inside an element that has been created as a RichEdit component, the buttons and controls in the RichEdit toolbar (*screenshot below*) become enabled. Otherwise they are grayed out.



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](#)<sup>51</sup> (*screenshot below*).



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](#)<sup>81</sup> within the Authentic View application. Once defined, these entities appear in the [Entities Entry Helper](#)<sup>51</sup> 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 `]]>` at the start and end, respectively. 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](#)<sup>42</sup>. 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.)

## 5.3 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](#)<sup>63</sup> 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](#)<sup>64</sup> and the [CALS/HTML Table editing icons](#)<sup>68</sup> are described below.

### 5.3.1 SPS Tables

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

#### Static 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.

Nanonull, Inc.	
<b>Street:</b>	119 Oakstreet, Suite 4876
<b>City:</b>	Vereno
<b>State &amp; Zip:</b>	DC 29213
<b>Phone:</b>	+1 (321) 555 5155
<b>Fax:</b>	+1 (321) 555 5155 - 9
<b>E-mail:</b>	office@nanonull.com

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

#### Dynamic 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.

Administration								
First	Last	Title	Ext	Email	Shares	Leave		
						Total	Used	Left
Vernon	Callaby	Office Manager	581	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	<a href="#">add Shares</a>	25	7	18
<b>Employees: 3 (20% of Office, 9% of Company)</b>					<b>Shares: 1500 (13% of Office, 6% of Company)</b>			
<b>Non-Shareholders: Frank Further, Loby Matise.</b>								


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.

### 5.3.2 CALS/HTML Tables

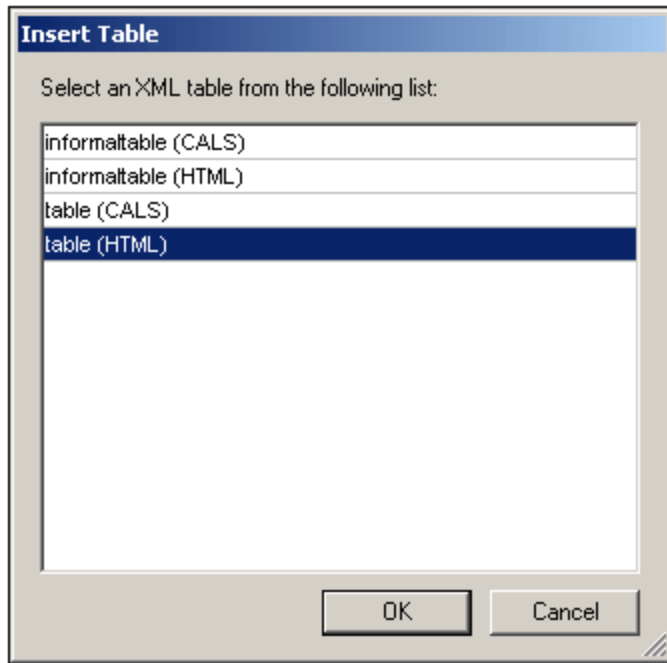
CALS/HTML tables can be inserted by you, the user of Authentic View, for certain XML data structures that have been specified to show a table format. There are three steps involved when working with CALS/HTML tables: inserting the table; formatting it; and entering data. The commands for working with CALS/HTML tables are available as icons in the toolbar (see [CALS/HTML table editing icons](#)<sup>68</sup>).

#### Inserting tables

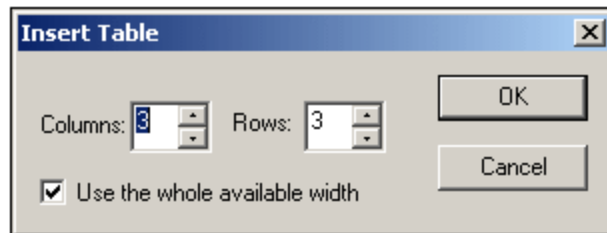
To insert a CALS/HTML table do the following:

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 (*screenshot below*) 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 (*screenshot below*), 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.


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:

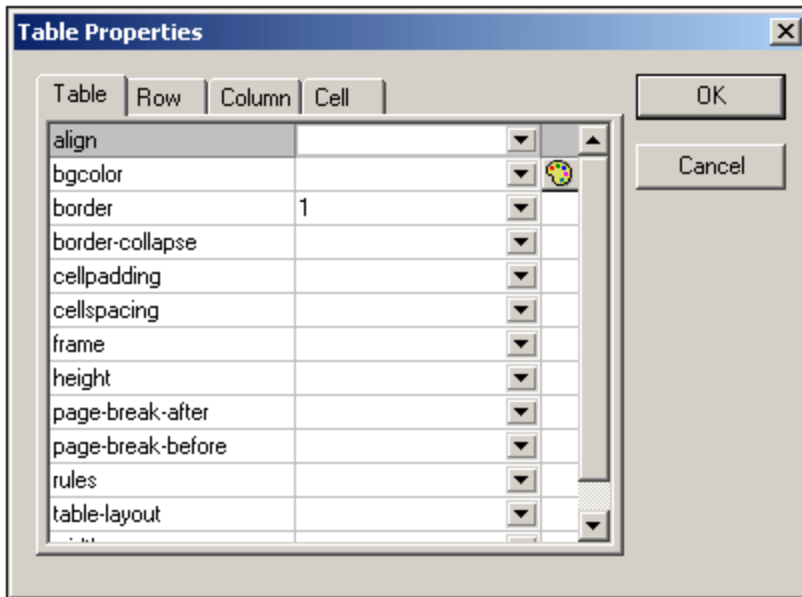
- The elements corresponding to the various table structure elements must have the relevant CALS or HTML table properties defined as attributes (in the underlying XML Schema). Only those attributes that

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.

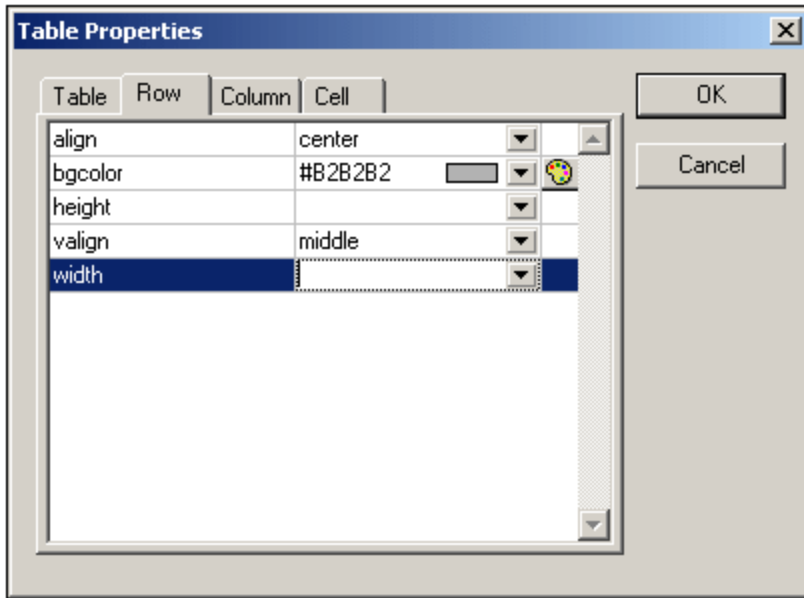
To format a table, row, column, or cell, do the following:

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:



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




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:

Name	Telephone	Email

Notice that the alignment is centered as specified.

- 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:

Name	Telephone	Email

- 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:

Name	Telephone		Email
	Office	Home	

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](#)<sup>68</sup>.

### 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.

<b>Name</b>	<b>Telephone</b>		<b>Email</b>
	<b>Office</b>	<b>Home</b>	

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.

**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.

## 5.3.3 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](#)<sup>64</sup>.

### *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

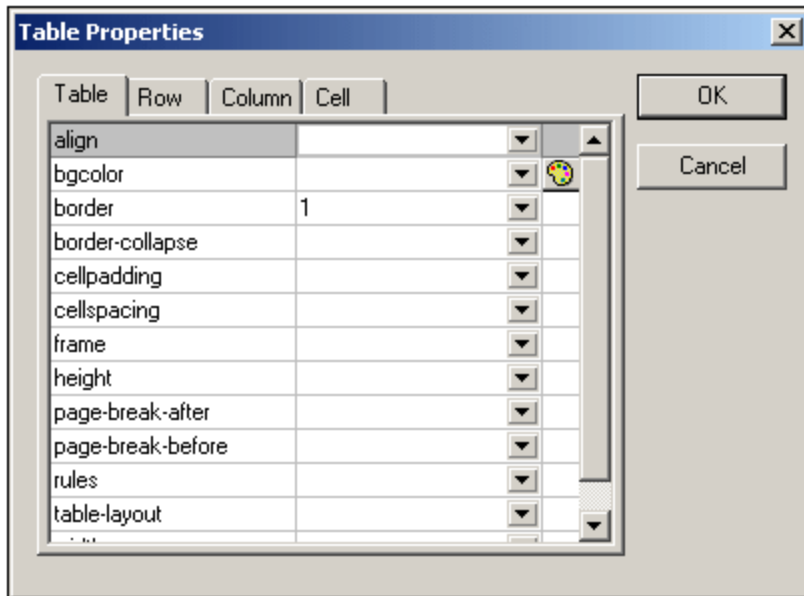


This command aligns the cell contents to the bottom of the cell.

### 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.



## 5.4 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](#)<sup>71</sup>), you can load and display the other records in the DB table.
- You can [query the DB](#)<sup>72</sup> 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](#)<sup>76</sup>.

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

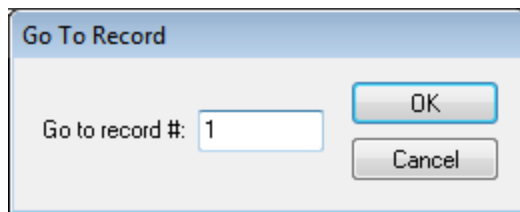
**Note:** In Authentic View, data coming from a SQLite database is not editable. When you attempt to save SQLite data in Authentic View, a message box will inform you of this known limitation.

### 5.4.1 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.



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.



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](#)<sup>219</sup> menu command.

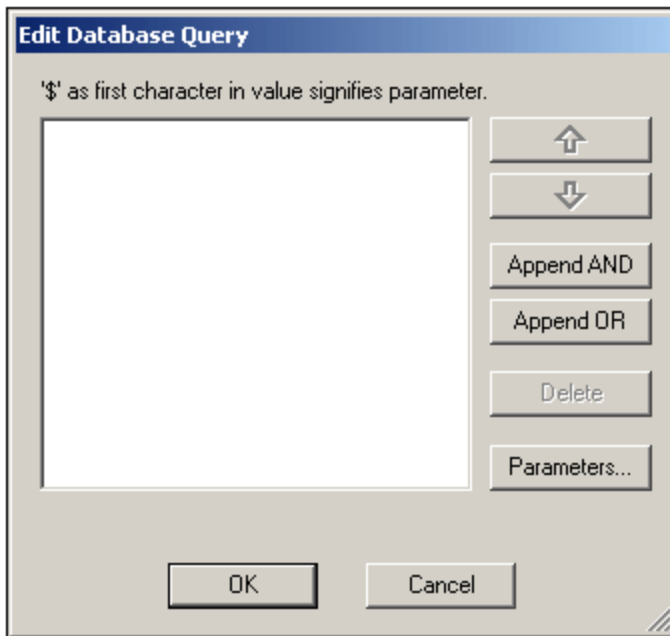
## 5.4.2 DB Queries

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/discarded. After you submit the query, the table is reloaded using the query conditions.

**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.

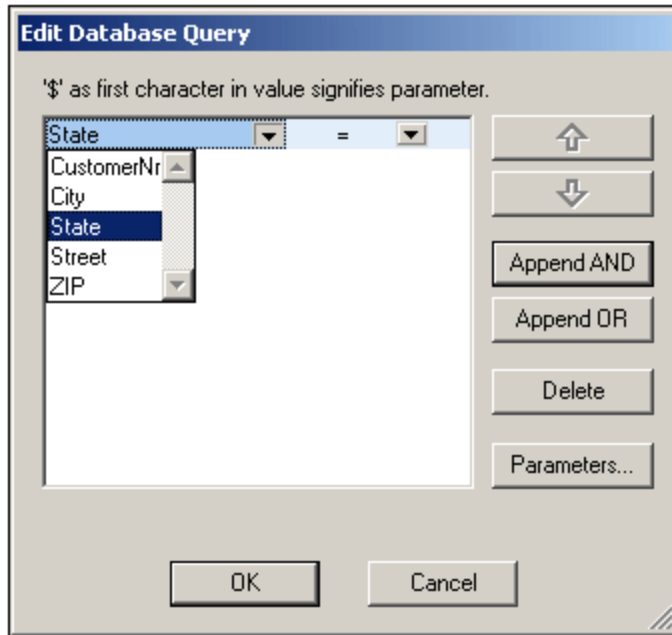
To create and submit a query:

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).





3. 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](#)<sup>73</sup> section.
4. 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](#)<sup>74</sup>.

### 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

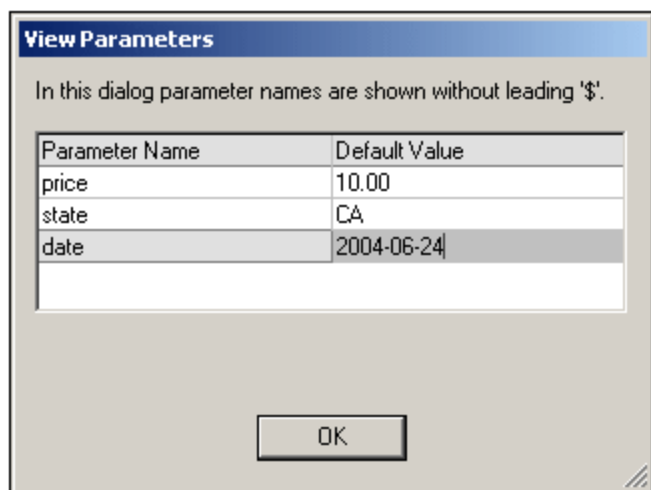
If **IS NULL** or **NOT NULL** is selected, the Value field is disabled. **Values** must be entered without quotes (or any other delimiter). Values must also have the same formatting as that of the corresponding DB field; otherwise

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).

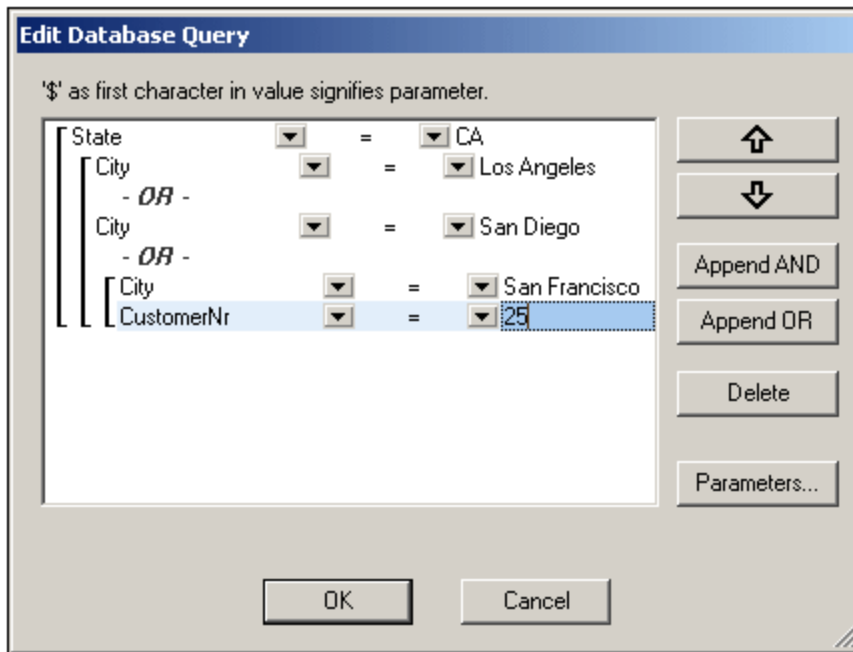
To view the parameters defined for the SPS, click the **Parameters** button in the Edit Database Query dialog. This opens the **View Parameters** dialog (see *screenshot*).



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:

```
State=CA AND (City=Los Angeles OR City=San Diego OR (City=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 . 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.

## 5.4.3 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](#) <sup>71</sup>).

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.


Note the following points:

- 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.

**Note:** Saving data to the DB resets the Undo command, so you cannot undo actions that were carried out prior to the save.

## 5.5 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](#)<sup>78</sup>.
- Dates are entered or modified by [typing in the value](#)<sup>79</sup>.

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.

### 5.5.1 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.

Location of logo:

Last Updated: 2003-09-01

**Nanonull, Inc.**

Location:

M	T	W	T	F	S	S
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today No Timezone

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.

## 5.5.2 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.

Invoice Number: 001
2006-03-10
Customer: The ABC Company
Invoice Amount: 40.00

## Errors

The following types of error will be flagged:

- 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*).

Invoice Number: 001
2006-03-32
Customer: <span style="background-color: yellow;">ERROR: Invalid value for datatype date in element 'InvoiceDate'</span>
Invoice Amount: 40.00

- If you try to change the format of the date, the date turns red to alert you to the error. (In the screenshot below, slashes are used instead of hyphens).

Invoice Number: 001  
2006/03/10  
Customer: The ABC Company  
Invoice Amount: 40.00



## 5.6 Defining Entities

### About 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.

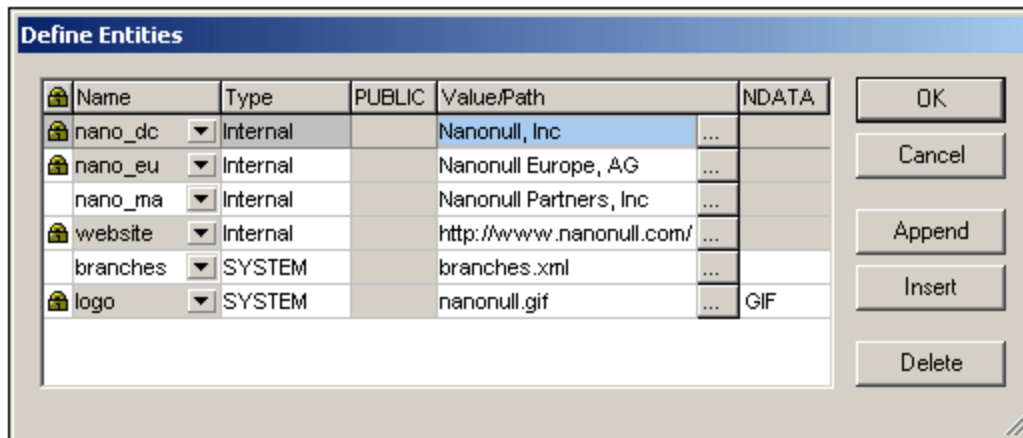
### Types of entity

There are two broad types of entity 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.

### Defining entities

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 is 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. `&amp;`.
- 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).


## 5.7 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.

## 5.8 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.

## 5.9 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).

**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.

## 6 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.

Altova Product	Authentic Scripts Creation	Authentic Scripts Enabled
StyleVision Enterprise	Yes	Yes
StyleVision Professional	Yes	Yes
StyleVision Basic *	No	No
XMLSpy Enterprise	No	Yes
XMLSpy Professional	No	Yes
AuthenticDesktop Enterprise	No	Yes
Authentic Browser Plug-in Enterprise Trusted **	No	Yes
Authentic Browser Plug-in Enterprise Untrusted	No	Yes

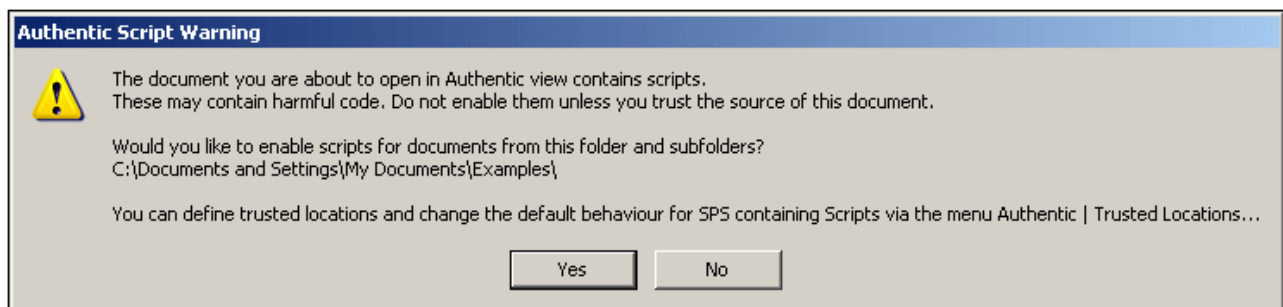
\* No AuthenticView

\*\* 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](#)<sup>226</sup>, 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](#)<sup>226</sup>, and add the folder or modify as required.

For a description of the Trusted Locations dialog, see the description of the [Authentic | Trusted Locations](#)<sup>226</sup> menu command in the User Reference.

**Note:** When Authentic Desktop is accessed via its COM interface (see [Programmers' Reference](#)<sup>281</sup> 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](#).

## 7 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.

### Browser engines in Browser View

By default, Browser View currently uses Microsoft's Internet Explorer as its browser engine. If you wish to use Microsoft's newer Edge WebView2 browser engine for Browser View, you can select this option in the [View section](#)<sup>261</sup> of the [Options dialog](#)<sup>253</sup>.

**Note:** Since Microsoft Edge WebView2 uses the Chromium software project, on which Google's Chrome browser is based, using WebView2 for Browser View also provides a good preview of the Chrome display of a web page.

### Notes about Microsoft Internet Explorer

Browser View requires Microsoft's Internet Explorer 5.0 or later, or Microsoft Edge WebView2 (*see above*).

Note the following points about Internet Explorer in Browser View:

- 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.
- 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.
- In general, you should check the support for XSLT of your version of Internet Explorer.
- If you encounter problems with the correct display of HTML in Internet Explorer, include the following `meta` tag in the `head` element of your HTML document:

```
<head>
... <meta http-equiv="X-UA-Compatible" content="ie=edge">...
</head>
```

### Developer tools in Browser View

You can use the Developer Tools of the underlying browser to inspect, debug, and test your HTML code. To open the tools, right-click in the Browser View pane and select **Open Developer Tools**.

### Markdown text and Browser View

If a document in Text View is marked up with [Markdown formatting](#), then switching to Browse View converts the Markdown formatting to simple HTML formatting and renders the document as an HTML page in Browser View.



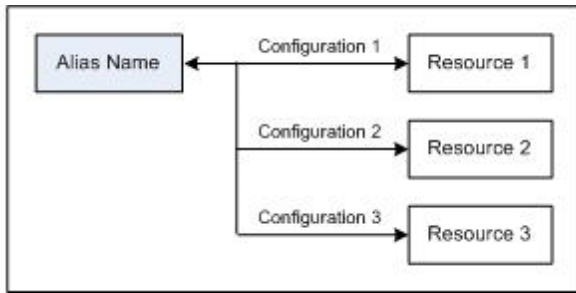
## Browser View features

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. To do this, click the menu command **Browser | Separate Window**. This is a toggle command that switches Browser View between two windows: (i) a separate window, and (ii) a tabbed view in the Main Window. These commands are also available in the dropdown menu of the **Browser View** button (at the bottom of the Main 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.
- *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.
- *Info Window:* There are options here to view the active HTML page with any of the web browsers installed on the machine and to open or remove the installed browsers.

## 8 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](#)<sup>91</sup>: Resources are defined and the definitions are stored in an XML file. These resources can be shared across multiple Altova applications.
- [Using Global Resources](#)<sup>102</sup>: 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.

## 8.1 Defining Global Resources

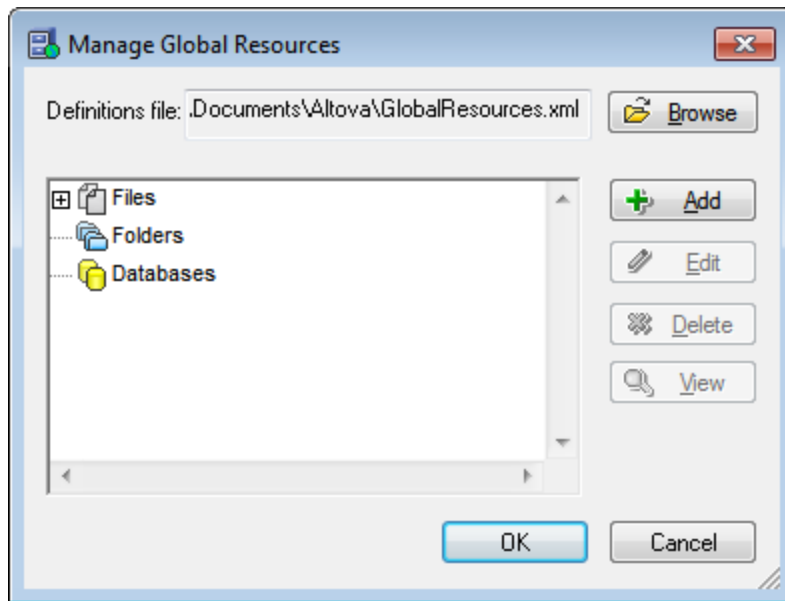
Altova Global Resources are defined in the Manage Global Resources dialog, which can be accessed in two ways:

- 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\\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](#)<sup>93</sup>, [folders](#)<sup>98</sup>, and [databases](#)<sup>91</sup> 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.

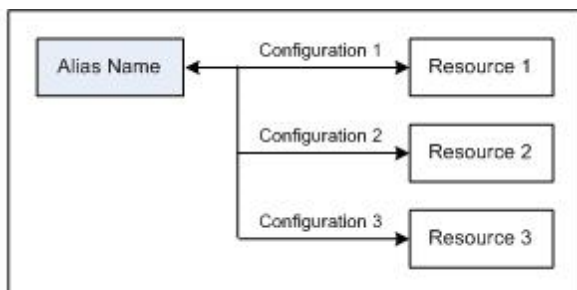
To **edit a global resource**, select it and click **Edit**. This pops up the relevant Global Resource dialog, in which you can make the necessary changes (*see the descriptions of [files](#)<sup>93</sup>, [folders](#)<sup>98</sup>, and [databases](#)<sup>100</sup> in the sub-sections of this section*).

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.

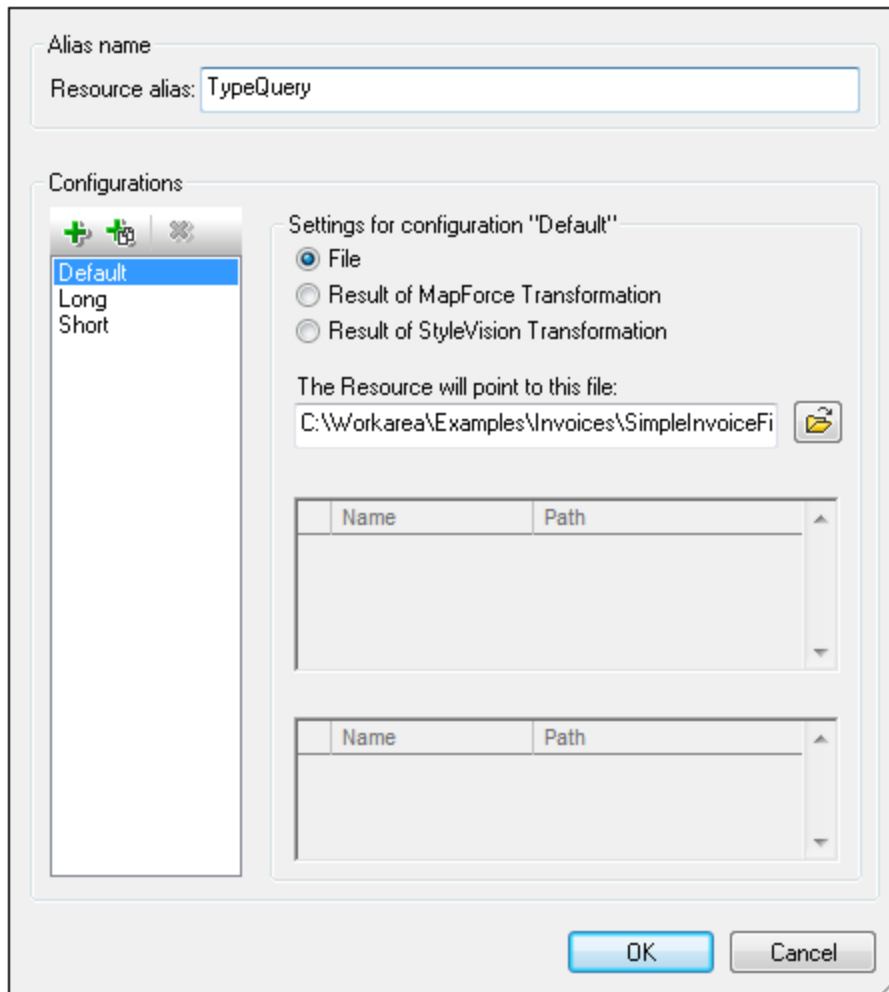
### 8.1.1 Files

The Global Resource dialog for Files (*screenshot below*) is accessed via the **Add | File** command in the [Manage Global Resources dialog](#)<sup>91</sup>. 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 | File** command in the [Manage Global Resources dialog](#)<sup>91</sup>).

#### 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.



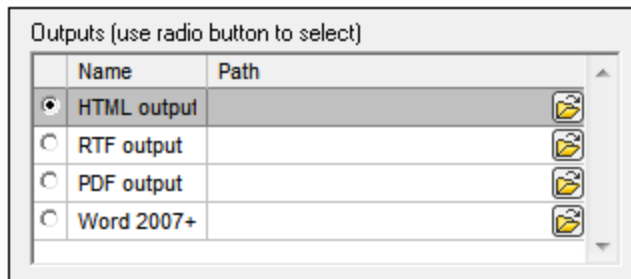
**Open:** Browse for the file to be created as the global resource.


## 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*).



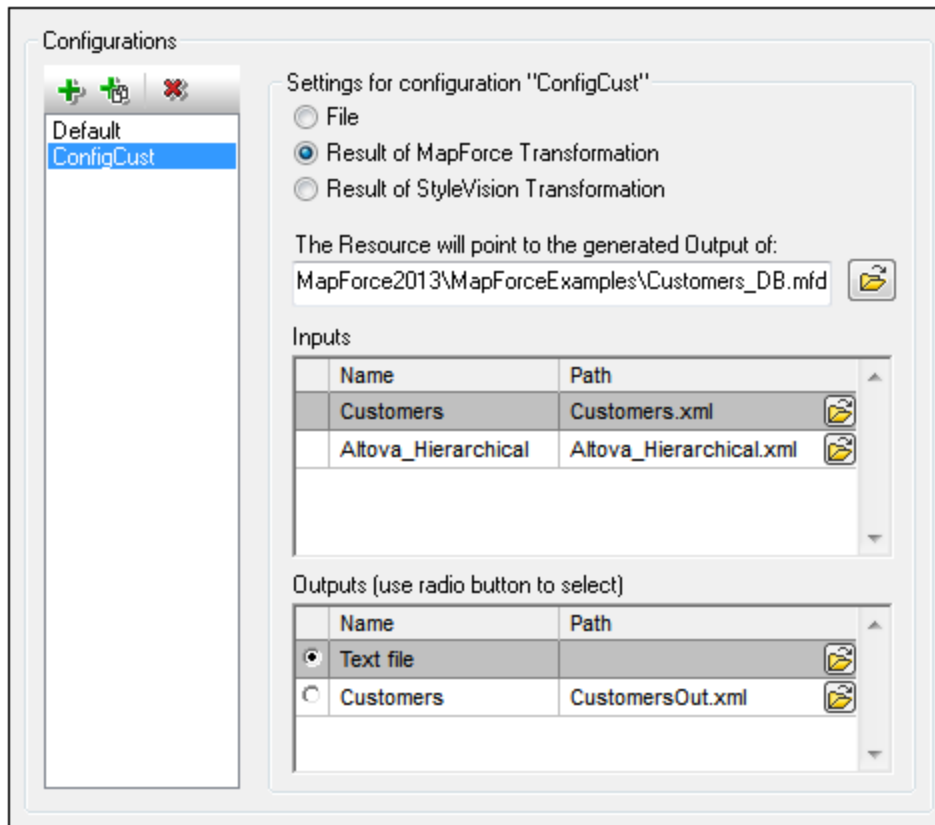
Select the radio button of the desired option (in the screenshot above, 'HTML output' is selected). If the resource is the result of a transformation, then the output can be saved as a file or itself as a global resource. Click the  icon and select, respectively, Global Resource (for saving the output 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](#)<sup>91</sup>.

## 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).

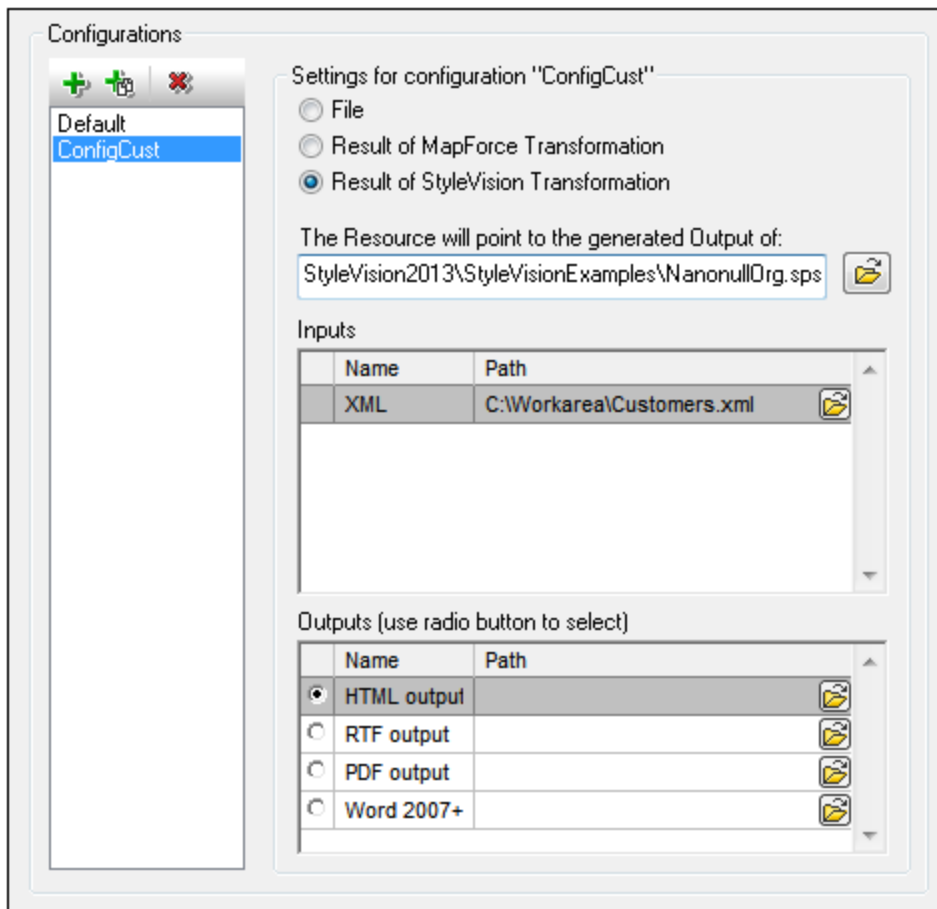
**Note:** Since MapForce is used to run the transformation, you must have Altova MapForce installed for this functionality to work.



## 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.



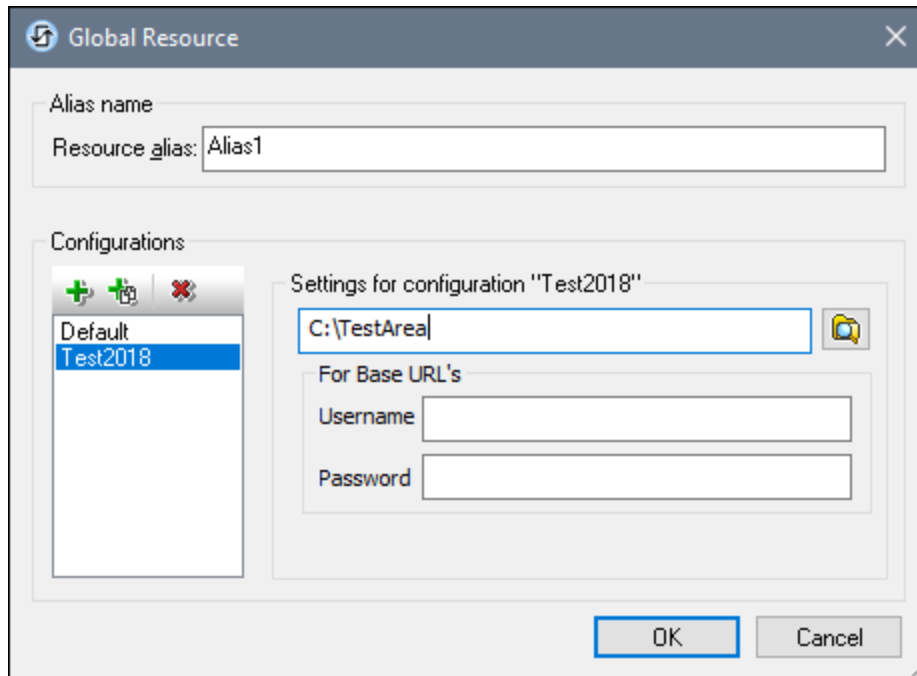
- **A .sps (SPS) file.** You must specify this file in the *Resource will point to generated output of* text box (see *screenshot above*).
- **Input file/s.** The input file might already be specified in the SPS file. If it is, it will appear automatically in the *Inputs* pane once the SPS file is selected. 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.

## 8.1.2 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

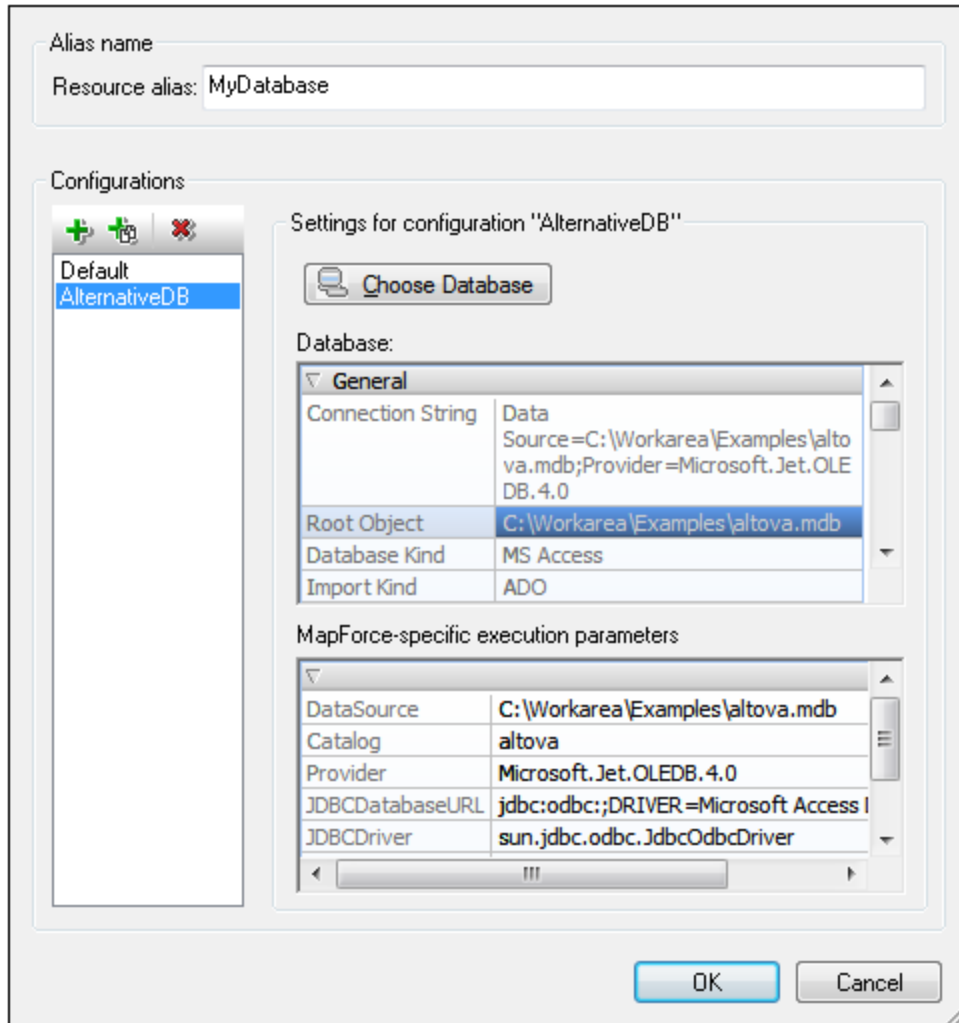
Define the alias (its name and configurations) as follows:

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](#)<sup>91</sup>.

### 8.1.3 Databases

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



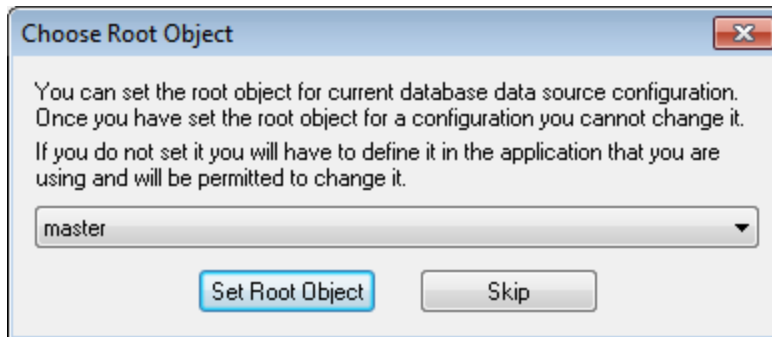
#### Global Resource dialog icons

-  **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.

## 8.2 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](#)<sup>102</sup>.

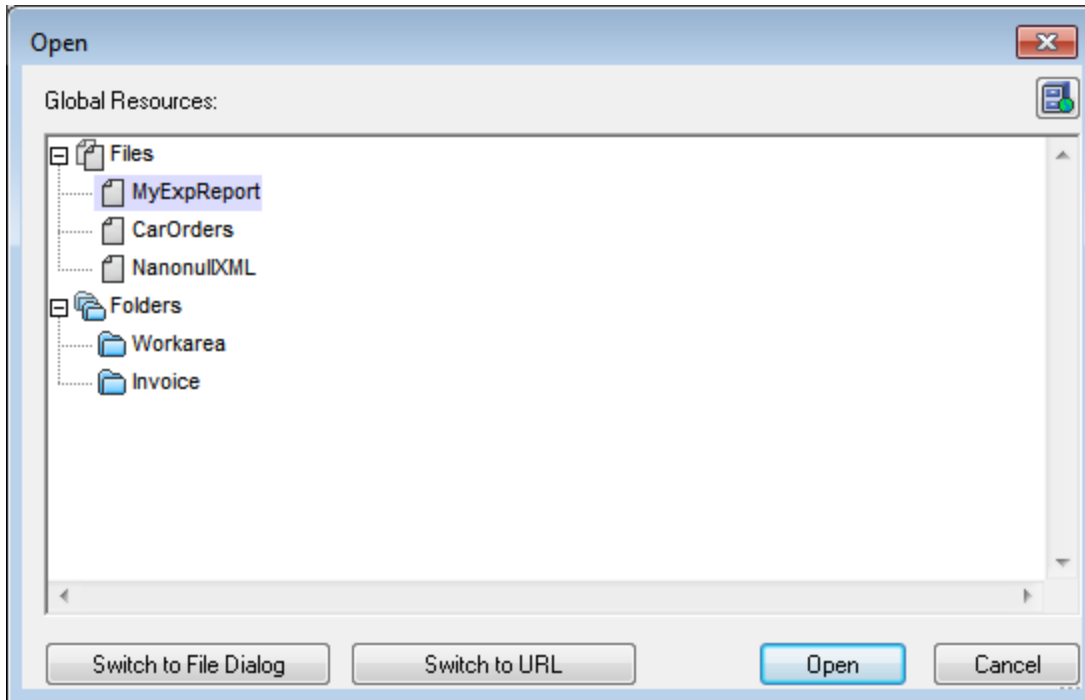
### Selections that determine which resource is used

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](#)<sup>91</sup>. 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](#)<sup>236</sup> 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.

### 8.2.1 Assigning Files and Folders

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



*Manage Global Resources*: Displays the [Manage Global Resources](#)<sup>91</sup> 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](#)<sup>91</sup> dialog.

## Usage scenarios

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

- [Opening global resources](#)<sup>103</sup>
- [Saving as global resource](#)<sup>104</sup>
- [XSLT transformation](#)<sup>104</sup>

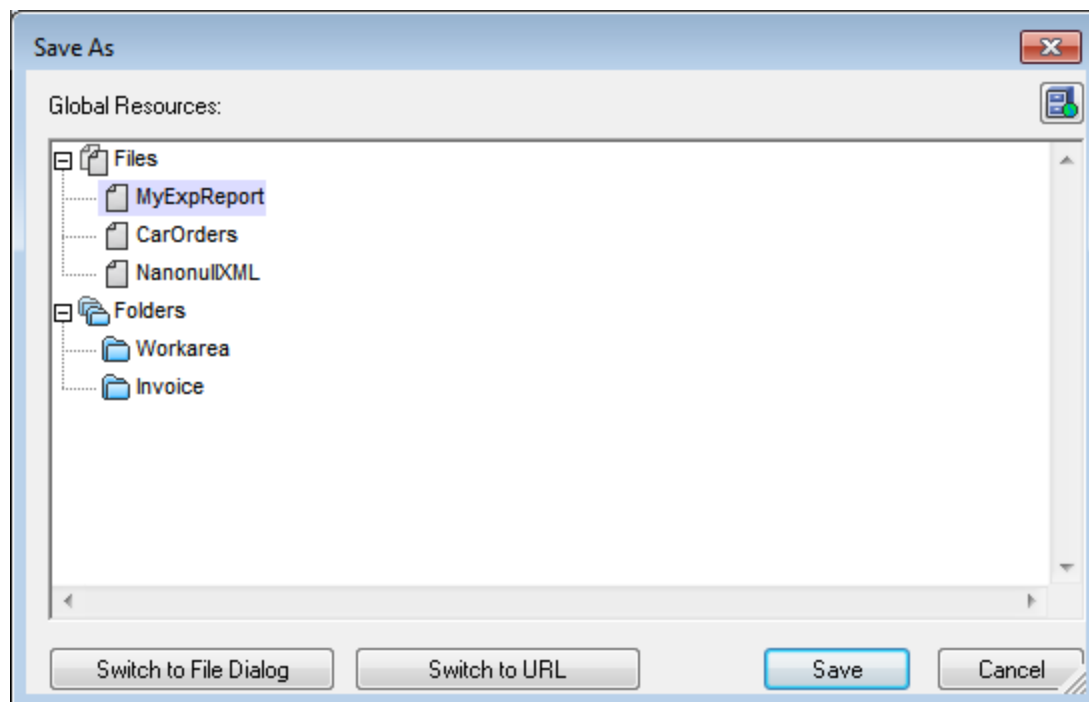
## Opening global resources

A global resource can be opened in Authentic Desktop with the [File | Open \(Switch to Global Resource\)](#)<sup>161</sup> 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 **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](#)<sup>93</sup>, the file will be saved directly. If the alias is a [folder alias](#)<sup>98</sup>, 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](#)<sup>105</sup>.

**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](#)<sup>210</sup> or [XSL/XQuery | XSL:FO Transformation](#)<sup>211</sup> pops up a dialog in which you can browse for the required XSLT or XML file. Click the **Browse** button and then the **Global Resource** button to pop up the Open Global Resource dialog (*screenshot at top of section*<sup>102</sup>). The file that is associated with the currently active configuration of the selected global resource is used for the transformation.



## 8.2.2 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.



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

## 9 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.

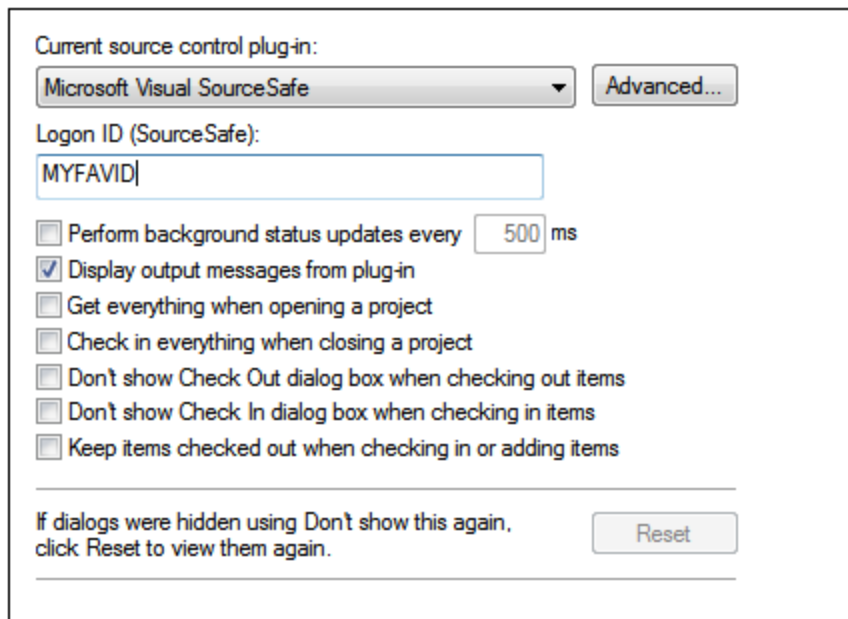
You can use as your source control provider any commercial or non-commercial plug-in that supports the Microsoft Source Control Plug-in API, and can connect to a compatible version control system. For the list of source control systems and plug-ins tested by Altova, see [Supported Source Control Systems](#)<sup>109</sup>.

### 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>.

## 9.1 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](#)<sup>109</sup>) 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](#)<sup>112</sup>, 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 section](#)<sup>182</sup> of the Menu 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](#)<sup>266</sup>), or (ii) in the Change Source Control dialog (**Project | Source Control | Change Source Control**).

## 9.2 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.

Source Control System	Source Code Control Clients
AccuRev 4.7.0 Windows	AccuBridge for Microsoft SCC 2008.2
Bazaar 1.9 Windows	Aigenta Unified SCC 1.0.6
Borland StarTeam 2008	Borland StarTeam Cross-Platform Client 2008 R2
Codice Software Plastic SCM Professional 2.7.127.10 (Server)	Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)
Collabnet Subversion 1.5.4	<ul style="list-style-type: none"> <li>• Aigenta Unified SCC 1.0.6</li> <li>• PushOK SVN SCC 1.5.1.1</li> <li>• PushOK SVN SCC x64 version 1.6.3.1</li> <li>• TamTam SVN SCC 1.2.24</li> </ul>
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 <a href="#">Source Control with Git</a> <sup>126</sup> )
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	<ul style="list-style-type: none"> <li>• Jalindi Igloo 1.0.3</li> <li>• March-Hare CVS Suite Client 2008 (3321)</li> <li>• PushOK CVS SCC NT 2.1.2.5</li> <li>• PushOK CVS SCC x64 version 2.2.0.4</li> <li>• TamTam CVS SCC 1.2.40</li> </ul>
Mercurial 1.0.2 for Windows	Sergey Antonov HgSCC 1.0.1
Microsoft SourceSafe 2005 with CTP	Microsoft SourceSafe 2005 with CTP

Source Control System	Source Code Control Clients
Microsoft Visual Studio Team System 2008/2010 Team Foundation Server	Microsoft Team Foundation Server 2008/2010 MSSCCI Provider
Perforce 2008 P4S 2008.1	Perforce P4V 2008.1
PureCM Server 2008/3a	PureCM Client 2008/3a
QSC Team Coherence Server 7.2.1.35	QSC Team Coherence Client 7.2.1.35
Reliable Software Code Co-Op 5.1a	Reliable Software Code Co-Op 5.1a
Seapine Surround SCM Client/Server for Windows 2009.0.0	Seapine Surround SCM Client 2009.0.0
Serena Dimensions Express/CM 10.1.3 for Win32 Server	Serena Dimensions 10.1.3 for Win32 Client
Softimage Alienbrain Server 8.1.0.7300	Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300
SourceGear Fortress 1.1.4 Server	SourceGear Fortress 1.1.4 Client
SourceGear SourceOffsite Server 4.2.0	SourceGear SourceOffsite Client 4.2.0 (Windows)
SourceGear Vault 4.1.4 Server	SourceGear Vault 4.1.4 Client
VisualSVN Server 1.6	<ul style="list-style-type: none"> <li>• Aigenta Unified SCC 1.0.6</li> <li>• PushOK SVN SCC 1.5.1.1</li> <li>• PushOK SVN SCC x64 version 1.6.3.1</li> <li>• TamTam SVN SCC 1.2.24</li> </ul>

## 9.3 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.

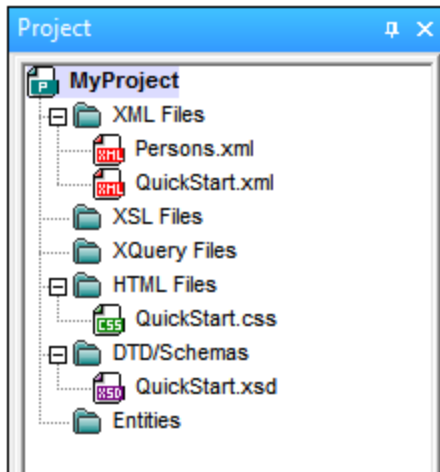
If the entire application project is placed under source control (by selecting the project name in the Projects window and placing it under source control), then the entire local folder structure is recreated in the repository.

**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.

## 9.4 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](#)<sup>(11)</sup> (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.

### Local Workspace Folder

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

### Repository

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



```
| | |-- Persons.xml          || |-- Persons
                             || | |-- Persons.xml
```

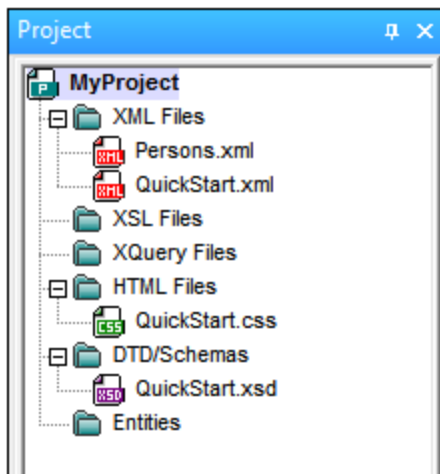
**Note:** An application project can contain project folders (green) and external folders (yellow). Only files in (green) project folders can be placed under source control. Files in (yellow) external folders cannot be placed under source control.

**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.

## 9.5 Add to Source Control

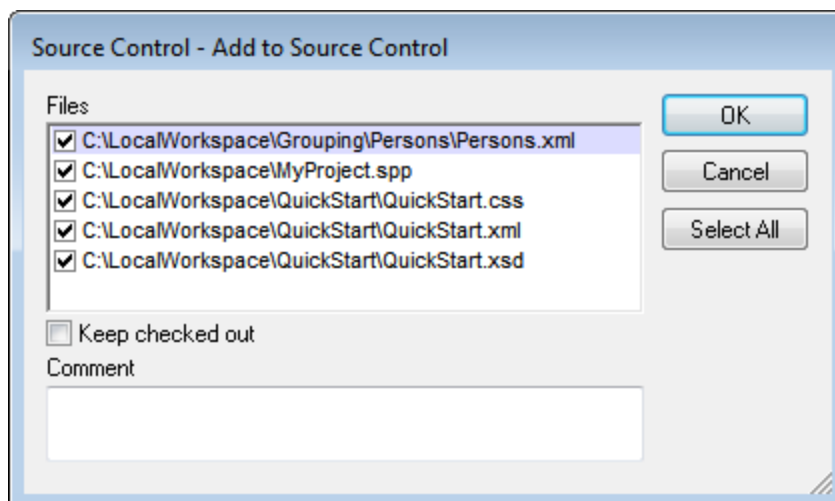
Adding the project to source control will automatically create the correct bindings and repository structure before adding the project file (.spp file) or individual files to source control. Add the project to source control as follows.

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**.



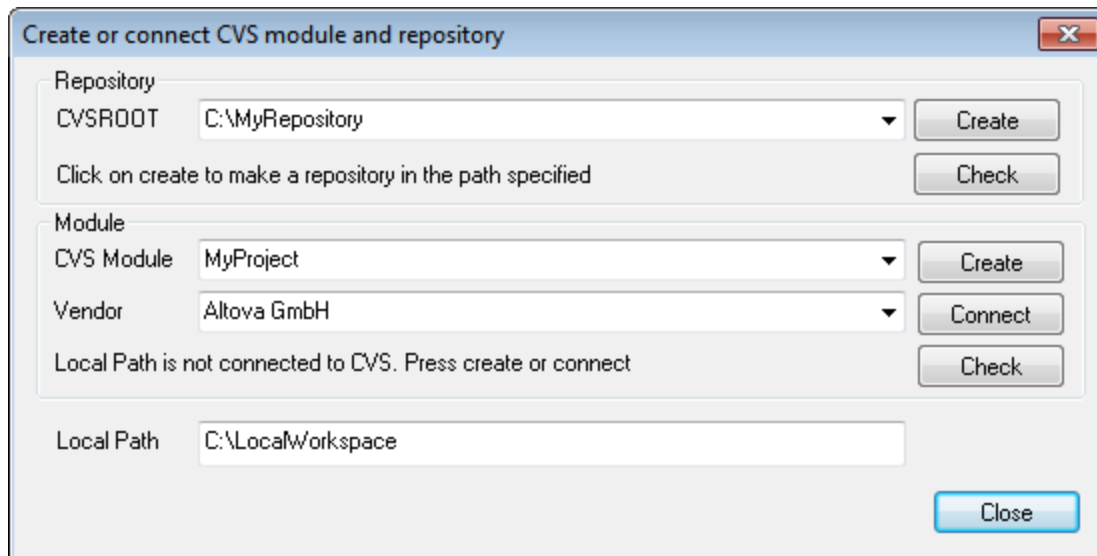
The files will be added to the repository and be either [checked in or checked out](#)<sup>117</sup> depending on whether the *Keep Checked Out* check box has been checked or not.

## 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](#)<sup>111</sup> will be bound to this folder created in the repository (*see diagrams below*).

<u>Local Workspace Folder</u>	<u>Repository</u>
-- MyProject.spp	-- <u>MyProject (bound to Local Workspace)</u>
-- QuickStart	-- MyProject.spp
-- QuickStart.css	-- QuickStart
-- QuickStart.xml	-- QuickStart.css
-- QuickStart.xsd	-- QuickStart.xml
-- Grouping	-- QuickStart.xsd
-- Persons	-- Grouping
-- Persons.xml	-- Persons
	-- Persons.xml

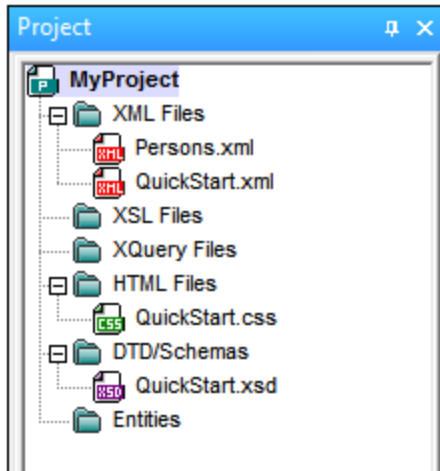
The configuration dialog of Jalindi Igloo is show below. The CVSROOT field is the path to the repository folder.



In the screenshot above, the local path locates the local workspace folder, which corresponds to the CVS module, `MyProject`, and is bound to it.

## 9.6 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:

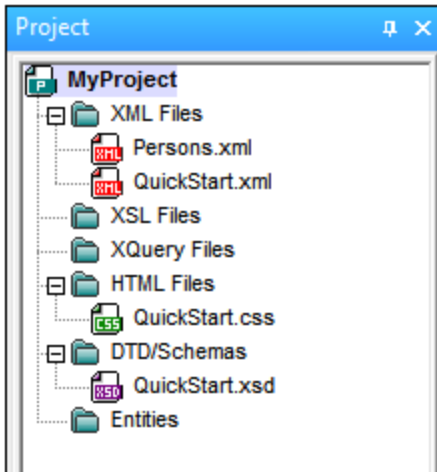
- [Add to, Remove from Source Control](#)<sup>116</sup>
- [Check Out, Check In](#)<sup>117</sup>
- [Getting Files as Read-Only](#)<sup>119</sup>
- [Copying and Sharing from Source Control](#)<sup>121</sup>
- [Changing Source Control](#)<sup>124</sup>

Additional commands in the **Project | Source Control** menu are described in the [Menu Reference section](#)<sup>182</sup> of the manual. For information specific to a particular source control system, please see the user documentation of that system.

### 9.6.1 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](#)<sup>112</sup> (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](#)<sup>114</sup>.

## Source control symbols

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

## 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.




## 9.6.2 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**.

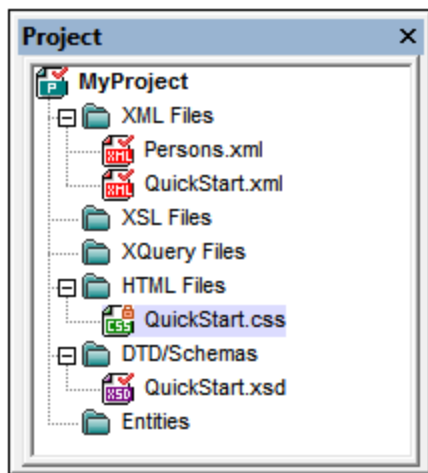
When a file is checked out, a copy from the repository is placed in the local folder. A file that is checked out can be edited. If a file that is under source control is not checked out, it cannot be edited. After a file has been edited, the changes can be saved to the repository by checking in the file. 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.

	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.

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.



[Getting Files as Read-Only](#) <sup>119</sup>

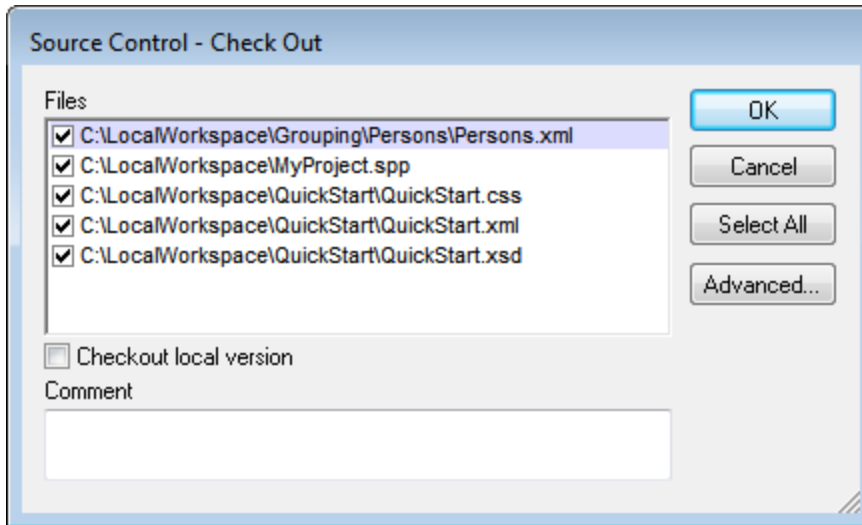
## Saving and rejecting editing changes

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**.

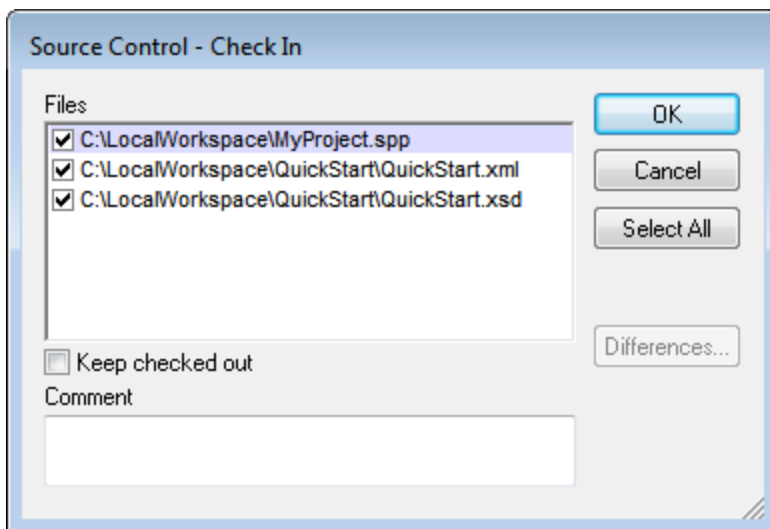
## Checking out

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.

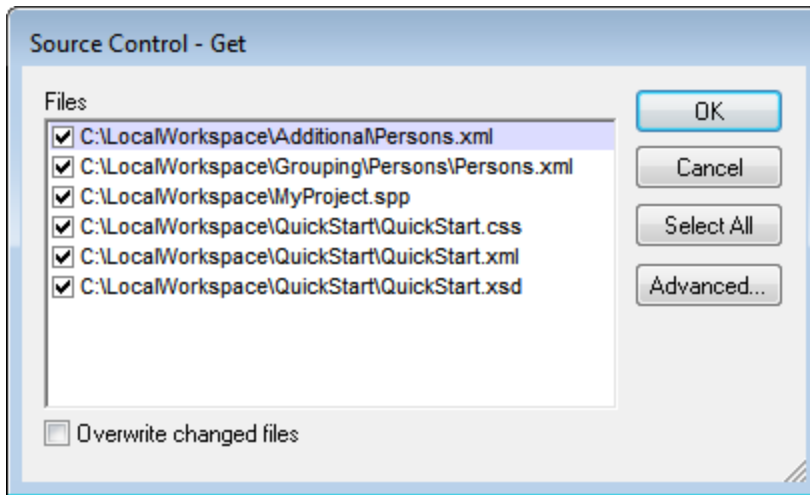


**Note:** In both dialogs (Check Out and Check In), multiple files appear if the selected object (project or project folder/s) contain multiple files.

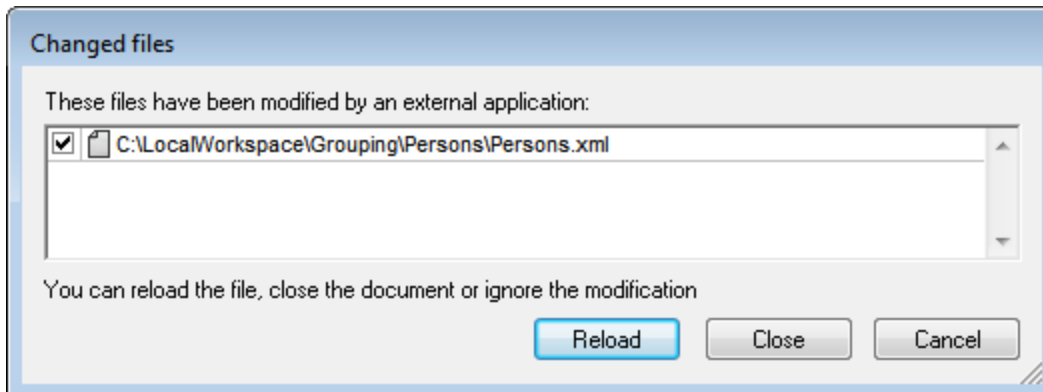
### 9.6.3 Getting Files as Read-Only

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](#) <sup>117</sup>.) 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, .

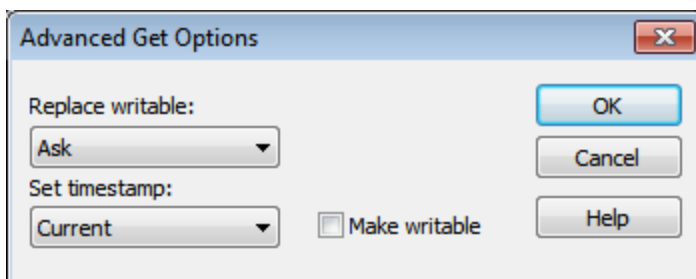


You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking **OK**, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (*screenshot below*) asking whether you wish to reload the file/s (**Reload**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

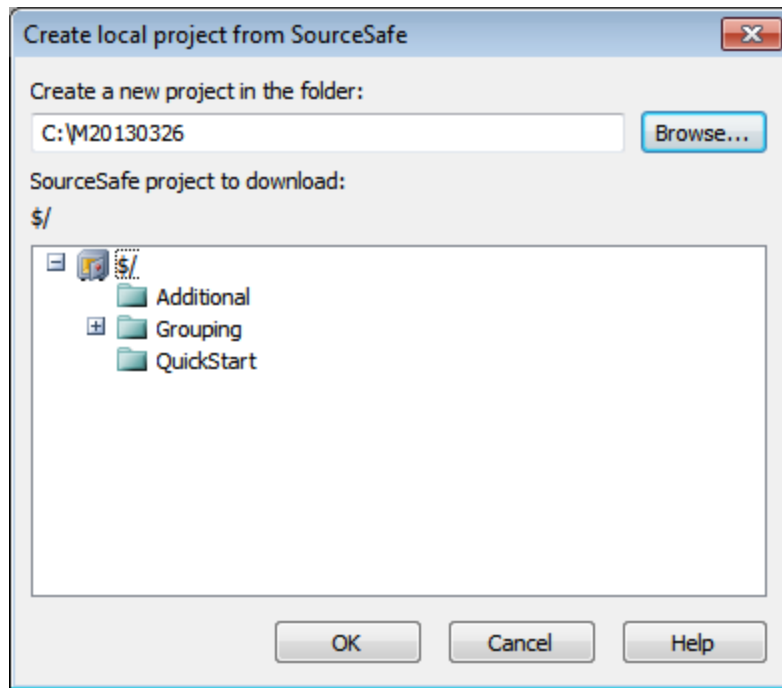
**Note:** This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.

## 9.6.4 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](#)<sup>112</sup> 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`.

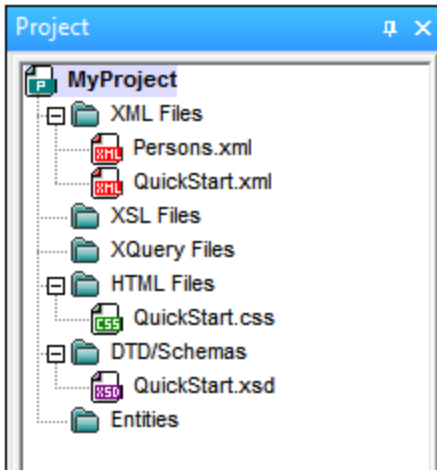


6. Click **OK**. The contents of the bound folder (`MyProject`) will be copied to the local folder `C:\M20130326.`, and a dialog pops up asking you to select the project file (`.spp` file) that is to be created as the new project.
7. Select the `.spp` file that will have been copied to the local folder. 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.`

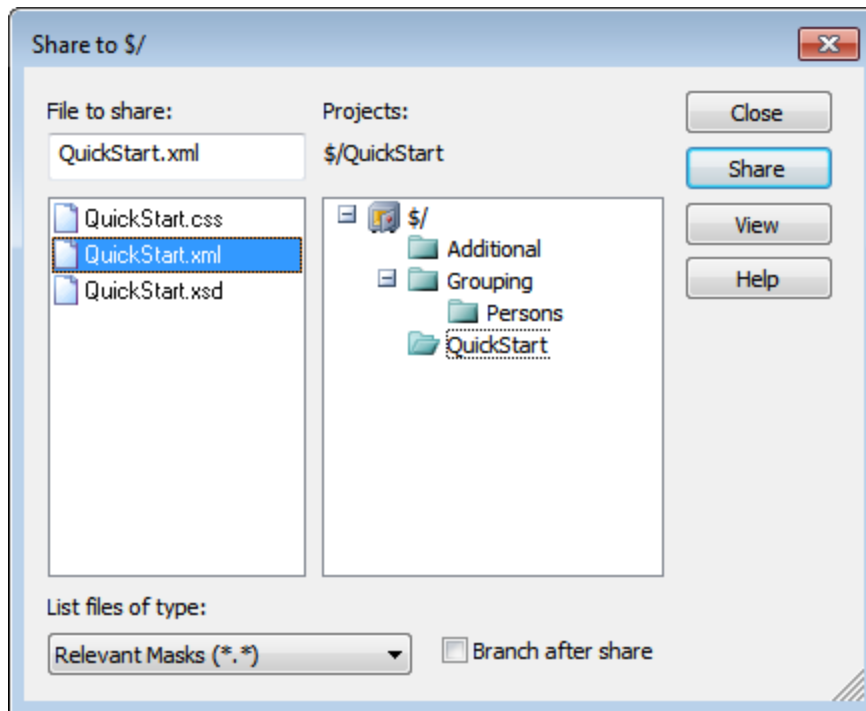
## Sharing from source control

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**.

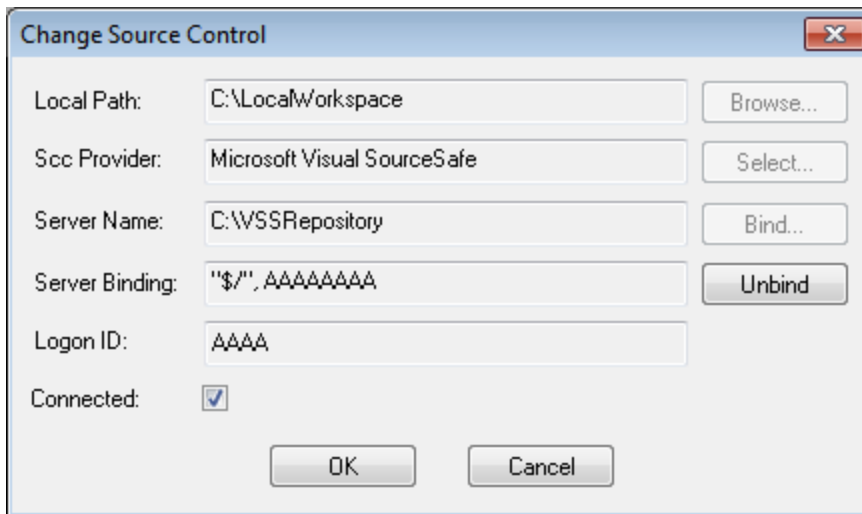


The Share To [Folder] dialog (screenshot below) pops up.



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.

## 9.6.5 Changing Source Control

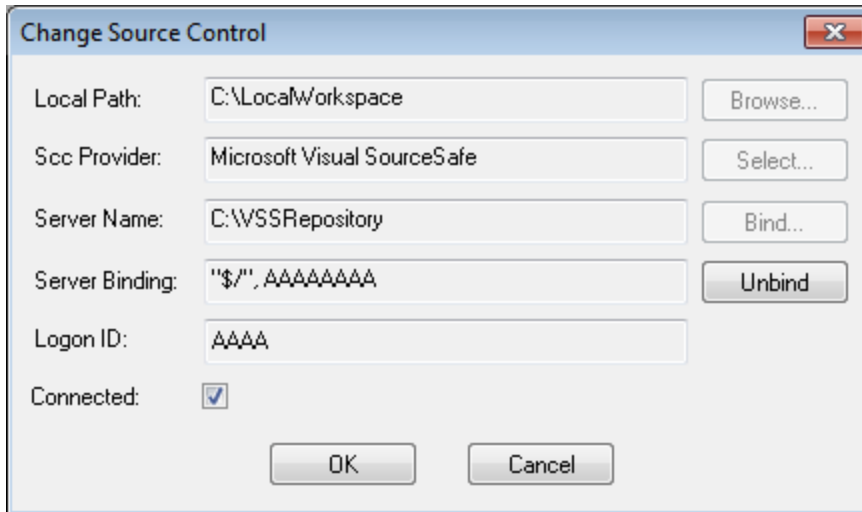
Source control settings can be changed via two commands in the **Project | Source Control** menu:

- **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 **Unbind** button. All the settings are now editable.



Change source control settings as follows:

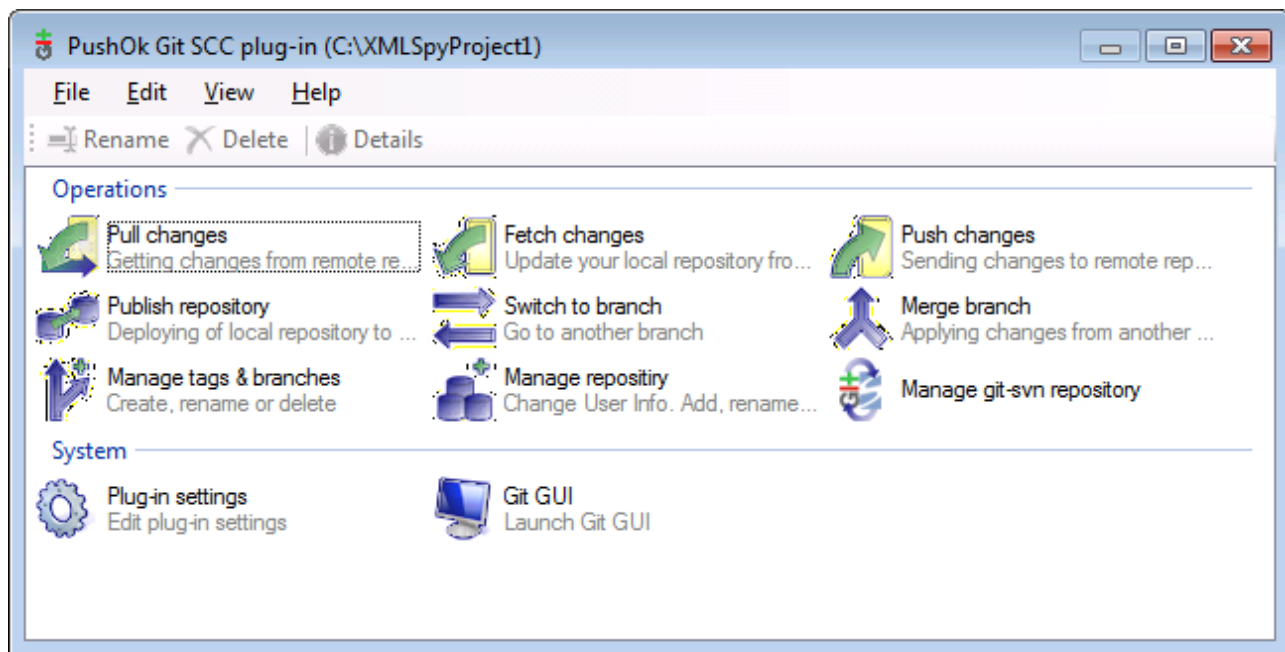
1. Use the **Browse** button to browse for the local folder and the **Select** button to select from among the installed source control systems.
2. After doing this you can bind the local folder to a repository database. Click the **Bind** button to do this. This pops up the connection dialog of your 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.

## 9.7 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>).

At the time when this documentation is written, the **GIT SCC 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 mediates 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.

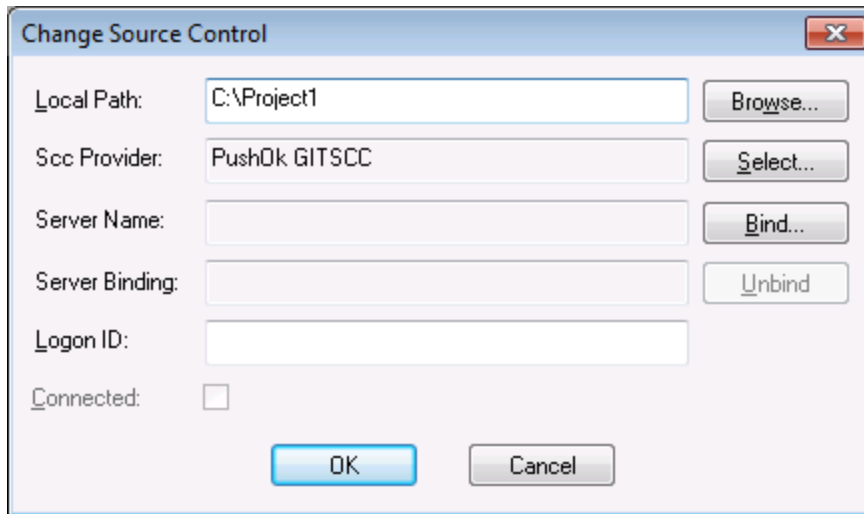
The following sections describe the initial configuration of the plug-in, as well as the basic workflow:

- [Enabling Git Source Control with GIT SCC Plug-in](#) <sup>(127)</sup>
- [Adding a Project to Git Source Control](#) <sup>(127)</sup>
- [Cloning a Project from Git Source Control](#) <sup>(129)</sup>

## 9.7.1 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>), 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.



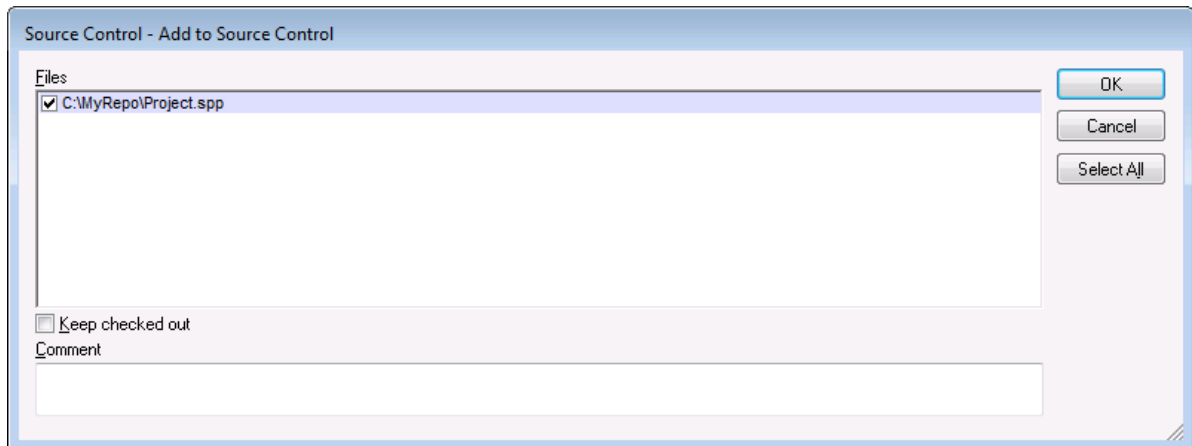
3. When a dialog box prompts you to register the plug-in, click **Registration** and follow the wizard steps to complete the registration process.

## 9.7.2 Adding a Project to Git Source Control

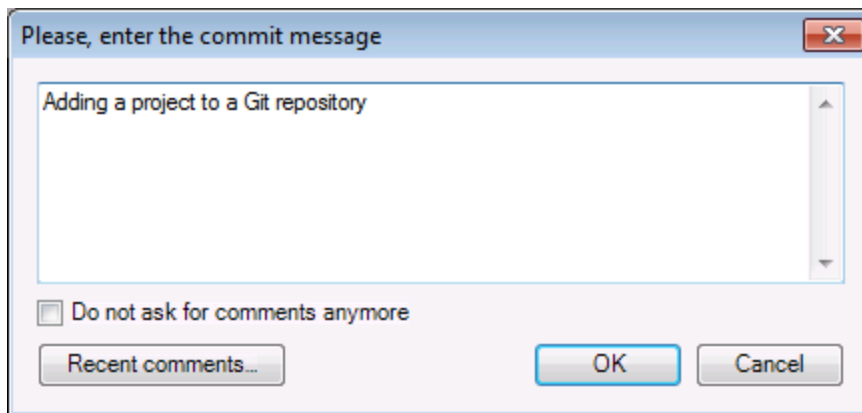
You can save Authentic Desktop projects as Git repositories. The structure of files or folders that you add to the project would then correspond to the structure of the Git repository.

**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](#)<sup>127</sup>).
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**.

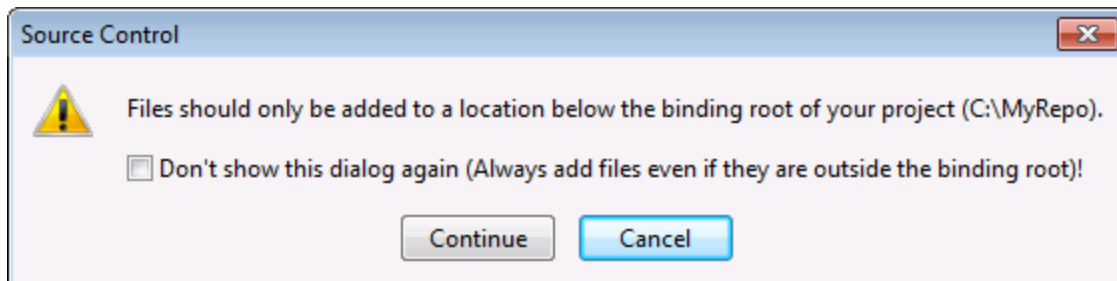


5. Click **OK**.



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:

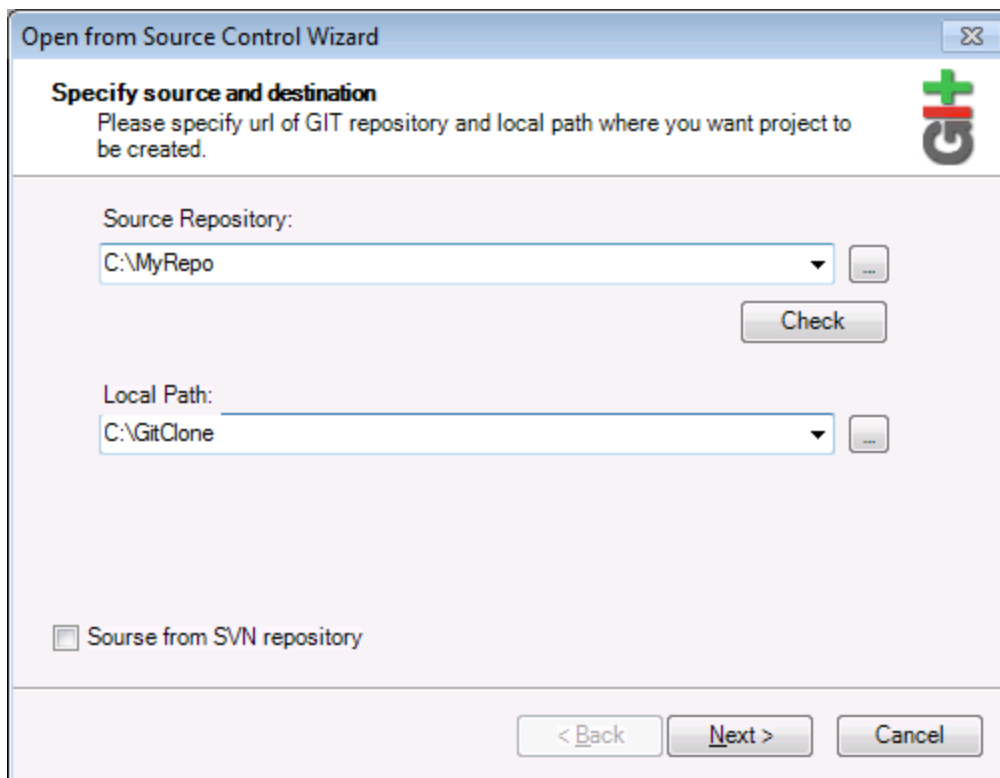




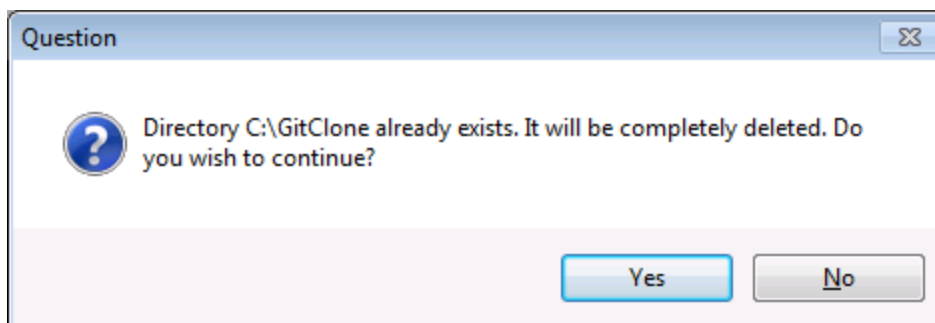
### 9.7.3 Cloning a Project from Git Source Control

Projects that have been previously added to Git source control (see [Adding a Project to Git Source Control](#)<sup>127</sup>) can be opened from the Git repository as follows:

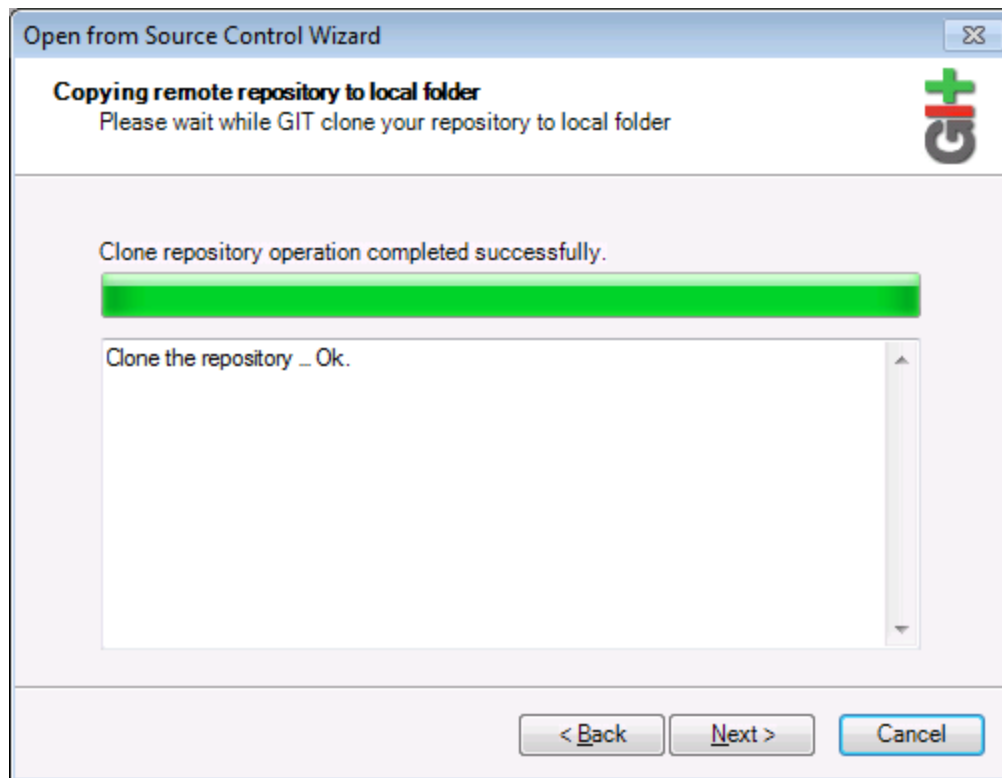
1. Make sure that **PushOK GIT SCC Plug-in** is set as source control provider (see [Enabling Git Source Control with GIT SCC Plug-in](#)<sup>127</sup>).
2. On the **Project** menu, click **Source Control | Open from Source Control**.
3. Enter the path or the URL of the source repository. Click **Check** 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:



5. Click **Yes** to confirm, and then click **Next**.

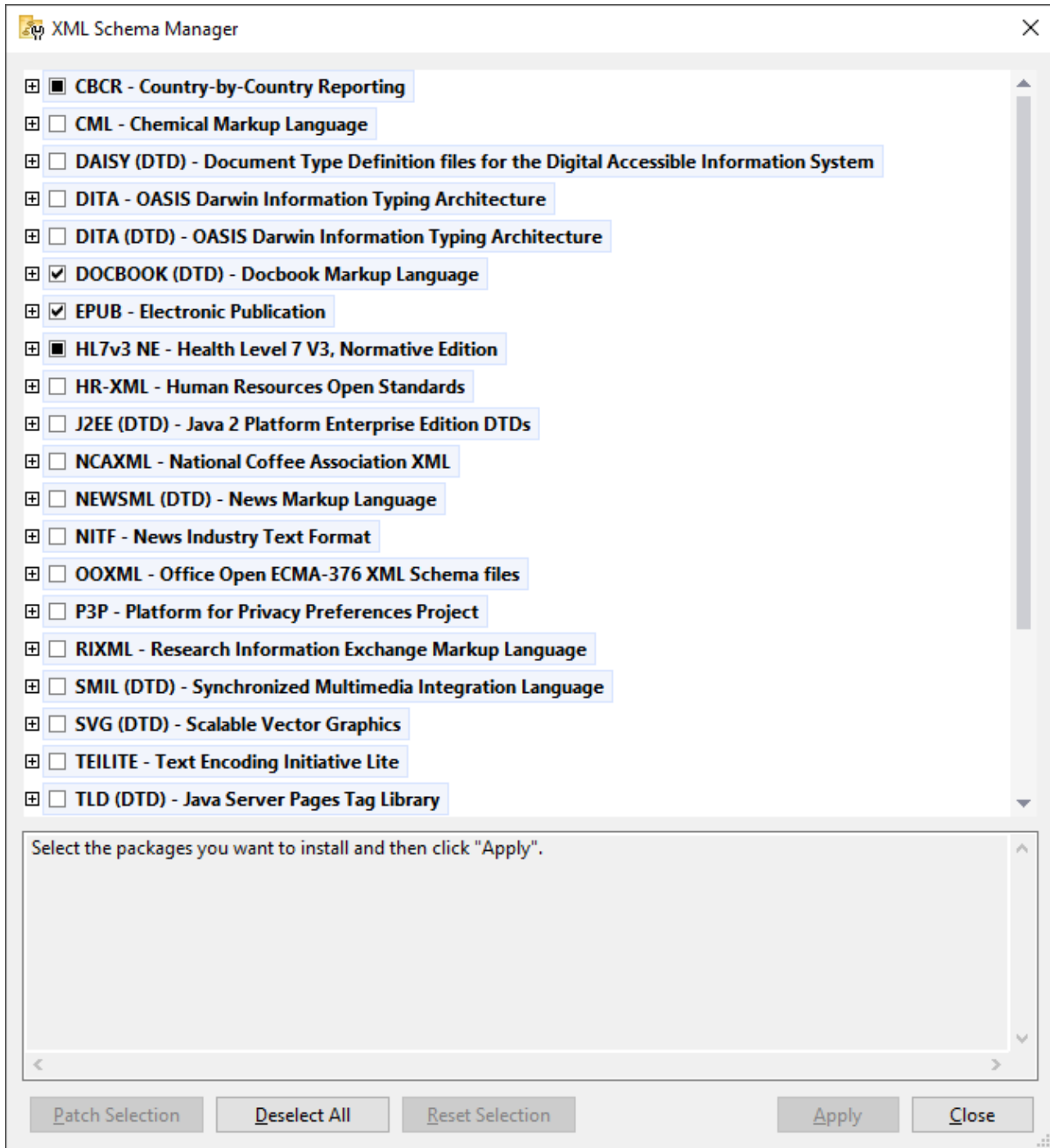


6. Follow the remaining wizard steps, as required by your specific case.
7. When the wizard completes, a Browse dialog box appears, asking you to open the Authentic Desktop Project (\*.spp) file. Select the project file to load the project contents into Authentic Desktop.

## 10 Schema Manager

XML Schema Manager is an Altova tool that provides a centralized way to install and manage XML schemas (DTDs for XML and XML Schemas) for use across all Altova's XML-Schema-aware applications, including Authentic Desktop.

- On Windows, Schema Manager has a graphical user interface (*screenshot below*) and is also available at the command line. (Altova's desktop applications are available on Windows only; *see list below*.)
- On Linux and macOS, Schema Manager is available at the command line only. (Altova's server applications are available on Windows, Linux, and macOS; *see list below*.)



*Altova applications that operate with Schema Manager*

Desktop applications (Windows only)	Server applications (Windows, Linux, macOS)
XMLSpy (all editions)	RaptorXML Server, RaptorXML+XBRL Server

MapForce (all editions)	StyleVision Server
StyleVision (all editions)	
Authentic Desktop Enterprise Edition	

## Installation and de-installation of Schema Manager

Schema Manager is installed automatically when you first install a new version of Altova Mission Kit or of any of Altova's XML-schema-aware applications (see *table above*).

Likewise, it is removed automatically when you uninstall the last Altova XML-schema-aware application from your computer.

## Schema Manager features

Schema Manager provides the following features:

- Shows XML schemas installed on your computer and checks whether new versions are available for download.
- Downloads newer versions of XML schemas independently of the Altova product release cycle. (Altova stores schemas online, and you can download them via Schema Manager.)
- Install or uninstall any of the multiple versions of a given schema (or all versions if necessary).
- An XML schema may have dependencies on other schemas. When you install or uninstall a particular schema, Schema Manager informs you about dependent schemas and will automatically install or remove them as well.
- Schema Manager uses the [XML catalog](#) mechanism to map schema references to local files. In the case of large XML schemas, processing will therefore be faster than if the schemas were at a remote location.
- All major schemas are available via Schema Manager and are regularly updated for the latest versions. This provides you with a convenient single resource for managing all your schemas and making them readily available to all of Altova's XML-schema-aware applications.
- Changes made in Schema Manager take effect for all Altova products installed on that machine.
- In an Altova product, if you attempt to validate on a schema that is not installed but which is available via Schema Manager, then installation is triggered automatically. However, if the schema package contains namespace mappings, then there will be no automatic installation; in this case, you must start Schema Manager, select the package/s you want to install, and run the installation. If, after installation, your open Altova application does not restart automatically, then you must restart it manually.

## How it works

Altova stores all XML schemas used in Altova products online. This repository is updated when new versions of the schemas are released. Schema Manager displays information about the latest available schemas when invoked in both its GUI form as well as on the CLI. You can then install, upgrade or uninstall schemas via Schema Manager.

Schema Manager also installs schemas in one other way. At the Altova website (<https://www.altova.com/schema-manager>) you can select a schema and its dependent schemas that you want to install. The website will prepare a file of type `.altova_xmlschemas` for download that contains information about your schema selection. When you double-click this file or pass it to Schema Manager via the CLI as an argument of the `install` <sup>145</sup> command, Schema Manager will install the schemas you selected.

**Local cache: tracking your schemas**

All information about installed schemas is tracked in a centralized cache directory on your computer, located here:

<i>Windows</i>	C:\ProgramData\Altova\pkgs\.cache
<i>Linux</i>	/var/opt/Altova/pkgs\.cache
<i>macOS</i>	/var/Altova/pkgs

This cache directory is updated regularly with the latest status of schemas at Altova's online storage. These updates are carried out at the following times:

- Every time you start Schema Manager.
- When you start Authentic Desktop for the first time on a given calendar day.
- If Authentic Desktop is open for more than 24 hours, the cache is updated every 24 hours.
- You can also update the cache by running the [update](#)<sup>148</sup> command at the command line interface.

The cache therefore enables Schema Manager to continuously track your installed schemas against the schemas available online at the Altova website.

**Do not modify the cache manually!**

The local cache directory is maintained automatically based on the schemas you install and uninstall. It should not be altered or deleted manually. If you ever need to reset Schema Manager to its original "pristine" state, then, on the command line interface (CLI): (i) run the [reset](#)<sup>146</sup> command, and (ii) run the [initialize](#)<sup>144</sup> command. (Alternatively, run the `reset` command with the `--i` option.)

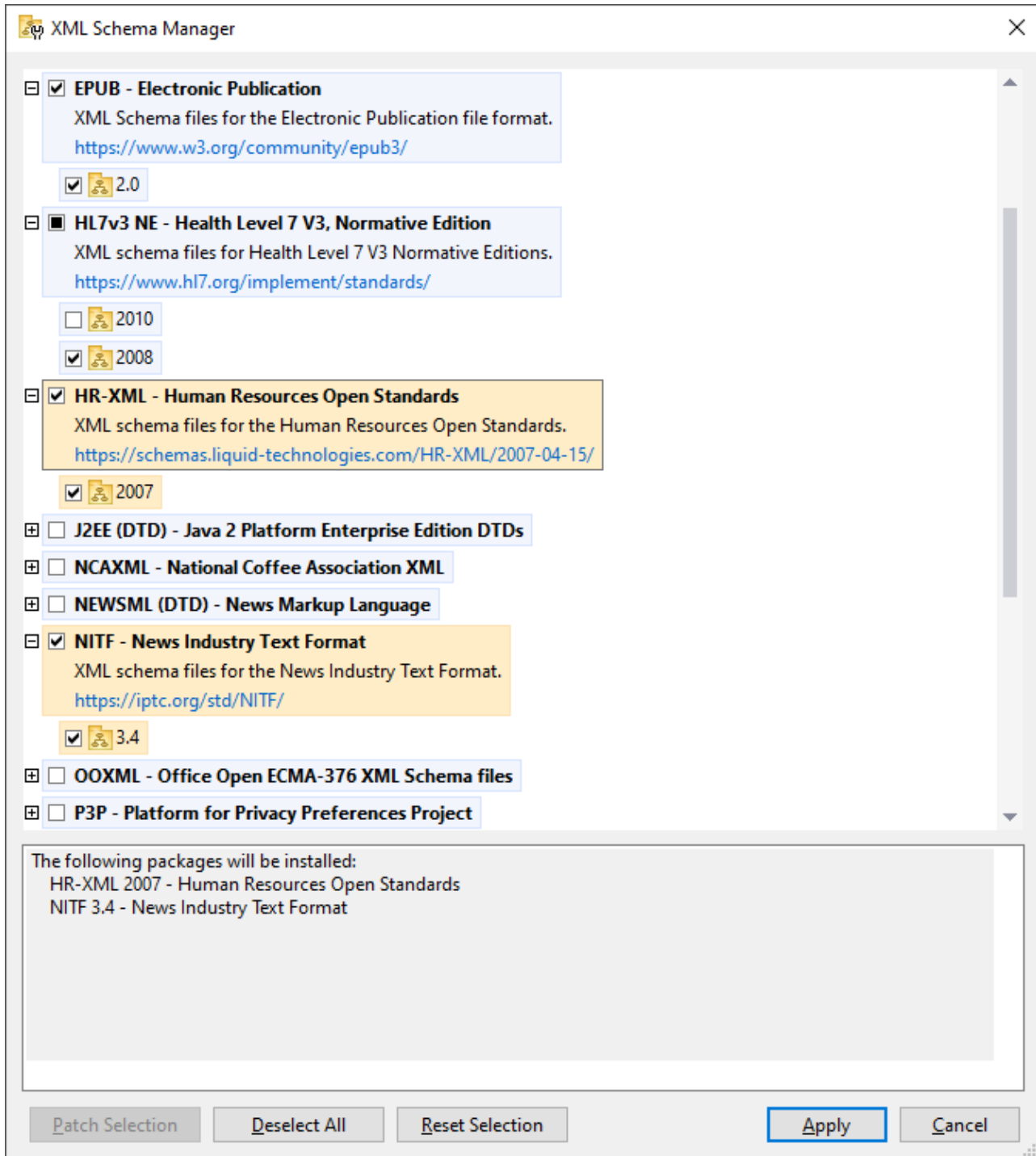
## 10.1 Run Schema Manager

### Graphical User Interface

You can access the GUI of Schema Manager in any of the following ways:

- *During the installation of Authentic Desktop:* Towards the end of the installation procedure, select the check box *Invoke Altova XML-Schema Manager* to access the Schema Manager GUI straight away. This will enable you to install schemas during the installation process of your Altova application.
- *After the installation of Authentic Desktop:* After your application has been installed, you can access the Schema Manager GUI at any time, via the menu command **Tools | XML Schema Manager**.
- Via the `.altova_xmlschemas` file downloaded from the [Altova website](#): Double-click the downloaded file to run the Schema Manager GUI, which will be set up to install the schemas you selected (at the website) for installation.

After the Schema Manager GUI (*screenshot below*) has been opened, already installed schemas will be shown selected. If you want to install an additional schema, select it. If you want to uninstall an already installed schema, deselect it. After you have made your selections and/or deselections, you are ready to apply your changes. The schemas that will be installed or uninstalled will be highlighted and a message about the upcoming changes will be posted to the Messages pane at the bottom of the Schema Manager window (see *screenshot*).



## Command line interface

You can run Schema Manager from a command line interface by sending commands to its executable file, `xmlschemamanager.exe`.



The `xmlschemamanager.exe` file is located in the following folder:

- *On Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`
- *On Linux or macOS (server applications only):* `%INSTALLDIR%/bin`, where `%INSTALLDIR%` is the program's installation directory.

You can then use any of the commands listed in the [CLI command reference section](#)<sup>143</sup>.

To display help for the commands, run the following:

- *On Windows:* `xmlschemamanager.exe --help`
- *On Linux or macOS (server applications only):* `sudo ./xmlschemamanager --help`

## 10.2 Status Categories

Schema Manager categorizes the schemas under its management as follows:

- *Installed schemas.* These are shown in the GUI with their check boxes selected (*in the screenshot below the checked and blue versions of the EPUB and HL7v3 NE schemas are installed schemas*). If all the versions of a schema are selected, then the selection mark is a tick. If at least one version is unselected, then the selection mark is a solid colored square. You can deselect an installed schema to **uninstall** it; (*in the screenshot below, the DocBook DTD is installed and has been deselected, thereby preparing it for de-installation*).
- *Uninstalled available schemas.* These are shown in the GUI with their check boxes unselected. You can select the schemas you want to **install**.



- *Upgradeable schemas* are those which have been revised by their issuers since they were installed. They are indicated in the GUI by a 🔄 icon. You can **patch** an installed schema with an available revision.

### Points to note

- In the screenshot above, both CBCR schemas are checked. The one with the blue background is already installed. The one with the yellow background is uninstalled and has been selected for installation. Note that the HL7v3 NE 2010 schema is not installed and has not been selected for installation.
- A yellow background means that the schema will be modified in some way when the **Apply** button is clicked. If a schema is unchecked and has a yellow background, it means that it will be uninstalled when the **Apply** button is clicked. In the screenshot above the DocBook DTD has such a status.

- When running Schema Manager from the command line, the [list](#)<sup>145</sup> command is used with different options to list different categories of schemas:

<code>xmlschemamanager.exe list</code>	Lists all installed and available schemas; upgradeables are also indicated
<code>xmlschemamanager.exe list -i</code>	Lists installed schemas only; upgradeables are also indicated
<code>xmlschemamanager.exe list -u</code>	Lists upgradeable schemas




**Note:** On Linux and macOS, use `sudo ./xmlschemamanager list`

## 10.3 Patch or Install a Schema

### Patch an installed schema

Occasionally, XML schemas may receive patches (upgrades or revisions) from their issuers. When Schema Manager detects that patches are available, these are indicated in the schema listings of Schema Manager and you can install the patches quickly.

#### *In the GUI*

Patches are indicated by the  icon. (Also see the previous topic about [status categories](#)<sup>138</sup>.) If patches are available, the **Patch Selection** button will be enabled. Click it to select and prepare all patches for installation. In the GUI, the icon of each schema that will be patched changes from  to , and the Messages pane at the bottom of the dialog lists the patches that will be applied. When you are ready to install the selected patches, click **Apply**. All patches will be applied together. Note that if you deselect a schema marked for patching, you will actually be uninstalling that schema.

#### *On the CLI*

To apply a patch at the command line interface:

1. Run the `list -u`<sup>145</sup> command. This lists any schemas for which upgrades are available.
2. Run the `upgrade`<sup>148</sup> command to install all the patches.

### Install an available schema

You can install schemas using either the Schema Manager GUI or by sending Schema Manager the install instructions via the command line.

**Note:** If the current schema references other schemas, the referenced schemas are also installed.

#### *In the GUI*

To install schemas using the Schema Manager GUI, select the schemas you want to install and click **Apply**.

You can also select the schemas you want to install at the [Altova website](#) and generate a downloadable `.altova_xmlschemas` file. When you double-click this file, it will open Schema Manager with the schemas you wanted pre-selected. All you will now have to do is click **Apply**.

#### *On the CLI*

To install schemas via the command line, run the `install`<sup>145</sup> command:

```
xmlschemamanager.exe install [options] Schema+
```

where `Schema` is the schema (or schemas) you want to install or a `.altova_xmlschemas` file. A schema is referenced by an identifier of format `<name>-<version>`. (The identifiers of schemas are displayed when you run the `list`<sup>145</sup> command.) You can enter as many schemas as you like. For details, see the description of the `install`<sup>145</sup> command.

**Note:** On Linux or macOS, use the `sudo ./xmlschemamanager` command.

***Installing a required schema***

When you run an XML-schema-related command in Authentic Desktop and Authentic Desktop discovers that a schema it needs for executing the command is not present or is incomplete, Schema Manager will display information about the missing schema/s. You can then directly install any missing schema via Schema Manager.

In the Schema Manager GUI, you can view all previously installed schemas at any time by running Schema Manager from **Tools | Schema Manager**.

## 10.4 Uninstall a Schema, Reset

### Uninstall a schema

You can uninstall schemas using either the Schema Manager GUI or by sending Schema Manager the uninstall instructions via the command line.

**Note:** If the schema you want to uninstall references other schemas, then the referenced schemas are also uninstalled.

#### *In the GUI*

To uninstall schemas in the Schema Manager GUI, clear their check boxes and click **Apply**. The selected schemas and their referenced schemas will be uninstalled.

To uninstall all schemas, click **Deselect All** and click **Apply**.

#### *On the CLI*

To uninstall schemas via the command line, run the [uninstall](#)<sup>147</sup> command:

```
xmlschemamanager.exe uninstall [options] Schema+
```

where each **schema** argument is a schema you want to uninstall or a `.altova_xmlschemas` file. A schema is specified by an identifier that has a format of `<name>-<version>`. (The identifiers of schemas are displayed when you run the [list](#)<sup>145</sup> command.) You can enter as many schemas as you like. For details, see the description of the [uninstall](#)<sup>147</sup> command.

**Note:** On Linux or macOS, use the `sudo ./xmlschemamanager` command.

### Reset Schema Manager

You can reset Schema Manager. This removes all installed schemas and the cache directory.

- In the GUI, click **Reset Selection**.
- On the CLI, run the [reset](#)<sup>146</sup> command.

After running this command, make sure to run the [initialize](#)<sup>144</sup> command in order to recreate the cache directory. Alternatively, run the [reset](#)<sup>146</sup> command with the `-i` option.

Note that [reset -i](#)<sup>146</sup> restores the original installation of the product, so it is recommended to run the [update](#)<sup>148</sup> command after performing a reset. Alternatively, run the [reset](#)<sup>146</sup> command with the `-i` and `-u` options.

## 10.5 Command Line Interface (CLI)

To call Schema Manager at the command line, you need to know the path of the executable. By default, the Schema Manager executable is installed here:

```
C:\ProgramData\Altova\SharedBetweenVersions\XMLSchemaManager.exe
```

**Note:** On Linux and macOS systems, once you have changed the directory to that containing the executable, you can call the executable with `sudo ./xmlschemamanager`. The prefix `./` indicates that the executable is in the current directory. The prefix `sudo` indicates that the command must be run with root privileges.

### Command line syntax

The general syntax for using the command line is as follows:

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

In the listing above, the vertical bar `|` separates a set of mutually exclusive items. The square brackets `[ ]` indicate optional items. Essentially, you can type the executable path followed by either `--h`, `--help`, or `--version` options, or by a command. Each command may have options and arguments. The list of commands is described in the following sections.

### 10.5.1 help

This command provides contextual help about commands pertaining to Schema Manager executable.

#### Syntax

```
<exec> help [command]
```

Where `[command]` is an optional argument which specifies any valid command name.

Note the following:

- You can invoke help for a command by typing the command followed by `-h` or `--help`, for example:  

```
<exec> list -h
```
- If you type `-h` or `--help` directly after the executable and before a command, you will get general help (not help for the command), for example: 

```
<exec> -h list
```

#### Example

The following command displays help about the `list` command:

```
xmlschemamanager help list
```

## 10.5.2 info

This command displays detailed information for each of the schemas supplied as a `Schema` argument. This information for each submitted schema includes the title, version, description, publisher, and any referenced schemas, as well as whether the schema has been installed or not.

### Syntax

```
<exec> info [options] Schema+
```

- The `Schema` argument is the name of a schema or a part of a schema's name. (To display a schema's package ID and detailed information about its installation status, you should use the [list](#)<sup>145</sup> command.)
- Use `<exec> info -h` to display help for the command.

### Example

The following command displays information about the latest `DocBook-DTD` and `NITF` schemas:

```
xmlschemamanager info doc nitf
```

## 10.5.3 initialize

This command initializes the Schema Manager environment. It creates a cache directory where information about all schemas is stored. Initialization is performed automatically the first time a schema-cognizant Altova application is installed. You would not need to run this command under normal circumstances, but you would typically need to run it after executing the `reset` command.

### Syntax

```
<exec> initialize | init [options]
```

#### Options

The `initialize` command takes the following options:

<code>--silent, --s</code>	Display only error messages. The default is <code>false</code> .
<code>--verbose, --v</code>	Display detailed information during execution. The default is <code>false</code> .
<code>--help, --h</code>	Display help for the command.

### Example

The following command initializes Schema Manager:

```
xmlschemamanager initialize
```



## 10.5.4 install

This command installs one or more schemas.

### Syntax

```
<exec> install [options] Schema+
```

To install multiple schemas, add the `schema` argument multiple times.

The `schema` argument is one of the following:

- A schema identifier (having a format of `<name>-<version>`, for example: `cbr-2.0`). To find out the schema identifiers of the schemas you want, run the [list](#)<sup>145</sup> command. You can also use an abbreviated identifier if it is unique, for example `docbook`. If you use an abbreviated identifier, then the latest version of that schema will be installed.
- The path to a `.altova_xmlschemas` file downloaded from the Altova website. For information about these files, see [Introduction to SchemaManager: How It Works](#)<sup>131</sup>.

### Options

The `install` command takes the following options:

<code>--silent, --s</code>	Display only error messages. The default is <code>false</code> .
<code>--verbose, --v</code>	Display detailed information during execution. The default is <code>false</code> .
<code>--help, --h</code>	Display help for the command.

### Example

The following command installs the CBCR 2.0 (Country-By-Country Reporting) schema and the latest DocBook DTD:

```
xmlschemamanager install cbr-2.0 docbook
```

## 10.5.5 list

This command lists schemas under the management of Schema Manager. The list displays one of the following

- All available schemas
- Schemas containing in their name the string submitted as a `schema` argument
- Only installed schemas
- Only schemas that can be upgraded

### Syntax

```
<exec> list | ls [options] Schema?
```

If no `schema` argument is submitted, then all available schemas are listed. Otherwise, schemas are listed as specified by the submitted options (see *example below*). Note that you can submit the `schema` argument multiple times.

### Options

The `list` command takes the following options:

<code>--installed, --i</code>	List only installed schemas. The default is <code>false</code> .
<code>--upgradeable, --u</code>	List only schemas where upgrades (patches) are available. The default is <code>false</code> .
<code>--help, --h</code>	Display help for the command.

## Examples

- To list all available schemas, run: `xmlschemamanager list`
- To list installed schemas only, run: `xmlschemamanager list -i`
- To list schemas that contain either "doc" or "nitf" in their name, run: `xmlschemamanager list doc nitf`

## 10.5.6 reset

This command removes all installed schemas and the cache directory. You will be completely resetting your schema environment. After running this command, be sure to run the [initialize](#)<sup>144</sup> command to recreate the cache directory. Alternatively, run the `reset` command with the `-i` option. Since `reset -i` restores the original installation of the product, we recommend that you run the [update](#)<sup>148</sup> command after performing a reset and initialization. Alternatively, run the `reset` command with both the `-i` and `-u` options.

### Syntax

```
<exec> reset [options]
```

### Options

The `reset` command takes the following options:

<code>--init, --i</code>	Initialize Schema Manager after reset. The default is <code>false</code> .
<code>--update, --u</code>	Updates the list of available schemas in the cache. The default is <code>false</code> .
<code>--silent, --s</code>	Display only error messages. The default is <code>false</code> .
<code>--verbose, --v</code>	Display detailed information during execution. The default is <code>false</code> .
<code>--help, --h</code>	Display help for the command.

## Examples

- To reset Schema Manager, run: `xmlschemamanager reset`
- To reset Schema Manager and initialize it, run: `xmlschemamanager reset -i`
- To reset Schema Manager, initialize it, and update its schema list, run: `xmlschemamanager reset -i -u`

## 10.5.7 uninstall

This command uninstalls one or more schemas. By default, any schemas referenced by the current one are uninstalled as well. To uninstall just the current schema and keep the referenced schemas, set the option `--k`.

### Syntax

```
<exec> uninstall [options] Schema+
```

To uninstall multiple schemas, add the `schema` argument multiple times.

The `schema` argument is one of the following:

- A schema identifier (having a format of `<name>-<version>`, for example: `cbcr-2.0`). To find out the schema identifiers of the schemas that are installed, run the `list -i` <sup>145</sup> command. You can also use an abbreviated schema name if it is unique, for example `docbook`. If you use an abbreviated name, then all schemas that contain the abbreviation in its name will be uninstalled.
- The path to a `.altova_xmlschemas` file downloaded from the Altova website. For information about these files, see [Introduction to SchemaManager: How It Works](#) <sup>131</sup>.

### Options

The `uninstall` command takes the following options:

<code>--keep-references, --k</code>	Set this option to keep referenced schemas. The default is <code>false</code> .
<code>--silent, --s</code>	Display only error messages. The default is <code>false</code> .
<code>--verbose, --v</code>	Display detailed information during execution. The default is <code>false</code> .
<code>--help, --h</code>	Display help for the command.

### Example

The following command uninstalls the CBCR 2.0 and EPUB 2.0 schemas and their dependencies:

```
xmlschemamanager uninstall cbcr-2.0 epub-2.0
```

The following command uninstalls the `eba-2.10` schema but not the schemas it references:

```
xmlschemamanager uninstall --k cbcr-2.0
```

## 10.5.8 update

This command queries the list of schemas available from the online storage and updates the local cache directory. You should not need to run this command unless you have performed a [reset](#)<sup>146</sup> and [initialize](#)<sup>144</sup>.

### Syntax

```
<exec> update [options]
```

### Options

The **update** command takes the following options:

<code>--silent, --s</code>	Display only error messages. The default is <b>false</b> .
<code>--verbose, --v</code>	Display detailed information during execution. The default is <b>false</b> .
<code>--help, --h</code>	Display help for the command.

### Example

The following command updates the local cache with the list of latest schemas:

```
xmlschemamanager update
```

## 10.5.9 upgrade

This command upgrades all installed schemas that can be upgraded to the latest available *patched* version. You can identify upgradeable schemas by running the [list -u](#)<sup>145</sup> command.

**Note:** The **upgrade** command removes a deprecated schema if no newer version is available.

### Syntax

```
<exec> upgrade [options]
```

### Options

The **upgrade** command takes the following options:

<code>--silent, --s</code>	Display only error messages. The default is <b>false</b> .
<code>--verbose, --v</code>	Display detailed information during execution. The default is <b>false</b> .
<code>--help, --h</code>	Display help for the command.

## 11 Authentic Desktop in Visual Studio

Authentic Desktop can be integrated into the Microsoft Visual Studio IDE versions 2012/2013/2015/2017/2019/2022. 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](#)<sup>150</sup> and the integration of the Authentic Desktop plugin in Visual Studio.
- [Differences](#)<sup>151</sup> between the Visual Studio version and the standalone version.

## 11.1 Installing the Authentic Desktop Plugin for Visual Studio

To install the Authentic Desktop Plug-in for Visual Studio, take the steps below:

1. Install Microsoft Visual Studio 2012/2013/2015/2017/2019/2022. Note that from Visual Studio 2022 onwards, Visual Studio is being made available only as a 64-bit application.
2. Install Authentic Desktop. If you have installed Visual Studio 2022+, then you must install the 64-bit version of Authentic Desktop.
3. Download and run the Authentic Desktop integration package for Microsoft Visual Studio. This package is available on the Authentic Desktop download page at [www.altova.com](http://www.altova.com).

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

### Important

You must use the integration package corresponding to your Authentic Desktop version (current version is 2024).

## 11.2 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, [GUI and Environment](#)<sup>13</sup>.)

- 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 **Authentic Tool Windows** submenu contains options to toggle on entry helper windows and other sidebars, and to switch between the editing views, and toggle certain editing guides on and off.
- **Authentic Help:** This Authentic Desktop menu appears as a submenu in Visual Studio's **Help** menu.

**Note:** In Visual Studio 2019 and later, Authentic Desktop functionality can be accessed in the **Extensions** menu of Visual Studio. In earlier versions of Visual Studio, Authentic Desktop features are available in top-level menus of Visual Studio.

**Note:** Toolbar commands are not supported. If you have set up a toolbar command in Authentic Desktop that runs a command or script, then this toolbar command will not be available in the plug-in.

### 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, choose whether you want to open an Authentic global resource or an Authentic file via a URL.

## 12 Authentic Desktop in Eclipse

Eclipse is an open source framework that integrates different types of applications delivered in the form of plugins. The Authentic Desktop Integration Package for Eclipse enables you to integrate and access the functionality of Authentic Desktop in the Eclipse Platform for Windows. Supported Eclipse versions are: 2024-03 (4.31), 2023-12 (4.30), 2023-09 (4.29), 2023-06 (4.28).

In this section, we describe the following:

- [How to install the Integration Package for Eclipse and integrate Authentic Desktop in Eclipse](#) <sup>153</sup>
- [Authentic Desktop Perspective in Eclipse](#) <sup>155</sup>
- [Other Authentic Desktop Entry Points in Eclipse](#) <sup>158</sup>



## 12.1 Install the Integration Package for Eclipse

### Prerequisites

- Eclipse 2024-03 (4.31), 2023-12 (4.30), 2023-09 (4.29), 2023-06 (4.28) (<http://www.eclipse.org>), 64-bit.
- A Java Runtime Environment (JRE) or Java Development Kit (JDK) for the 64-bit platform.
- Authentic Desktop 64-bit.

**Note:** All the prerequisites listed above must have the 64-bit platform. Integration with older Eclipse 32-bit platforms is no longer supported, although it may still work.

After the prerequisites listed above are in place, you can install the Authentic Desktop Integration Package (64-bit) to integrate Authentic Desktop in Eclipse. The integration can be carried out either during the installation of the Integration Package or manually from Eclipse after the Integration Package has been installed. The Authentic Desktop Integration Package is available for download at <https://www.altova.com/components/download>.

**Note:** Eclipse must be closed while you install or uninstall the Authentic Desktop Integration Package.

### Integrate Authentic Desktop during installation of the Integration Package

You can integrate Authentic Desktop in Eclipse during the installation of the Authentic Desktop Integration Package. Do this as follows:

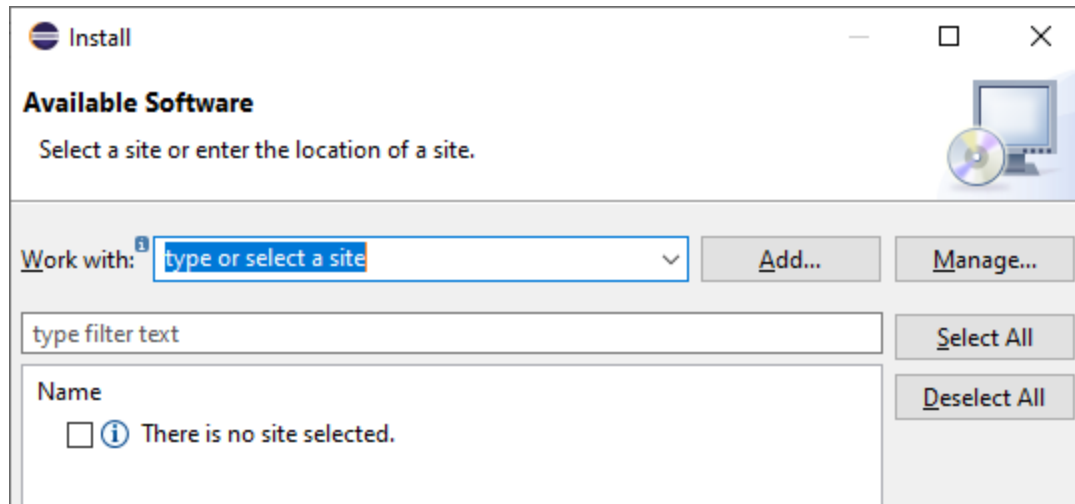
1. Run the Authentic Desktop Integration Package to start the installation wizard.
2. Go through the initial steps of the installation with the wizard.
3. In the Integration step, select *Let this wizard integrate Altova Authentic Desktop plug-in into Eclipse*, and browse for the directory where the Eclipse executable (`eclipse.exe`) is located.
4. Click **Next** and complete the installation.

The Authentic Desktop perspective and menus will be available in Eclipse the next time you start it.

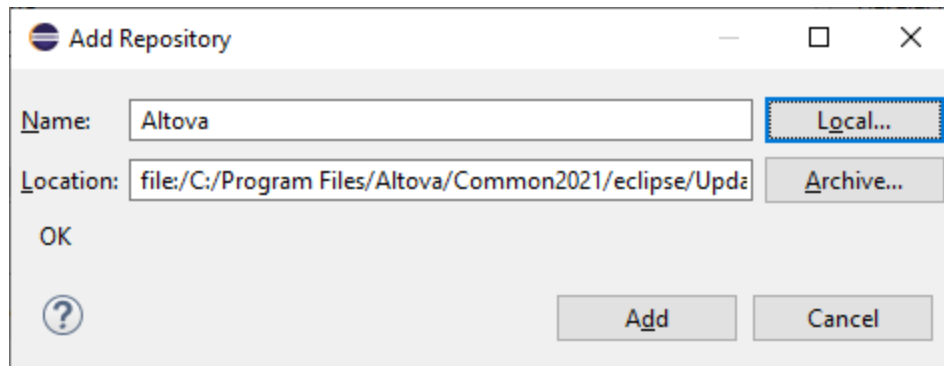
### Integrate Authentic Desktop in Eclipse manually

After you have installed the Authentic Desktop Integration Package, you can manually integrate Authentic Desktop in Eclipse as follows:

1. In Eclipse, select the menu command **Help | Install New Software**.
2. In the Install dialog box, click **Add**.



3. In the Add Repository dialog box, click **Local**. Browse for the folder `C:\Program Files\Altova\Common2024\eclipse\UpdateSite`, and select it. Provide a name for the site (such as "Altova").



4. Repeat the steps 2-3 above, this time selecting the folder `C:\Program Files\Altova\Authentic\eclipse\UpdateSite` and providing a name such as "Altova Authentic Desktop".
5. On the Install dialog box, select *Only Local Sites*. Next, select the "Altova category" folder and click **Next**.
6. Review the items to be installed and click **Next** to proceed.
7. To accept the license agreement, select the respective check box.
8. Click **Finish** to complete the installation.

**Note:** If there are problems with the plug-in (missing icons, for example), start Eclipse from the command line with the `-clean` flag.

## 12.2 Authentic Desktop Perspective in Eclipse

In Eclipse, a perspective is a GUI view that is configured with the functionality of a specific application. After Authentic Desktop has been integrated in Eclipse, a new perspective, named Authentic Desktop, becomes available in Eclipse. This perspective is a GUI that resembles the Authentic Desktop GUI and includes a number of its components.

When a file having a filetype associated with Authentic Desktop is opened (.xml) (.mfd), 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 (*see below*), thus allowing you to edit or process that file in another environment.

There are therefore two main advantage of perspectives:

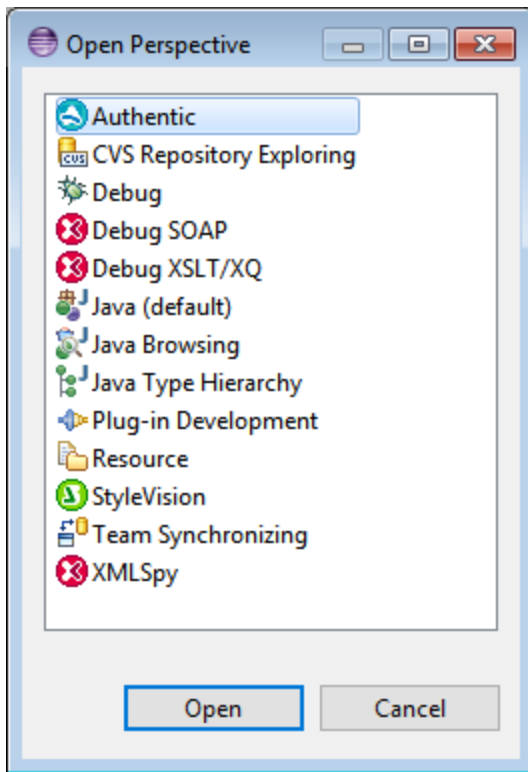
1. Being able to quickly change the working environment of the active file, and
2. Being able to switch between files without having to open a new development environment (the associated environment is available in a perspective)

Working with the Authentic Desktop perspective involves the following key procedures, which are described further below:

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

### Switch to the Authentic Desktop perspective

In Eclipse, select the command **Window | Perspective | Open Perspective | Other**. In the dialog that appears (*screenshot below*), select **Authentic Desktop**, and click **Open**.



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, you can set the required perspective to 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**.

### Preferences for the Authentic Desktop perspective

To access the Preferences of a perspective, select the command **Window | Preferences**. In the list of perspectives in the left pane, select Authentic Desktop, then select the required preferences. Finish by clicking **OK**.

The preferences of a perspective include:

- To automatically switch to the Authentic Desktop perspective when a file of an associated filetype is opened (*see above*)
- Options for including or excluding individual Authentic Desktop toolbars
- Access to Authentic Desktop options.

## Customize 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, make that perspective the active perspective and select the command **Window | Perspective | Customize Perspective**.

- In the *Toolbar Visibility* and *Menu Visibility* tabs, you can specify which toolbars and menus are to be displayed.
- In the *Action Set Availability* tab, you can add action sets to their parent menus and to the toolbar. If you wish to enable an action 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 **Apply and Close** to complete the customization and for the changes to take effect.

## 12.3 Other Authentic Desktop Entry Points in Eclipse

In addition to the Authentic Desktop perspective, two other entry points in Eclipse can be used to access Authentic Desktop functionality:

- Authentic Desktop menu
- Authentic Desktop toolbar

### Authentic Desktop menu in Eclipse

The **Authentic Desktop** menu of Eclipse contains Authentic Desktop commands that provide key Authentic Desktop functionality. These commands occur in various menus of the standalone version of Authentic Desktop.

### Authentic Desktop toolbar in Eclipse

The Authentic Desktop toolbar in Eclipse (*screenshot below*) contains two buttons.



These buttons do the following:

- Open the Authentic Desktop Help
- Provide access to Authentic Desktop commands (as an alternative to accessing them from the **Authentic Desktop** menu, *see above*).

**Note:** Toolbar commands are not supported. If you have set up a toolbar command in Authentic Desktop that runs a command or script, then this toolbar command will not be available in the plug-in.

## 13 Menu Commands

This section contains a description of all Authentic Desktop menu commands. Standard Windows commands, such as (**Open**, **Save**, **Cut**, **Copy** and **Paste**) are in the [File](#)<sup>160</sup> and [Edit](#)<sup>175</sup> menus.

Given below is a list of Authentic Desktop menus.

- [File Menu](#)<sup>160</sup>
- [Edit Menu](#)<sup>175</sup>
- [Project Menu](#)<sup>178</sup>
- [XML Menu](#)<sup>208</sup>
- [XSL/XQuery Menu](#)<sup>210</sup>
- [Authentic Menu](#)<sup>217</sup>
- [View Menu](#)<sup>227</sup>
- [Browser Menu](#)<sup>228</sup>
- [Tools Menu](#)<sup>229</sup>
- [Window Menu](#)<sup>272</sup>
- [Help Menu](#)<sup>274</sup>

This section also contains a description of the commands that can be used via the [command line](#)<sup>280</sup>.


## 13.1 File Menu

The **File** menu contains commands for file operations, ordered as in most Windows applications. These are:

- [New](#) <sup>160</sup>
- [Open](#) <sup>161</sup>
- [Reload](#) <sup>166</sup>
- [Encoding](#) <sup>166</sup>
- [Close, Close All, Close All But Active](#) <sup>167</sup>
- [Save, Save As, Save All](#) <sup>167</sup>
- [Send by Mail](#) <sup>172</sup>
- [Print](#) <sup>173</sup>
- [Print Preview](#) <sup>173</sup>
- [Print Setup](#) <sup>173</sup>
- [Recent Files](#) <sup>174</sup>
- [Exit](#) <sup>174</sup>

### 13.1.1 New

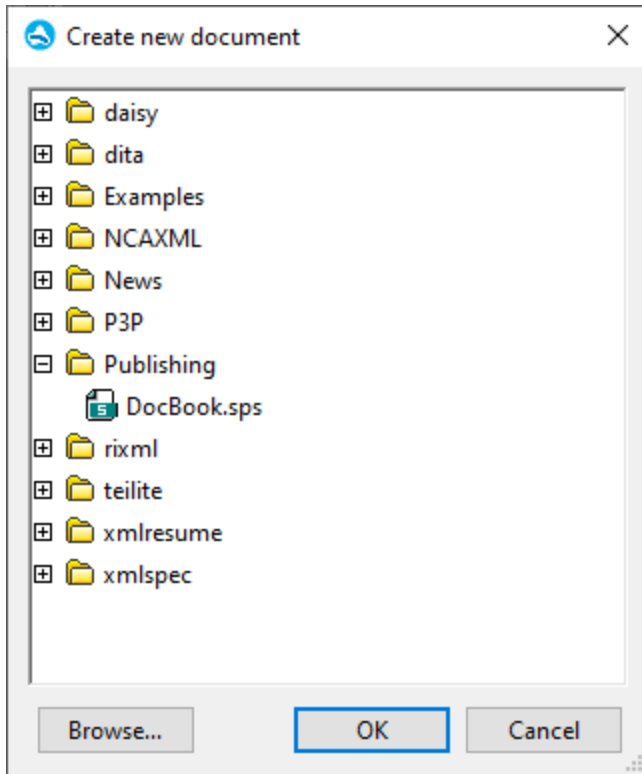
Icon and shortcut

<i>Icon:</i>	
<i>Shortcut:</i>	<b>Ctrl+N</b>

#### 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.

## 13.1.2 Open

### Icon and shortcut

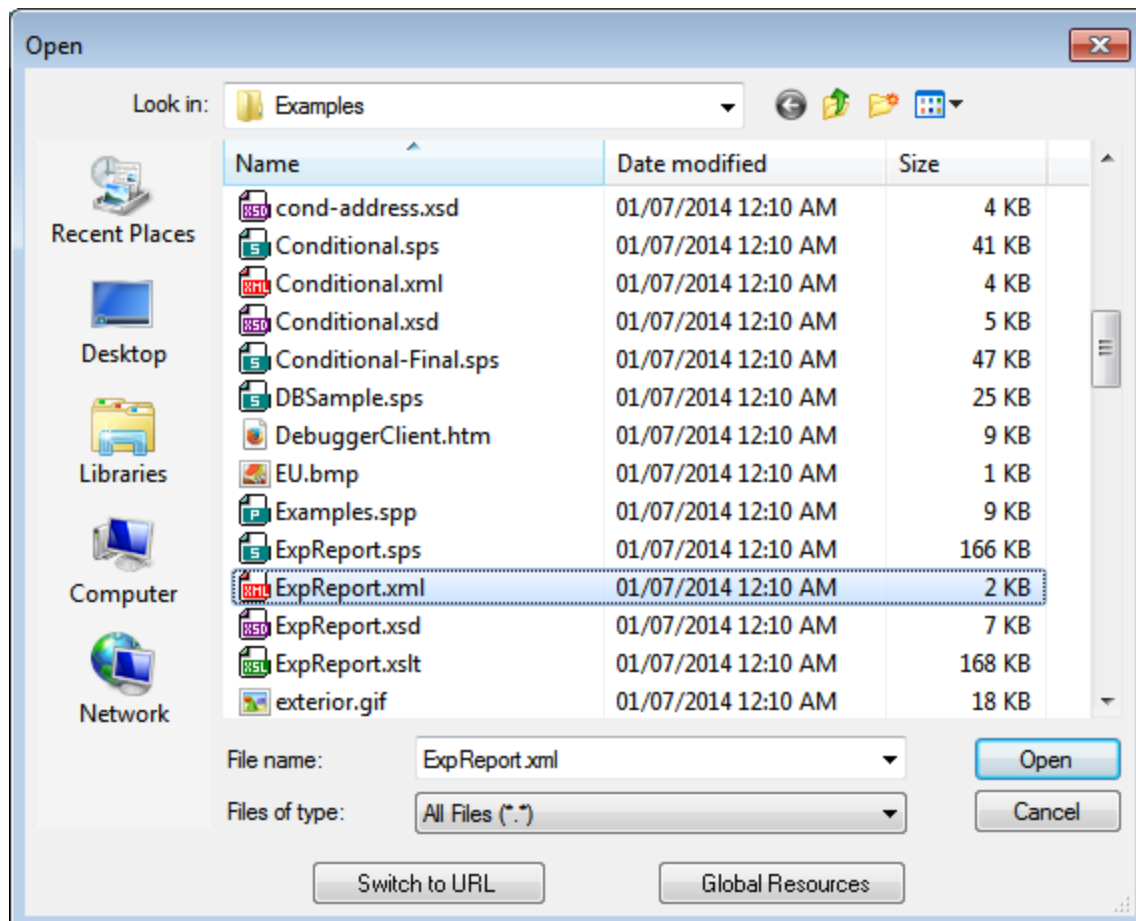
Icon:	
Shortcut:	<b>Ctrl+O</b>

## 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](#)<sup>256</sup>.) 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 \(F7\)](#)<sup>208</sup> to recheck. If you have opted for automatic [validation upon opening](#)<sup>254</sup> and the file is invalid, you will get an error message. Fix the error and select the menu command [XML | Validate XML \(F8\)](#)<sup>208</sup> to revalidate.

### ▼ Selecting and saving files via URLs and Global Resources

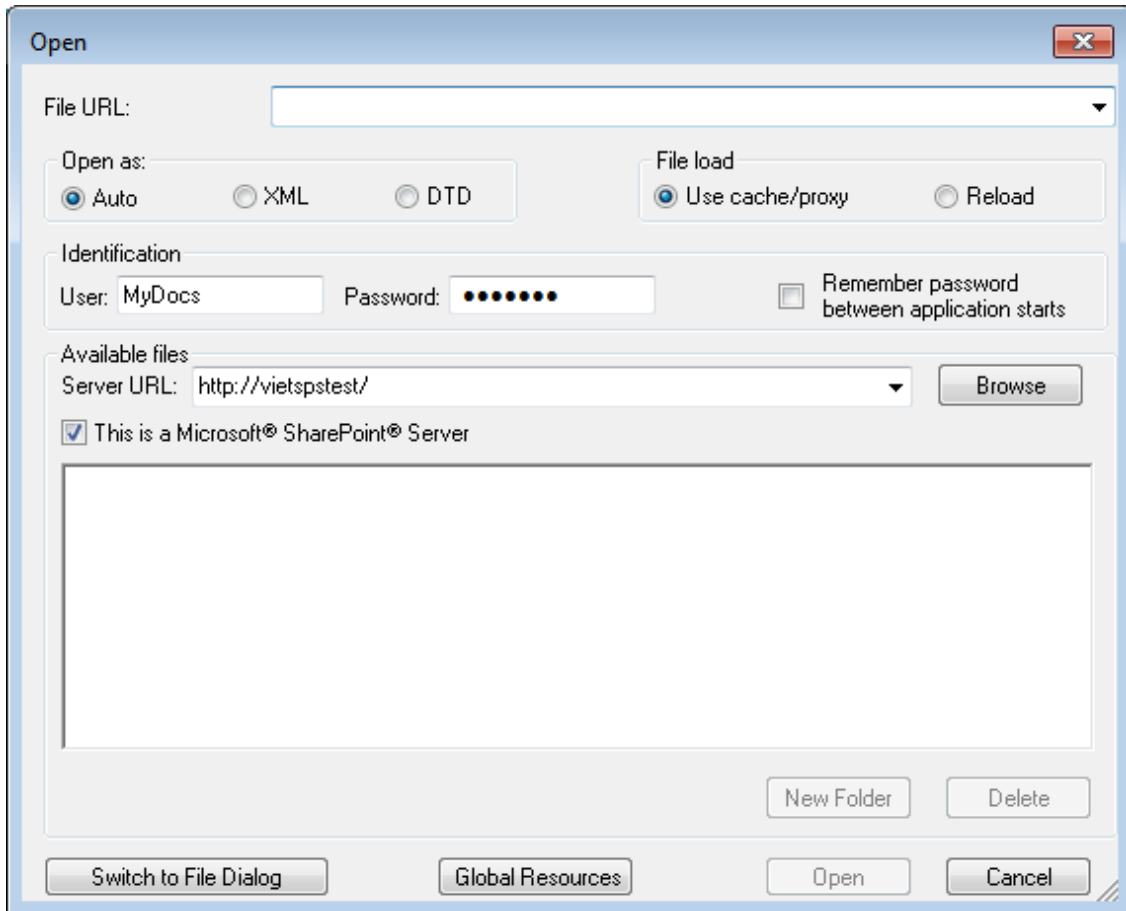
In several File Open and File Save dialogs, you can choose to select the required file or save a file via a URL or a global resource (see *screenshot below*). Click **Switch to URL** or **Global Resource** to go to one of these selection processes.



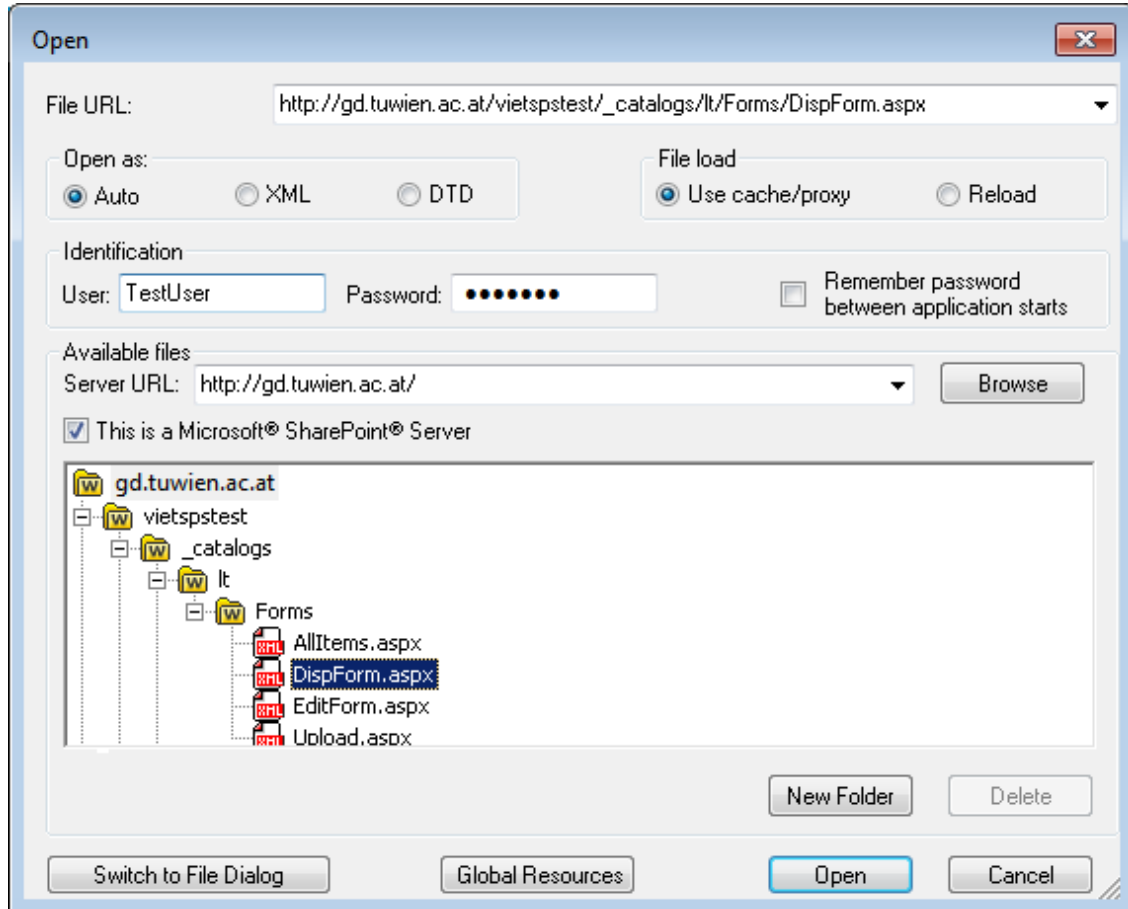
## Selecting files via URLs

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.

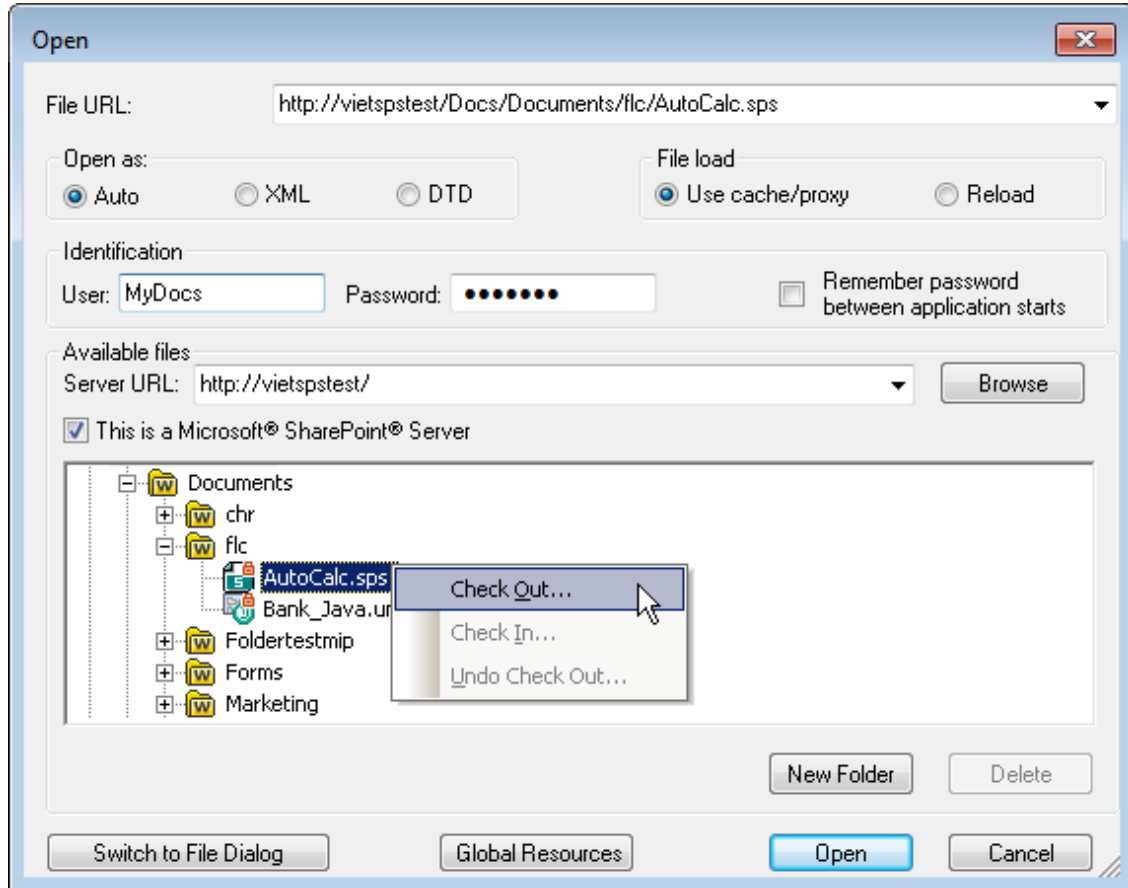
Note the following:

- 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 resources, click **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](#)<sup>102</sup>. For a general description of Global Resources, see the [Global Resources](#)<sup>90</sup> section in this documentation.

### 13.1.3 Reload

#### Icon

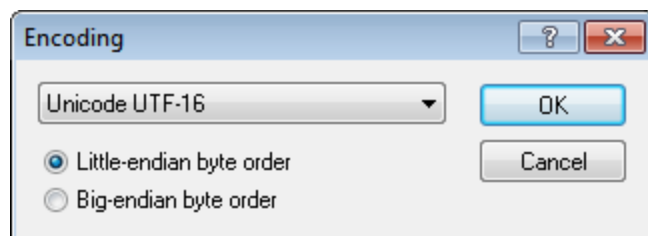


#### Description

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.

### 13.1.4 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](#)<sup>258</sup>.

**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.

### 13.1.5 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.

### 13.1.6 Save, Save As, Save All

#### Icons and shortcuts

Command	Icon	Shortcut
Save		Ctrl+S
Save All		

#### 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](#)<sup>208</sup>. The file will also be validated automatically if this option has been set in the File section of the Options dialog (**Tools | Options**<sup>254</sup>). The XML declaration is also checked for the [encoding](#)<sup>258</sup> specification, and this encoding is applied to the document when the file is saved.

## Save As

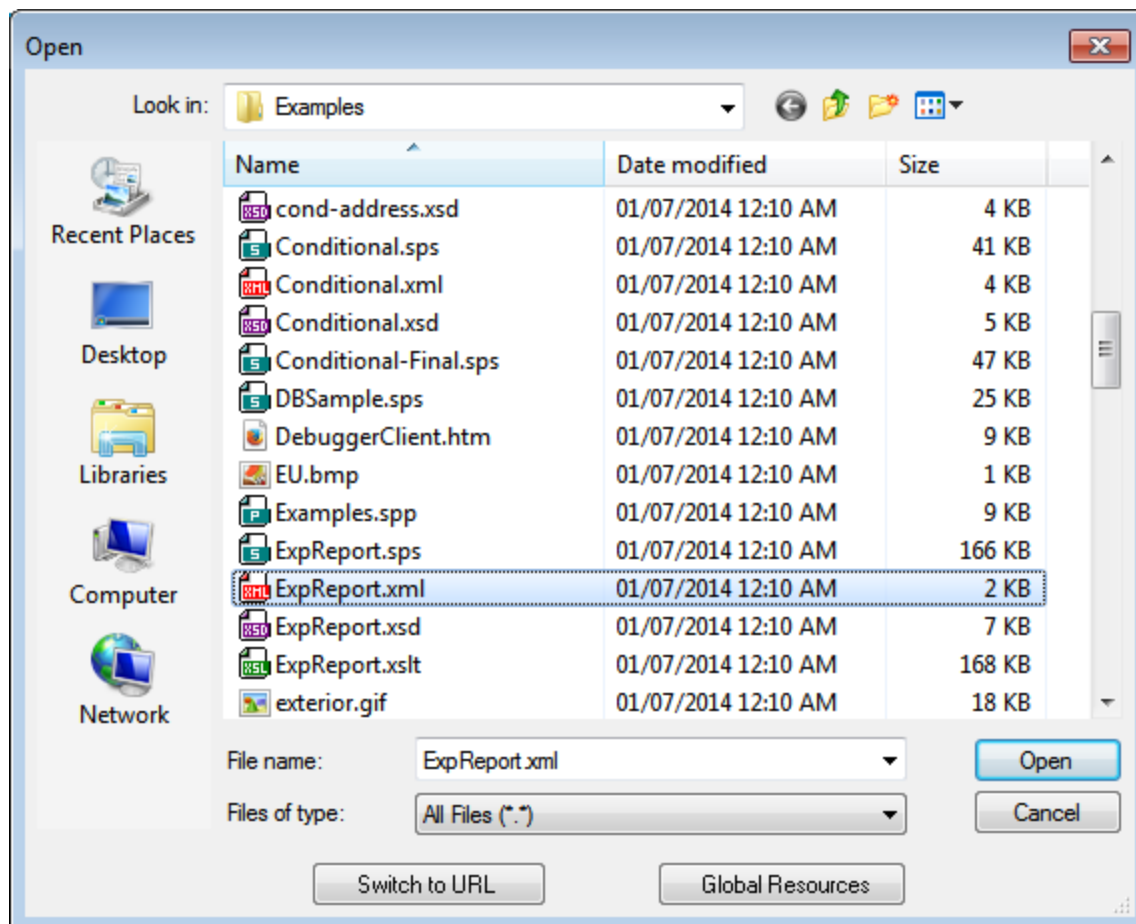
The **Save As** command pops up the familiar Windows Save As dialog box, in which you enter the name and location of the file you wish to save the active file as. The same checks and validations occur as for the **Save** command.

## 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 **Global Resource** to go to one of these selection processes.

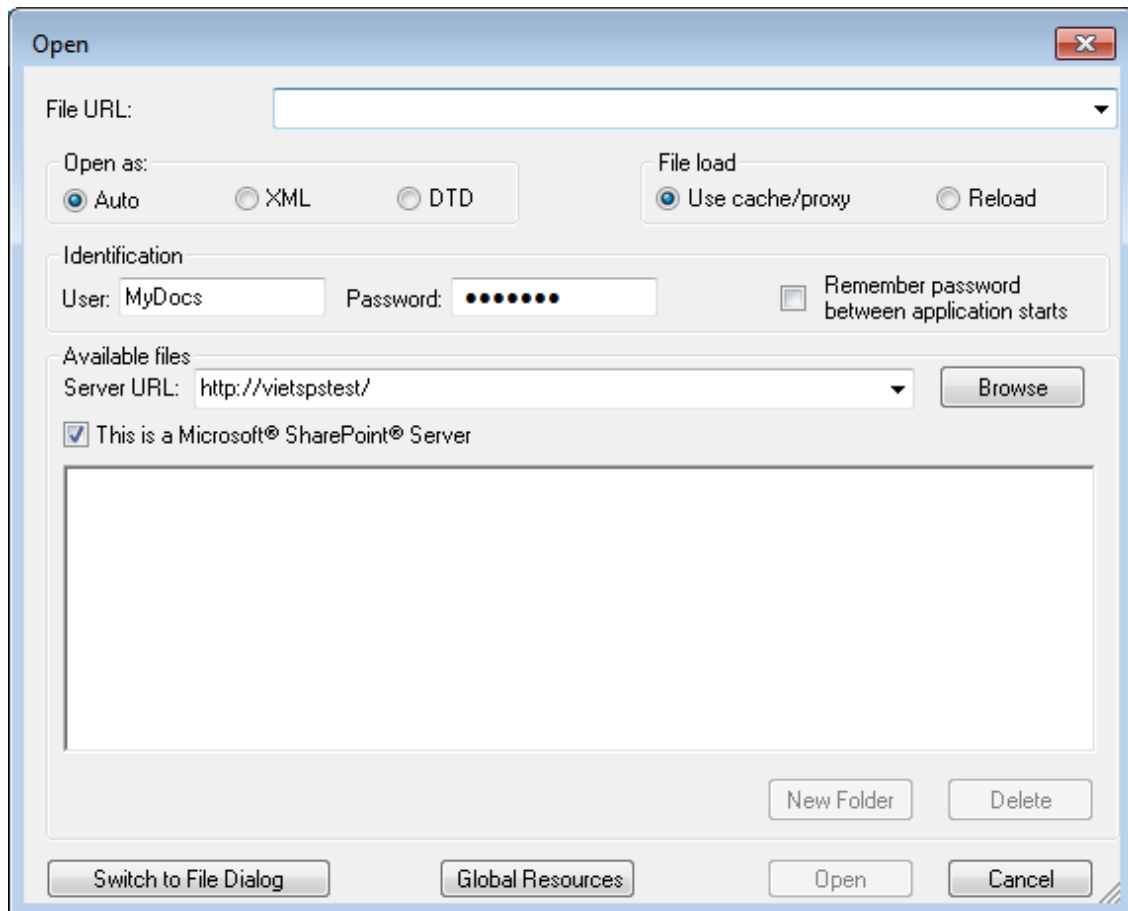




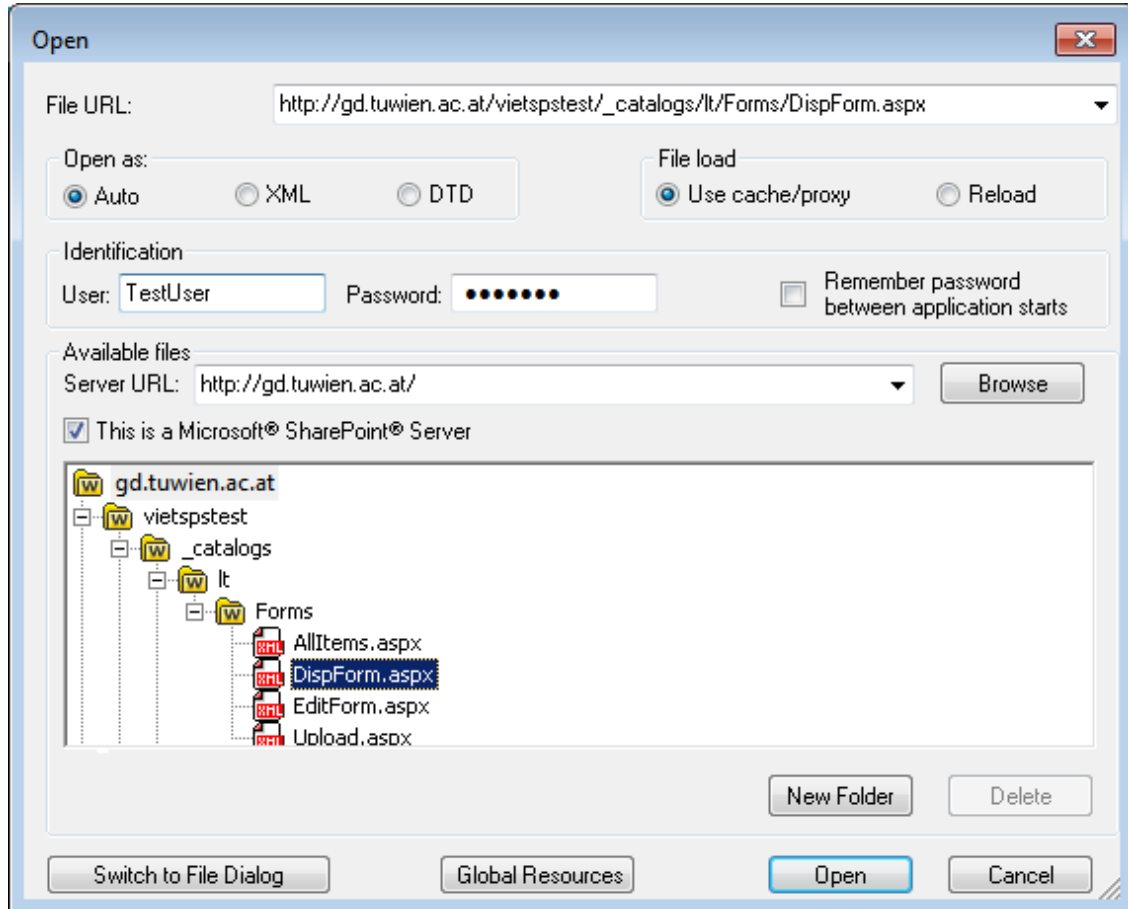
## Selecting files via URLs

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.

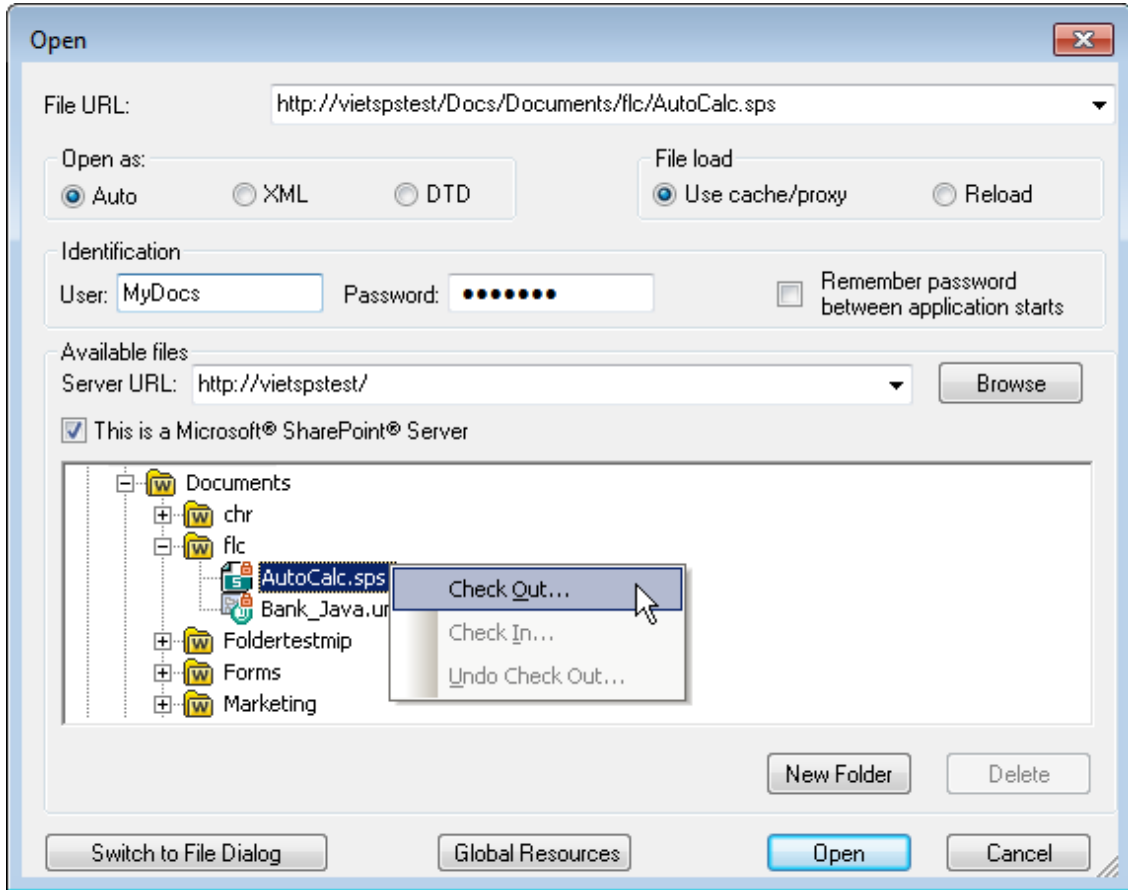
Note the following:

- 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.

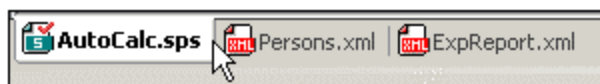


Right-clicking a file pops up a context menu containing commands available for that file (screenshot above).

- 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

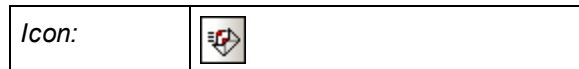
- command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in 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 resources, click **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](#)<sup>102</sup>. For a general description of Global Resources, see the [Global Resources](#)<sup>90</sup> section in this documentation.

## 13.1.7 Send by Mail

### Icon

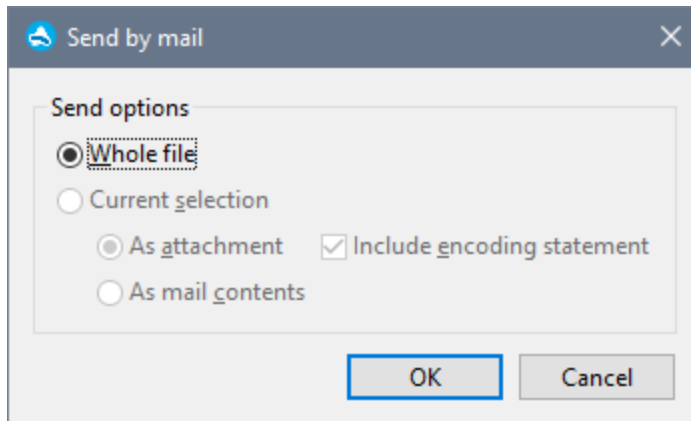


### Description

The **Send by Mail** command lets you send the active XML or PXF document as an e-mail attachment. You can also select multiple files in the Project window to send as Email attachments. 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.

What can be sent	How it can be sent
Active XML or PXF document	As e-mail attachment
One or more files in Project window	As e-mail attachment
One or more URLs in Project window	As e-mail attachment or link


- 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 with the *Whole File* being the only option that is enabled; the other options are disabled. Click **OK** to open an email with the selected file as an attachment.



- When you send files from the Project window, an email is opened with the selected files added as attachments.
- URLs in the project window can be sent as an attachment or as a link. Choose the option you want and click **OK**.

## 13.1.8 Print

### Icon and shortcut

Icon:	
Shortcut:	<b>Ctrl+P</b>

### Description

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

## 13.1.9 Print Preview, Print Setup

### Print Preview

The **Print Preview** command is available in Authentic View. It 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. Navigation buttons are at the bottom of 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.

## 13.1.10 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.

## 13.2 Edit Menu

The **Edit** menu contains commands for editing documents in Authentic Desktop. These include the familiar [Undo](#)<sup>175</sup>, [Redo](#)<sup>175</sup>, [Cut](#)<sup>175</sup>, [Copy](#)<sup>175</sup>, [Paste](#)<sup>175</sup>, [Delete](#)<sup>175</sup>, [Select All](#)<sup>176</sup>, [Find](#)<sup>176</sup>, [Find Next](#)<sup>176</sup> and [Replace](#)<sup>177</sup> commands.

### 13.2.1 Undo, Redo

Icons and shortcuts

Command	Icon	Shortcut
<b>Undo</b>		<b>Ctrl+Z</b>
<b>Redo</b>		<b>Ctrl+Y</b>

#### 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.

### 13.2.2 Cut, Copy, Paste, Delete

Icons and shortcuts

Command	Icon	Shortcut
<b>Cut</b>		<b>Ctrl+X</b> or <b>Shift+Del</b>
<b>Copy</b>		<b>Ctrl+C</b>
<b>Paste</b>		<b>Ctrl+V</b>
<b>Delete</b>		<b>Del</b>

## 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.

### 13.2.3 Select All

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

### 13.2.4 Find, Find Next

Icons and shortcuts

Command	Icon	Shortcut
<b>Find</b>		<b>Ctrl+F</b>
<b>Find Next</b>		<b>F3</b>

#### 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.

**Note:** 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.



## 13.2.5 Replace

### Icons and shortcuts

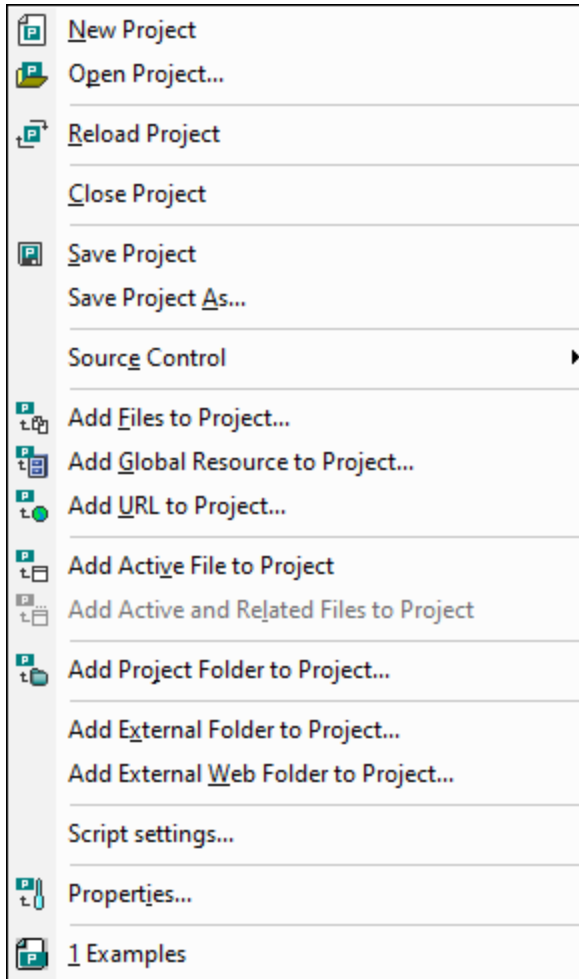
<i>Command</i>	<i>Icon</i>	<i>Shortcut</i>
<b>Replace</b>		<b>Ctrl+H</b>

### Description

The **Replace** command enables you to find and replace one text string with another. It features the same options as the [Find](#) <sup>176</sup> command. You can replace each item individually, or you can use the **Replace All** button to perform a global find-and-replace operation.

## 13.3 Project Menu

Authentic Desktop uses the familiar tree view to manage multiple files or URLs in XML projects. [Files](#)<sup>196</sup> and [URLs](#)<sup>196</sup> can be grouped into [folders](#)<sup>197</sup> 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
|      |
|      |-Folder4
```

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.

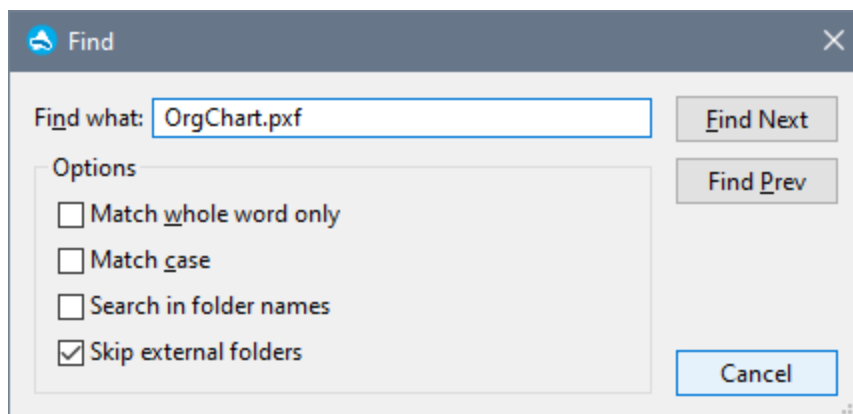
## Drag-and-drop

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](#)<sup>197</sup> 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](#)<sup>90</sup>. 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 file's 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 shown 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.

### 13.3.1 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. The project's name is assigned when you save the project as a `.spp` file.

### 13.3.2 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.

### 13.3.3 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.


**Note:** Project files (`.spp` files) are actually XML documents that you can edit like any regular XML File.

### 13.3.4 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.

### 13.3.5 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.

### 13.3.6 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](#)<sup>109</sup>. 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](#)<sup>188</sup>.
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](#)<sup>182</sup> 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](#)<sup>266</sup>), or (ii) in the Change Source Control dialog ([Project | Source Control | Change Source Control](#)<sup>195</sup>).

**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.

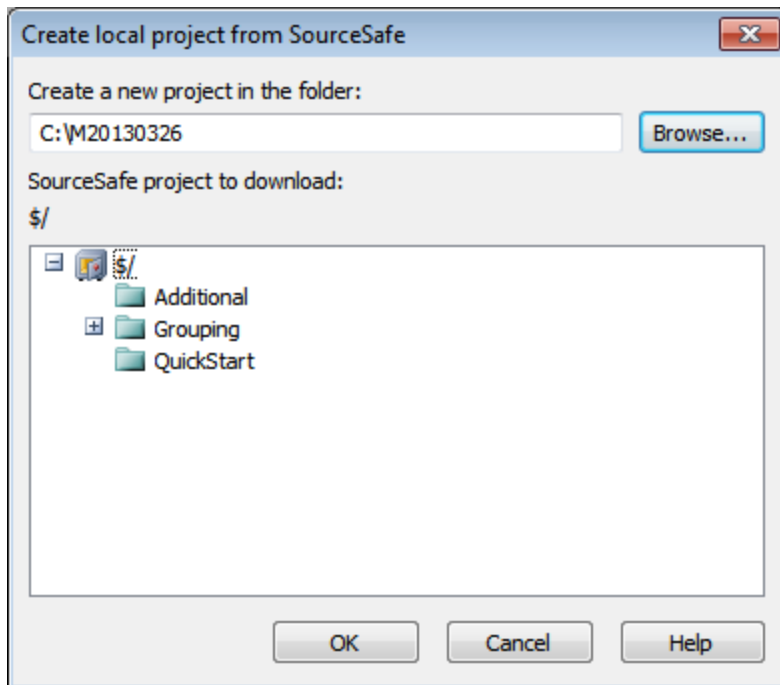
For additional information, see the section, [Source Control](#)<sup>106</sup>.

### 13.3.6.1 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 by the \$ sign; the local folder is `C:\M20130326`.



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.

	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.

### 13.3.6.2 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.

	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.

### 13.3.6.3 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** <sup>184</sup> 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.

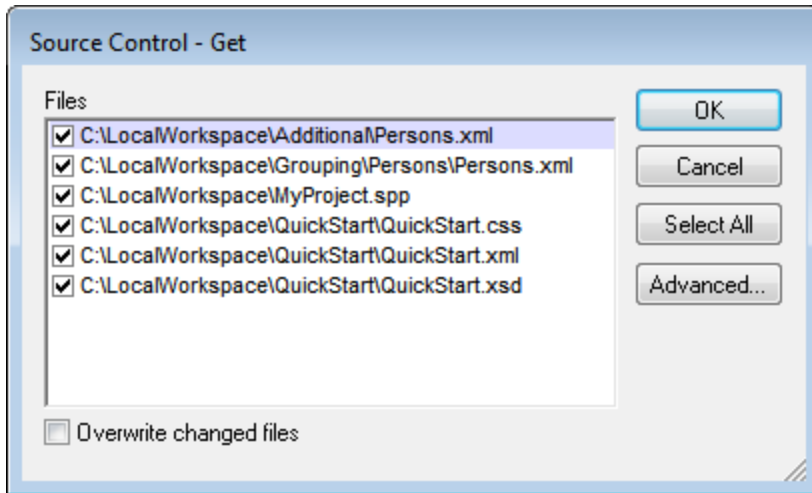
**Note:** This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.

### 13.3.6.4 Get, Get Folders

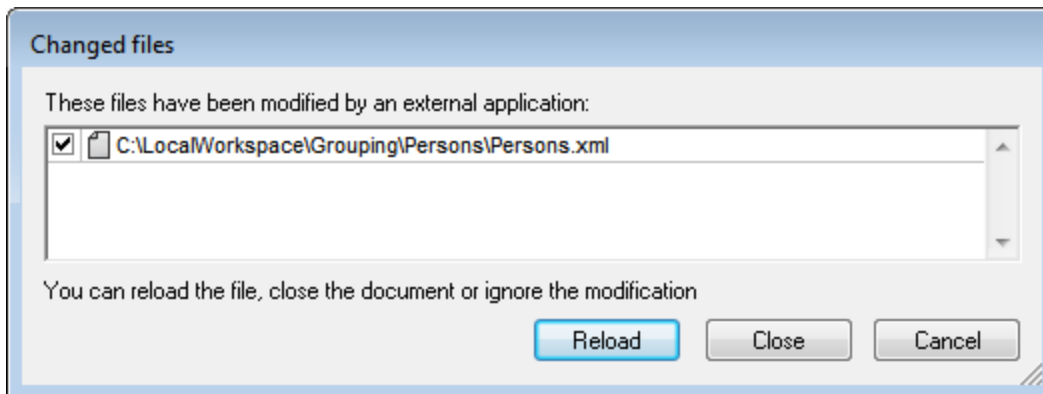
The **Get** command (in the **Project | Source Control** menu) retrieves files from the repository as read-only files. (To be able to edit a file, you must check it out.) The Get dialog lists the files in the object (project or folder) on which the **Get** command was executed (*see screenshot below*).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



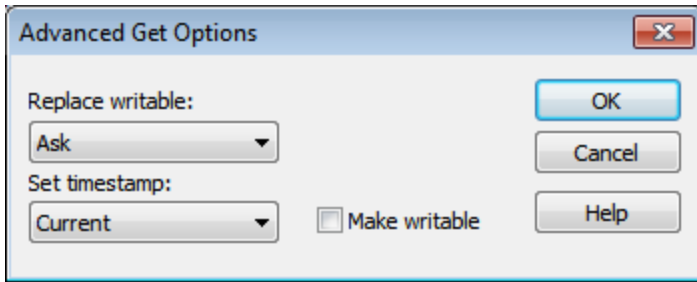


You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking **OK**, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (*screenshot below*) asking whether you wish to reload the file/s (**Reload**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

### 13.3.6.5 Check Out, Check In

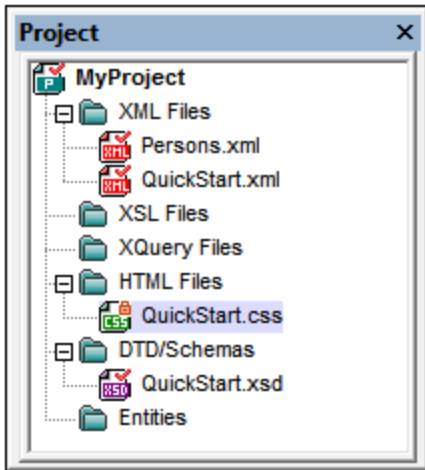
After a project file has been placed under source control, it can be checked out or checked in by selecting the file (in the Project window) and clicking the respective command in the **Project | Source Control** menu: **Check Out** and **Check In**.

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

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.



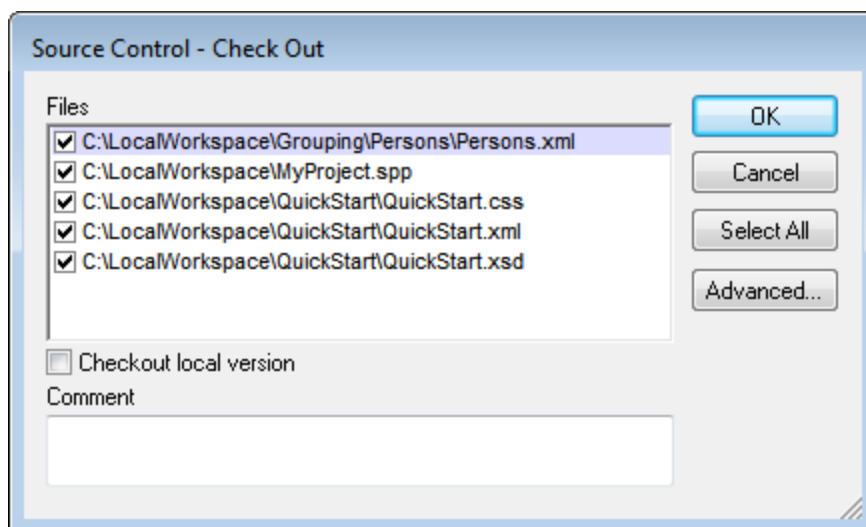
### Saving and rejecting editing changes

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**.

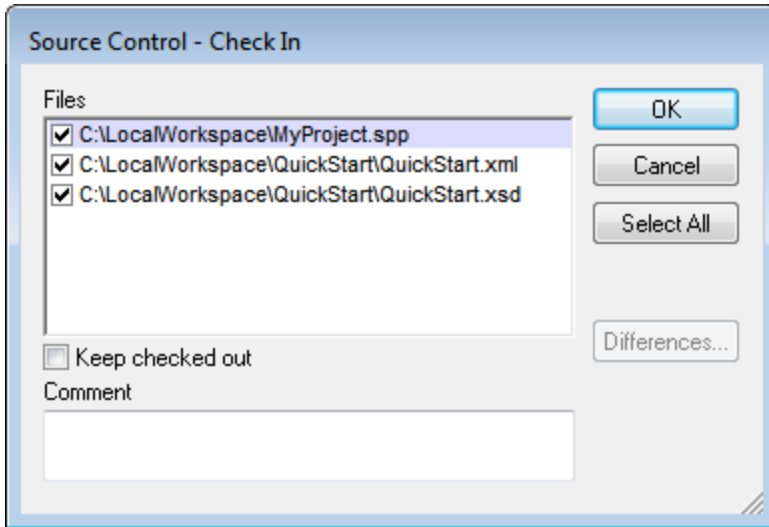
### Checking out

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.

### 13.3.6.6 Undo Check Out

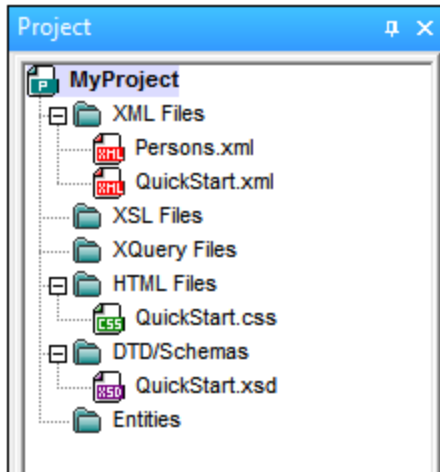
If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command **Project | Source Control | Undo Check Out**.

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

	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.

### 13.3.6.7 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.






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](#)<sup>114</sup>.

### Source control symbols

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

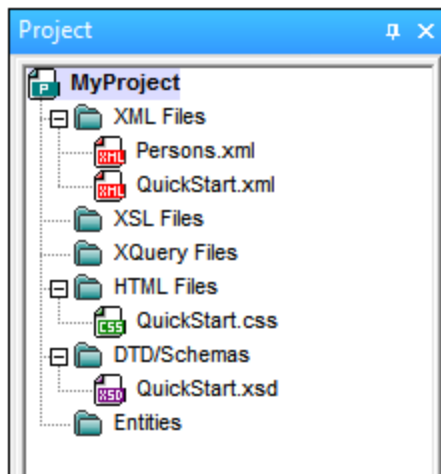
### 13.3.6.8 Remove from Source Control

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.

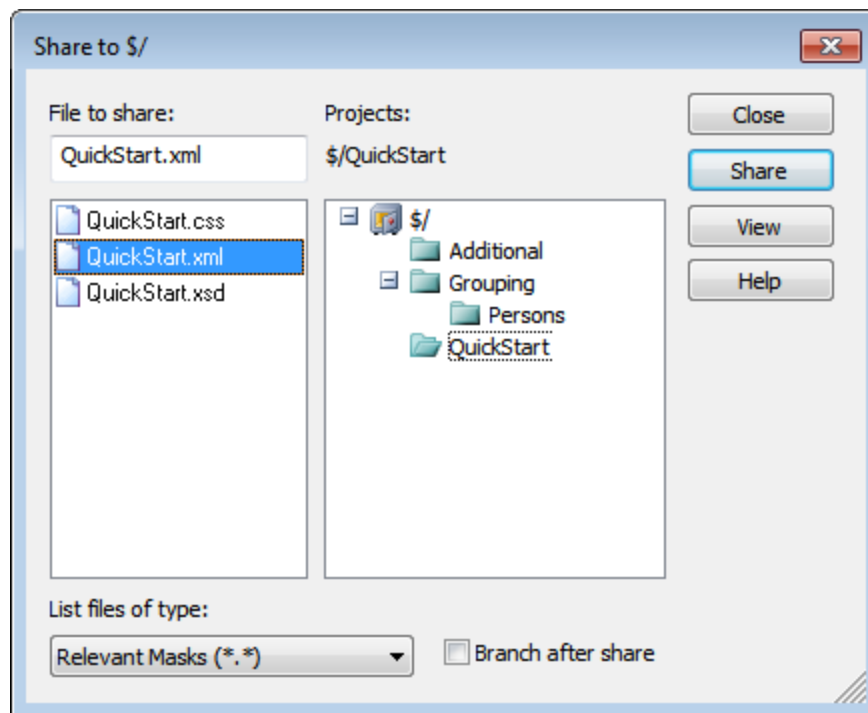
### 13.3.6.9 Share from Source Control

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**.



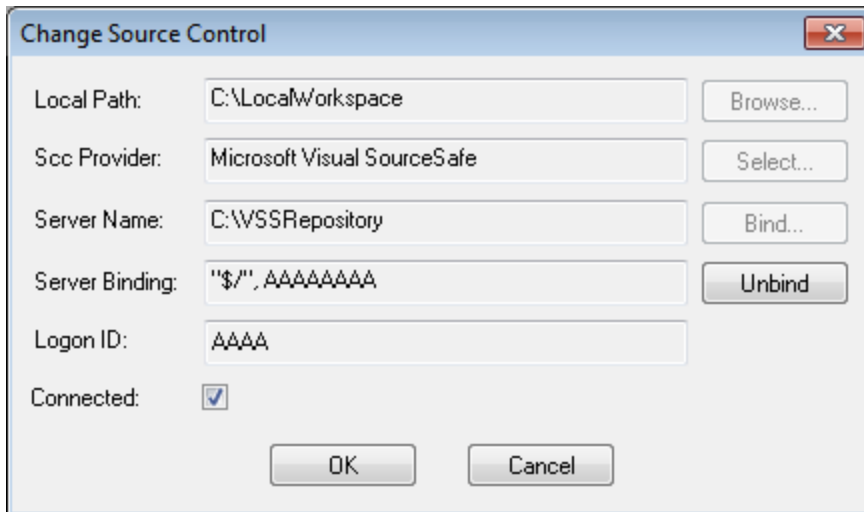
The Share To [Folder] dialog (*screenshot below*) pops up.



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.

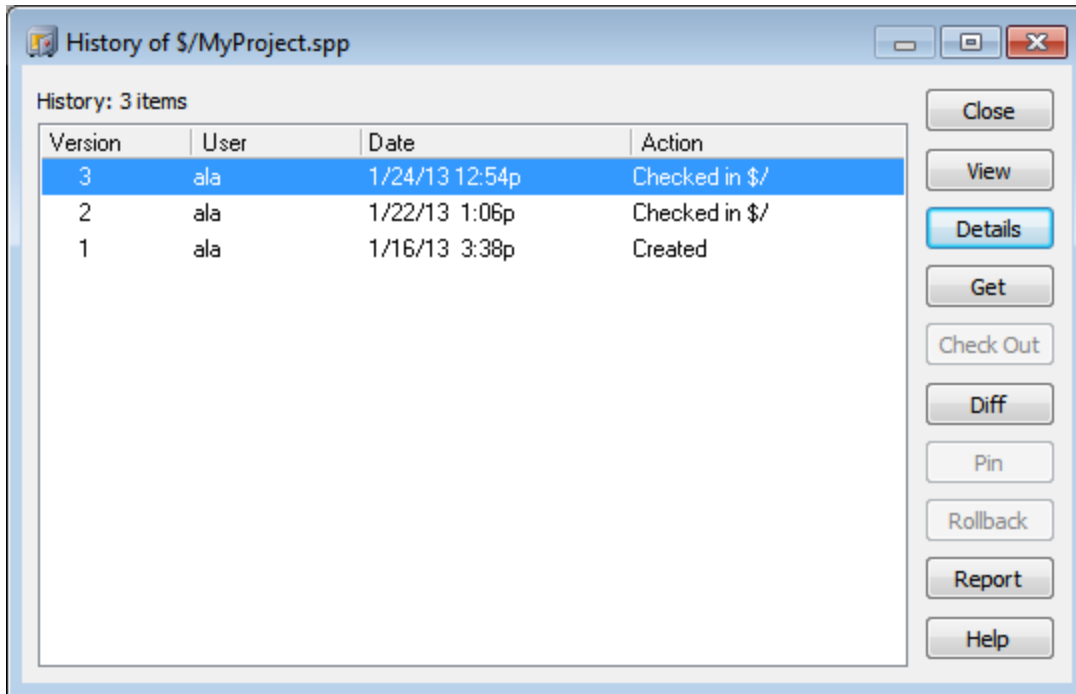


For more details about sharing using your source control system, see the source control system's user documentation.

### 13.3.6.10 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.



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:

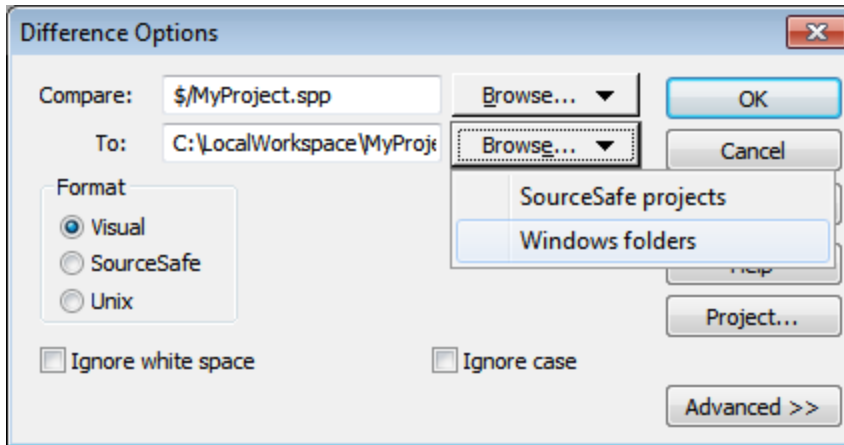
- *Close*: Closes this dialog box.
- *View*: Opens a dialog box in which you can select the type of file viewer.
- *Details*: Opens a dialog box in which you can see the [properties](#)<sup>194</sup> of the currently active file.
- *Get*: Retrieves a previous file version and places it in the working directory.
- *Check Out*: Allows you to check out a previous version of the file.
- *Diff*: Opens the [Difference options](#)<sup>192</sup> 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.

### 13.3.6.11 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.

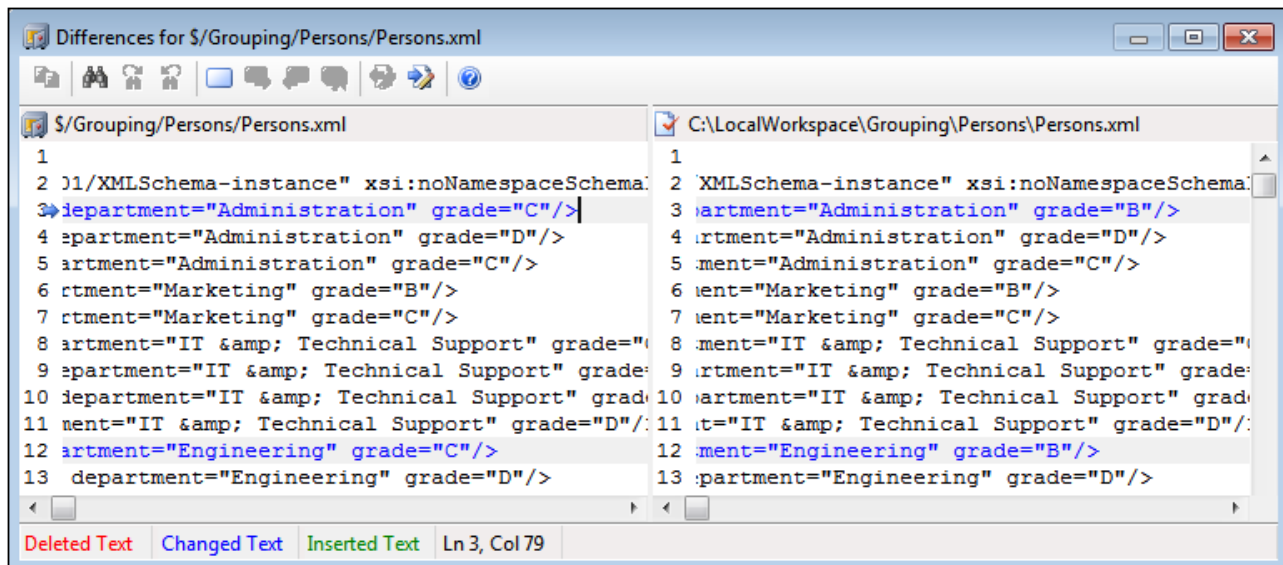




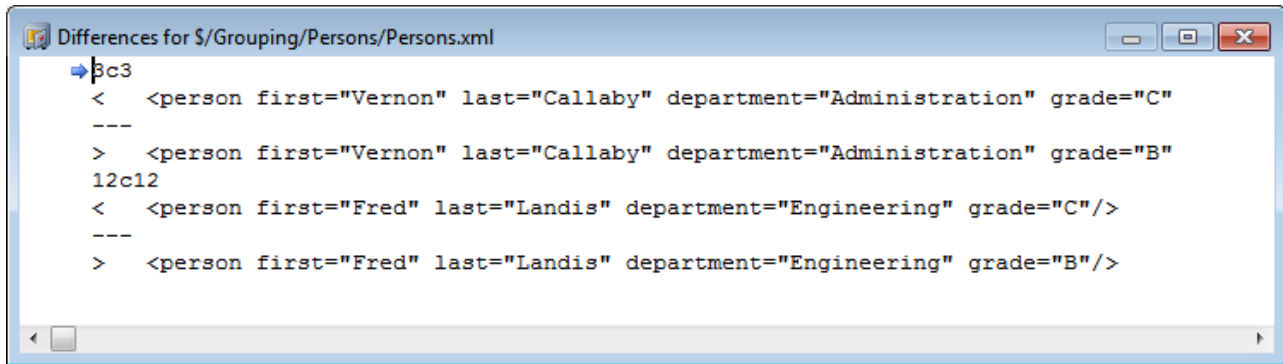
The repository and local versions are shown by default in the *Compare* and *To* text fields respectively. You can browse for other files as follows:

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.



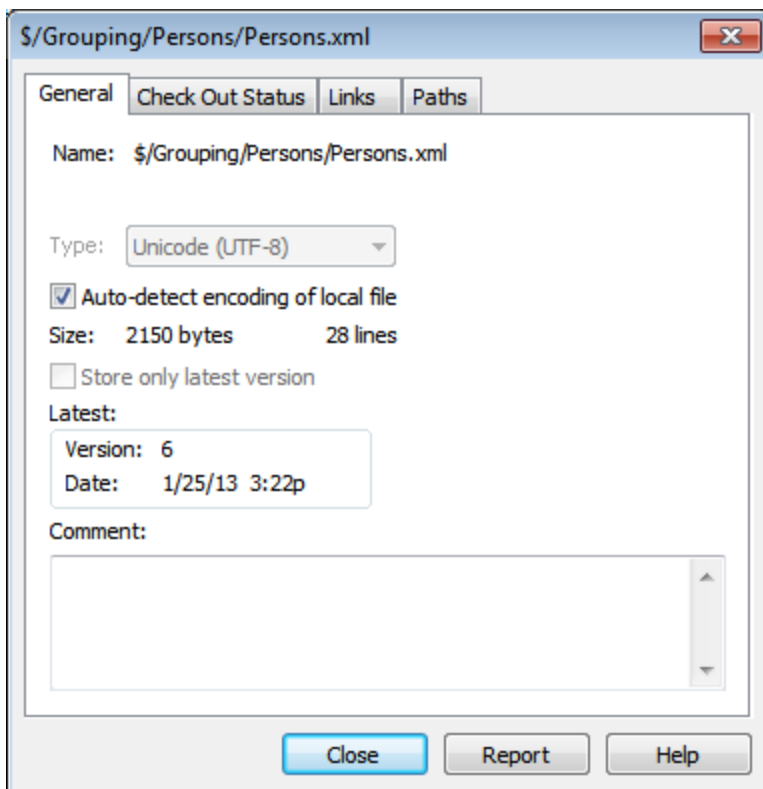
```
→ 3c3
< <person first="Vernon" last="Callaby" department="Administration" grade="C"
---
> <person first="Vernon" last="Callaby" department="Administration" grade="B"
12c12
< <person first="Fred" last="Landis" department="Engineering" grade="C"/>
---
> <person first="Fred" last="Landis" department="Engineering" grade="B"/>
```

For a detailed description of how your source control system handles differencing, see the product's user documentation.

### 13.3.6.12 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.

### 13.3.6.13 Refresh Status

The **Refresh Status** command refreshes the status of all project files independent of their current status.

### 13.3.6.14 Source Control Manager

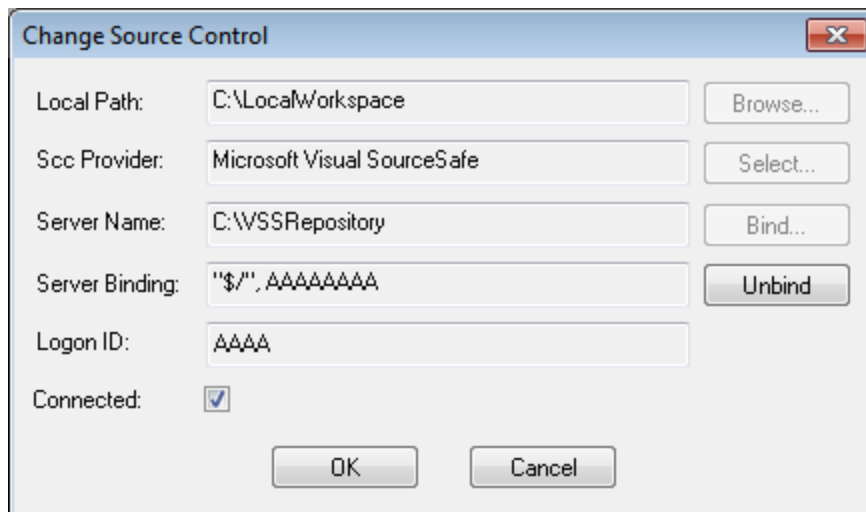
The **Source Control Manager** command starts your source control software with its native user interface.

### 13.3.6.15 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.



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.

### 13.3.7 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](#)<sup>204</sup> dialog box.

### 13.3.8 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](#)<sup>204</sup> 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](#)<sup>204</sup> dialog box. For a description of global resources, see the Global Resources section in this documentation.

### 13.3.9 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.

### 13.3.10 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.

### 13.3.11 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.

### 13.3.12 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](#)<sup>197</sup> are yellow.

### 13.3.13 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.

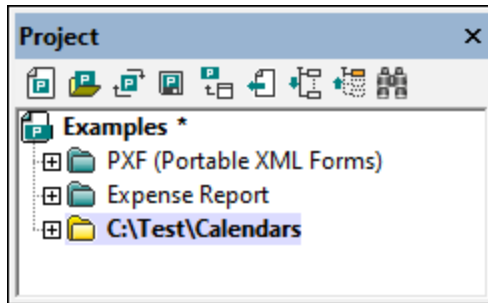
**Note:** External folders are yellow, while [project folders](#)<sup>197</sup> are green.

**Note:** Files contained in external folders cannot be placed under source control.

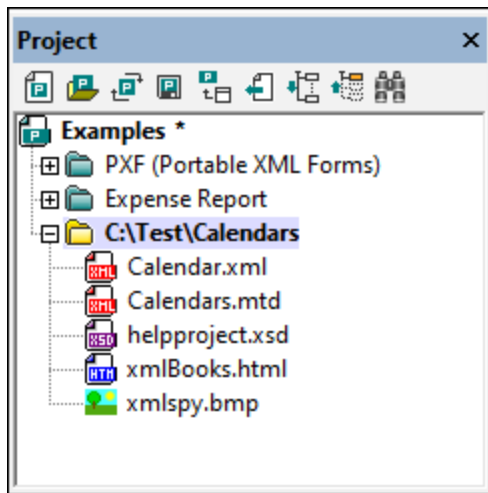
### Adding external folders to projects

To add an external folder to the project:

1. Select the menu command **Project | Add External Folder to Project**.
2. Select the folder you want to include and click **OK** to confirm. The selected folder now appears in the Project window.



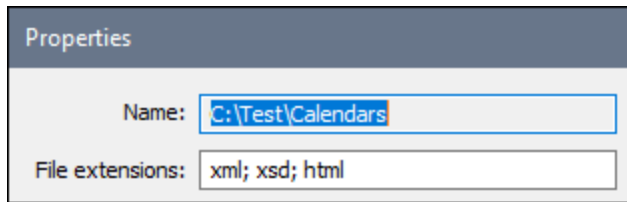
3. Click the plus icon to view the folder contents.



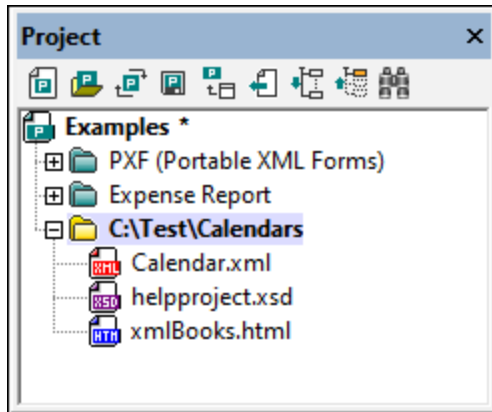
### Filtering contents of folders

To filter the contents of the folder:

1. Right-click the external folder that you added, and select **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, separating file types with a semicolon (see *screenshot above*).
3. Click **OK** to confirm.



The Project window now only shows the selected file types.

## 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. Select the folder and click the menu command **XML | Check well-formedness** or **Validate XML** (hotkey **F7** or **F8**, respectively). 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**.

## 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.

### 13.3.14 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:

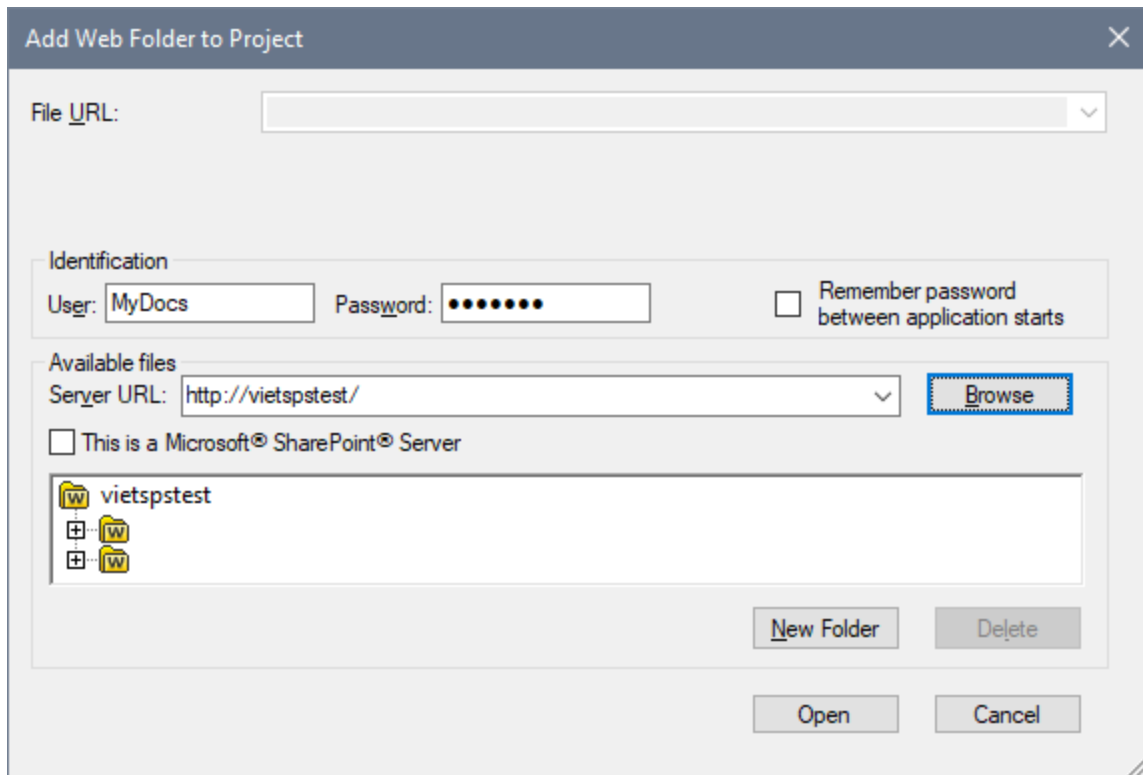
1. Select the menu option **Project | Add External Web Folder to Project**. This opens the Add Web Folder to Project dialog box (*screenshot below*).

The screenshot shows the 'Add Web Folder to Project' dialog box. It features a title bar with a close button. The main content area includes a 'File URL' field with a dropdown arrow. Below this is the 'Identification' section, which contains a 'User' field with the text 'MyDocs', a 'Password' field with masked characters, and a checked checkbox labeled 'Remember password between application starts'. The 'Available files' section includes a 'Server URL' field with the text 'http://vietsptest/' and a 'Browse' button. Below the 'Server URL' field is a checkbox labeled 'This is a Microsoft SharePoint Server'. A large empty text area is located below the checkbox. At the bottom of the dialog are four buttons: 'New Folder', 'Delete', 'Open', and 'Cancel'.

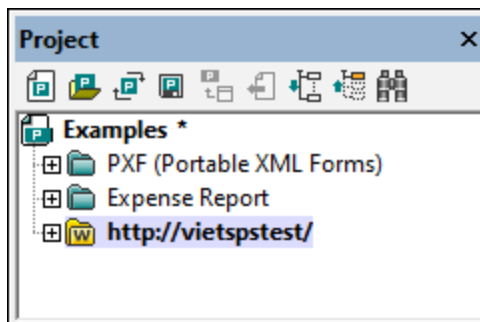
2. Click in the Server URL field and enter the URL of the server URL. If the server is a Microsoft® SharePoint® Server, check this option. See the *Folders on a Microsoft® SharePoint® Server* section below for further information about working with files on this type of server.



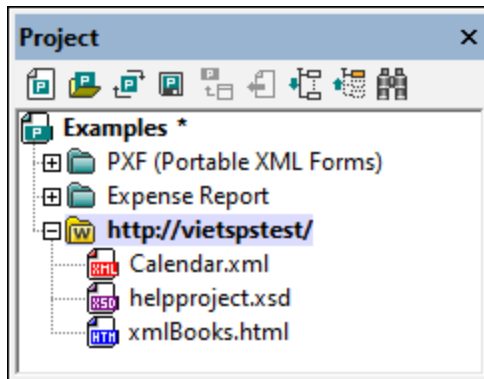
3. If the server is password-protected, enter your User ID and password in the *User* and *Password* fields.
4. Click **Browse** to connect to the server and view the available folders.



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 menu command **XML | Check well-formedness** or **XML | Validate XML** icon (hotkey **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

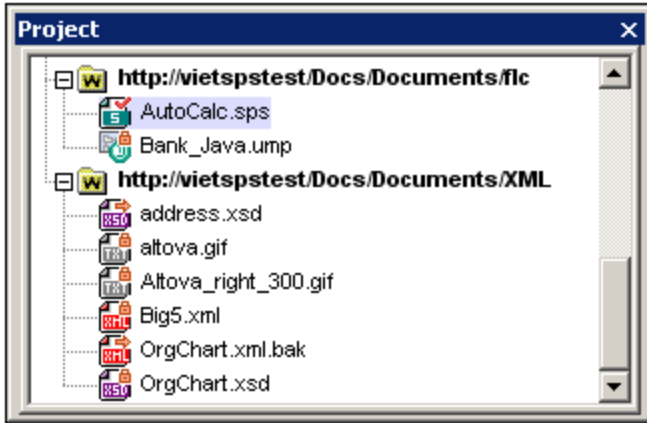
Since it is the Web folder that has been added to the project, it is only the Web folder (and not files within it) that can be deleted from the project. You can delete a Web folder from a project, by either (i) right-clicking the folder and selecting **Delete**, or (ii) selecting the folder and pressing 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](#)<sup>204</sup>, thereby enabling you to skip the verification process each time the server is accessed.

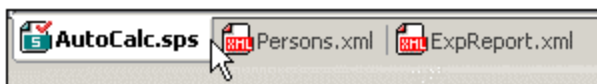


In the Project window (*screenshot below*), file icons have symbols that indicate the check-in/check-out status of files. The various file icons are shown below:

	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*).

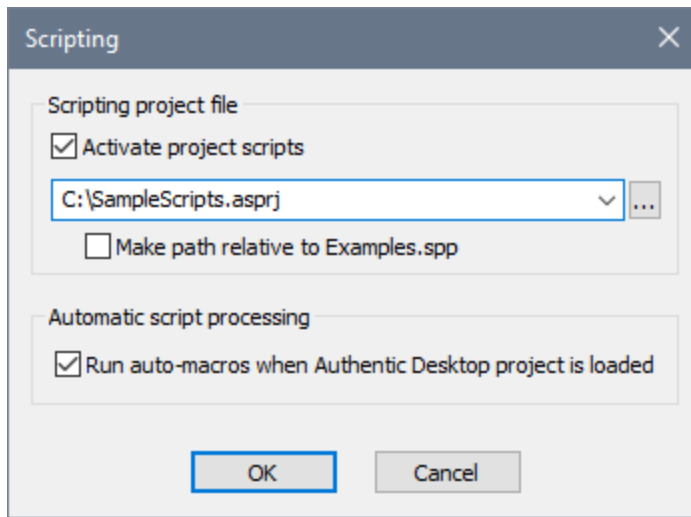


- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in 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**.

### 13.3.15 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.

### 13.3.16 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

Name: Invoices-EU

File

Validation

Validate with: Browse... Window...

XSL transformation of XML files

Use this XSL: C:\Invoices\reports.xslt Browse... Window...

XSL:FO transformation of XML files

Use this XSL: C:\Invoices\reportsFO.xslt Browse... Window...

XQuery/Update transformation of XML files

Use this XQuery: Browse... Window...

Input XML for XSL/XQuery/Update transformation

Use this XML: Browse... Window...

Output files for XSL/XQuery/Update transformation

Save in folder: C:\Invoices\Reports Browse...

File extension: .html

XULE execution

Use this XBRL: C:\Invoices\XBRL\InvoicesEU.xbrl Browse... Window...

Authentic view

Use config.: Browse... Window...

JSON conformant files

Treat as: Auto detect

## 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, 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](#)<sup>208</sup> 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](#)<sup>210</sup> 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.

### XQuery/Update transformation of XML files

The XQuery or XQuery Update file to be used for XQuery executions or XQuery Update executions of XML files in the folder.

### Input XML for XSL/XQuery/Update transformation of XML files

The XML file to use as input for XSLT transformations or XQuery/XQuery Update executions with the respective XSLT, XQuery, or XQuery Update files in the folder.

### Output files for XSL/XQuery/Update transformation

The destination directory of transformations, and, optionally, the file extension of the result document.

### XULE execution

The XBRL instance file to process with the XULE document that is active in the XMLSpy application window.

### 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.

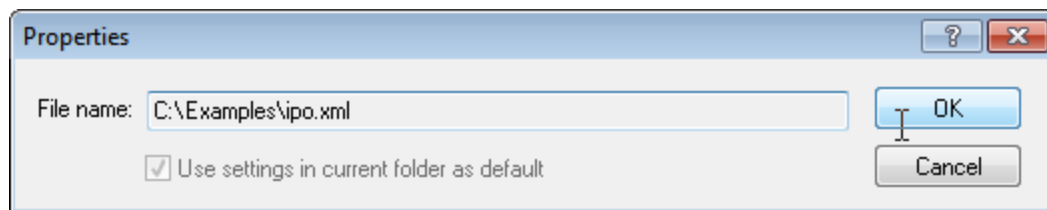
### JSON conformant files

This property specifies whether a project folder contains JSON Schema files or JSON instance files. It can be very useful to help identify JSON Schema files if the files are not clearly identified as a JSON Schema file by the `$schema` keyword and the files reference each other. You can set it to *JSON Instance*, *JSON Schema*, or *Auto detect*. The default setting of *Auto-detect* would cause XMLSpy to check the structure and content of JSON files to determine its type.

## Notes about project properties

Note the following points about precedence when, within a hierarchy, different files are specified to be used for procedures such as validation or transformation:

- 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 inside the XML file.
- 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.



### 13.3.17 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](#)<sup>254</sup> that you used, whenever you start Authentic Desktop. (**Tools | Options | File** section, Project | Open last project on program start).

## 13.4 XML Menu

The **XML** menu contains commands that are commonly used when working with XML documents. Among the most frequently used XML tasks are checks for the [well-formedness](#)<sup>208</sup> of documents and [validity](#)<sup>208</sup> of XML documents. Commands for these tasks are in this menu.

### 13.4.1 Check Well-Formedness



F7

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, reloaded, or saved. If an XML document is not well-formed, this will be automatically reported in the Messages window and the XML document will be displayed in a Text View. You can fix the error in the Text View and then switch back to Authentic View.

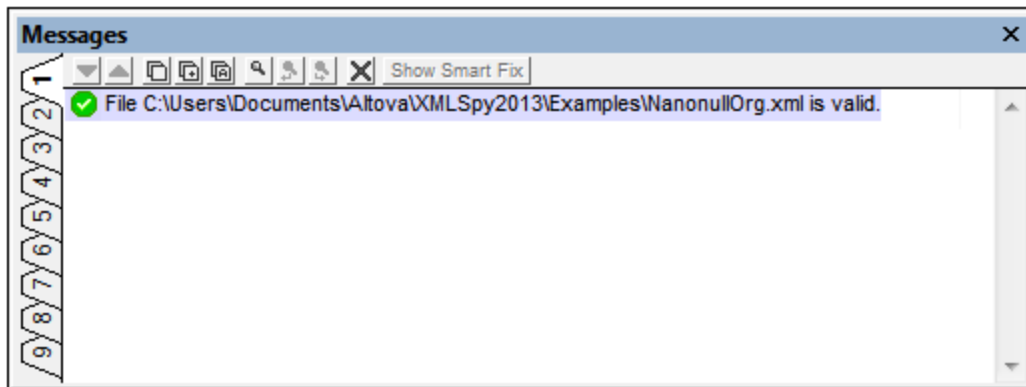
### 13.4.2 Validate XML



F8

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**). Note that you can also validate as you edit if you switch on the [Validate on Edit](#)<sup>209</sup> command.

If a document is valid, a message to this effect is displayed in the Messages window (*see screenshot below*). 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. After correcting an error, you should 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 2024

**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.

### Validation and Schema Manager

If a document is validated against a schema that is not installed but is available via [Schema Manager](#)<sup>131</sup>, then the installation via Schema Manager will be triggered automatically. However, if the schema package to be installed via Schema Manager contains namespace mappings, then there will be no automatic installation; in this case, you must start Schema Manager, select the package/s you want to install, and run the installation. If, after installation, Authentic Desktop is not able to correctly locate a schema component, then restart Authentic Desktop and try again.

## 13.4.3 Validate on Edit

The **Validate on Edit** command toggles on/off the *Validate on Edit* mode, which enables validation as you type in Authentic View. The mode can also be switched on/off via the command's toolbar button or the *Validation > On Edit* option of the [File section of the Options dialog](#)<sup>254</sup>.

## 13.5 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](#)<sup>261</sup>) in order to be able to use it directly from within the Authentic Desktop interface.

The following commands are available in the XSL/XQuery menu:

- [XSL Transformation](#)<sup>210</sup>
- [XSL-FO Transformation](#)<sup>211</sup>
- [XSL Parameters / XQuery Variables](#)<sup>212</sup>

### 13.5.1 XSL Transformation



F10

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](#)<sup>261</sup> 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](#)<sup>204</sup> 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](#)<sup>161</sup> button) or a file in one of the open windows in XMLSpy (click the **Window** button).

#### Automating transformations with RaptorXML

**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).

## 13.5.2 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.

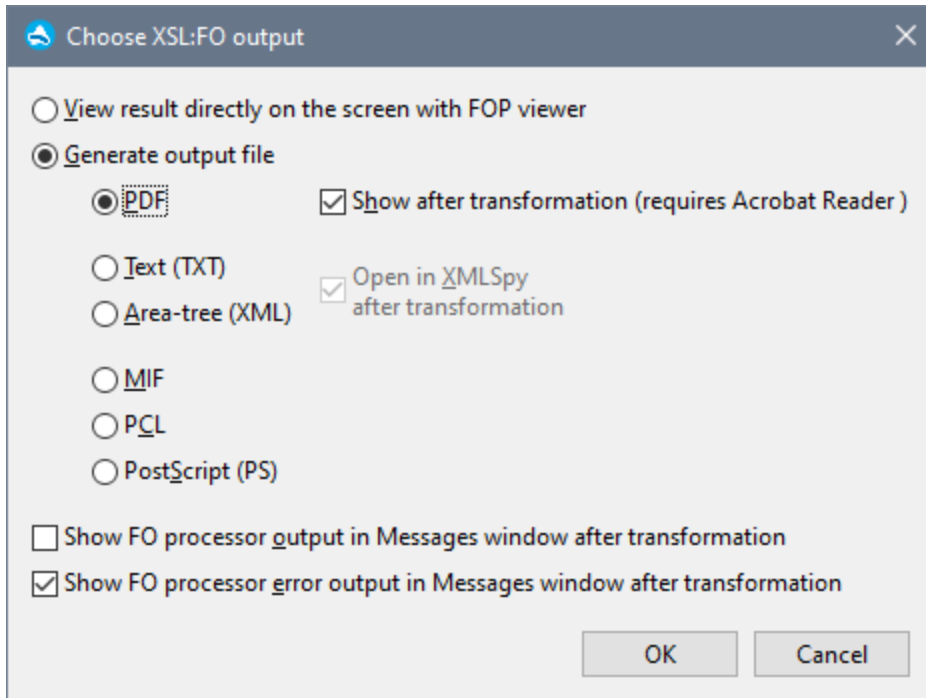
1. The XML document is transformed to an FO document using an XSLT stylesheet.
2. The FO document is processed by an FO processor to generate PDF (or some alternative output).

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**<sup>161</sup> 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**<sup>261</sup> 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**<sup>261</sup> 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**<sup>261</sup> 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 of these two options, check the appropriate check box at the bottom of the dialog.

Note the following points:

- 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](#)<sup>261</sup> 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.

### 13.5.3 XSL Parameters / XQuery Variables

The **XSL/XQuery | XSL Parameters/XQuery Variables** command opens the XSLT Input Parameters/XQuery External Variables dialog (see *screenshot below*). 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.

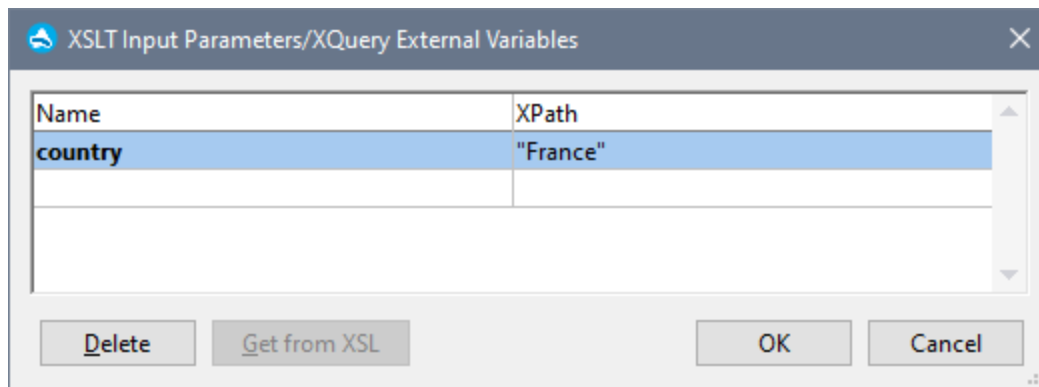
- When the **XQuery Execution** command in the XSL/XQuery menu is used to process an XQuery document, the XQuery external variable values currently saved in the dialog are passed to the XQuery document for the execution.

**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 is an XPath expression. Note that text strings in XPath are delimited by quotes.



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.

**Note:** The **Get from XSL** button is enabled in Authentic View only when an XSLT document is the active document. It inserts parameters declared in the active XSLT document into the dialog together with the default values of these parameters.

## Usage example for XSLT parameters

We have an XML document that contains the names of countries and their respective capitals:

```
<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>
```

```
</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*).

```
<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:

```
<country><name>USA</name><capital>Washington DC</capital></country>
```

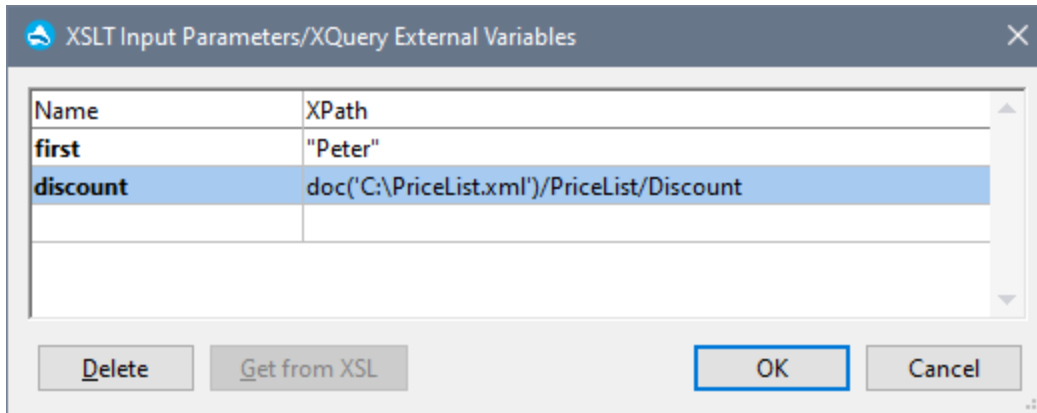
Now, if in the XSLT Input Parameters/XQuery External Variables dialog you create a parameter named `country` and give it a value (*see screenshot above*), then this value will be passed to the parameter `country` in the XSLT stylesheet for the transformation. In this way, you can pass different values to different parameters at run time.

Note the following:

- If you use the **XSL:FO Transformation** command (**XSL/XQuery | XSL:FO Transformation**), then parameters entered in the XSLT Input Parameters/XQuery External Variables dialog are **not** passed to the stylesheet. 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** 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.



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 Authentic Desktop has been closed.

### 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:

```
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 version "1.0";
declare variable $first as xs:string external;
let $first := "Paul"
let $last := "Jones"
return concat($first, " ", $last )
```



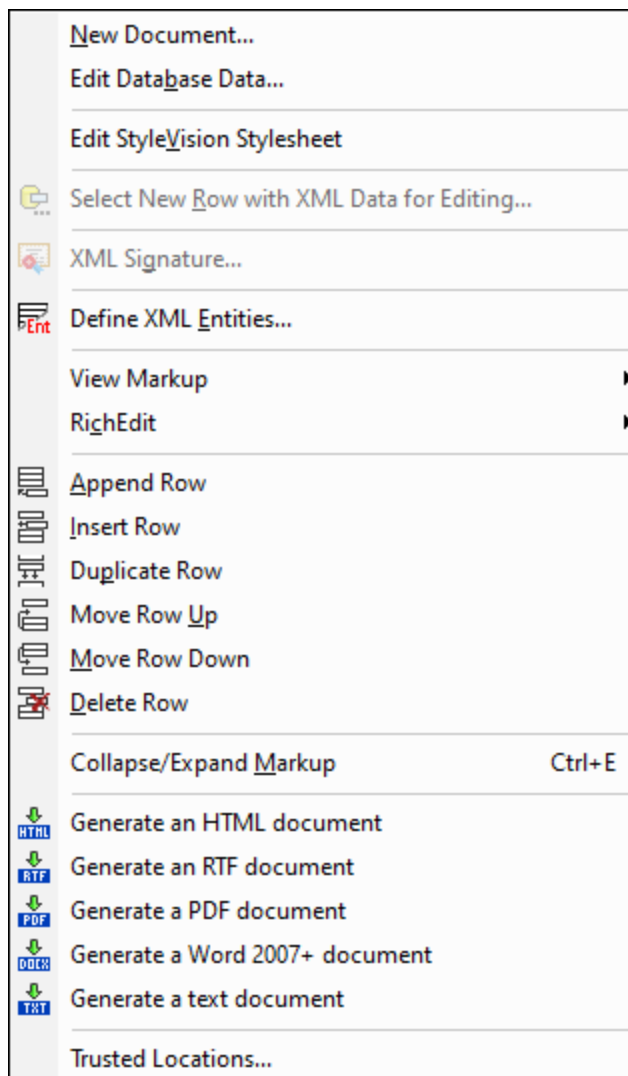


## 13.6 Authentic Menu

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.

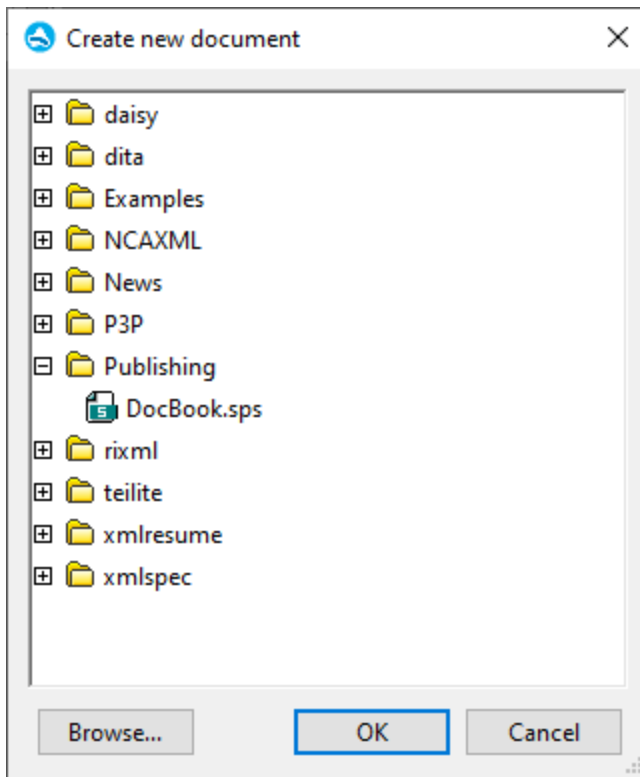
Additionally, StyleVision Power Stylesheets can be created to display an editable XML view of a database. The StyleVision Power Stylesheet contains information for connecting to the database, displaying the data from the database in Authentic View, and writing back to the database.

The **Authentic** menu contains commands relevant to editing XML documents in Authentic View. For a tutorial on Authentic View, see the [Authentic View Tutorials](#)<sup>23</sup> section.



## 13.6.1 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.



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.

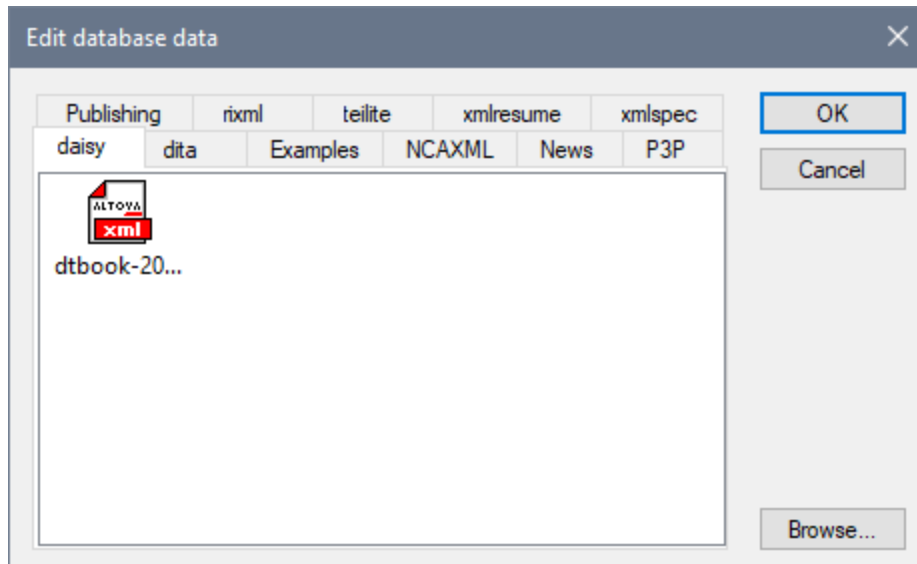
The new XML document template will therefore have the documentation presentation properties defined in the SPS and the data of the XML file that was selected as the Template XML File. The Authentic View user can now edit the XML document template in a graphical WYSIWYG interface, and save it as an XML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

## 13.6.2 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 (*screenshot below*). 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.



**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.

**Note:** StyleVision Power Stylesheets are created using Altova StyleVision.

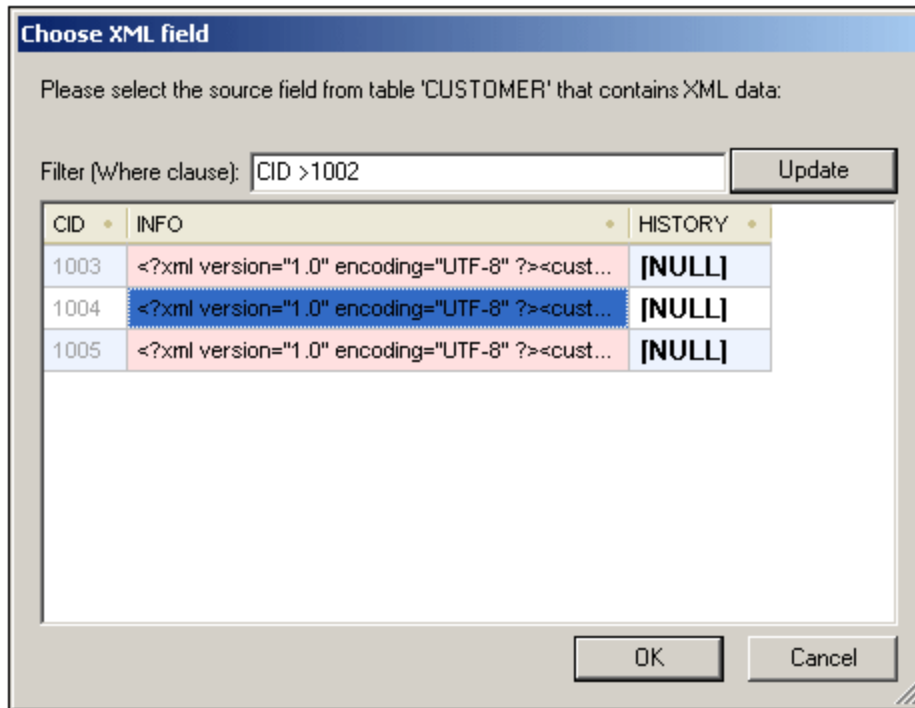
### 13.6.3 Edit StyleVision Stylesheet

The **Authentic | Edit StyleVision Stylesheet** command is available only in Authentic View, that is, only if a StyleVision Power Stylesheet has been assigned to the XML document. It starts StyleVision and allows you to edit the StyleVision Power Stylesheet immediately in StyleVision.

### 13.6.4 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.



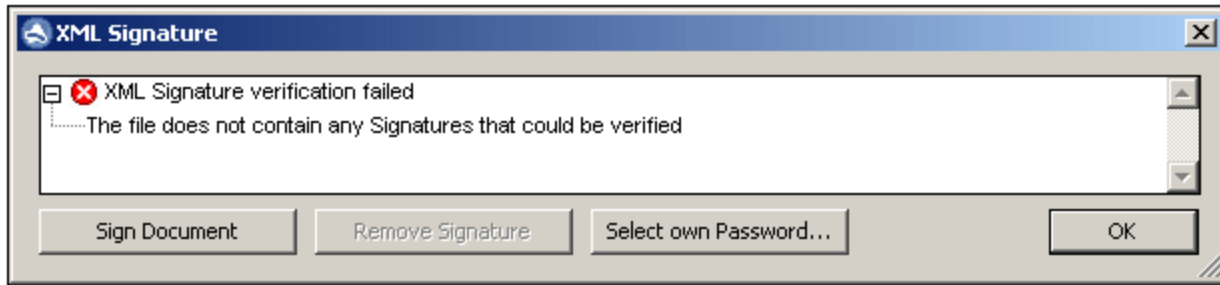
You can enter a filter for this table. The filter should be an SQL `WHERE` clause (just the condition, without the `WHERE` keyword, for example: `CID>1002`). Click **Update** to refresh the dialog. In the screenshot above, you can see the result of a filtered view. Next, select the cell containing the required XML document and click **OK**. The XML document in the selected cell (row) is loaded into Authentic View.

### 13.6.5 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

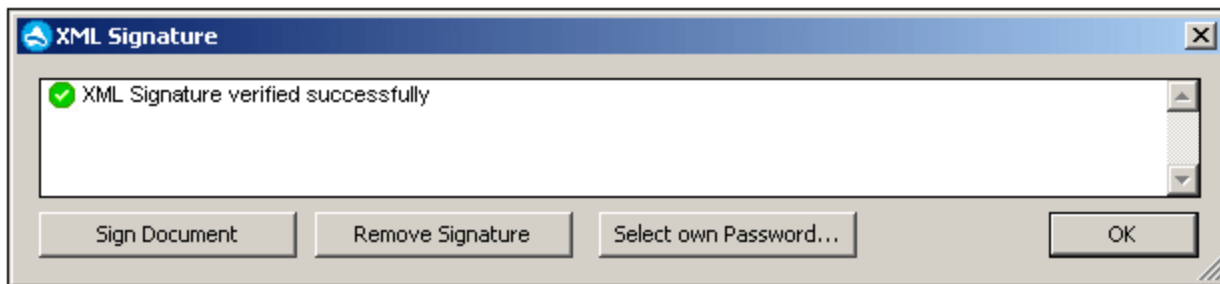
Clicking the **XML Signature** command starts the signature verification process. If no signature is present in the document, a message to that effect is displayed in the XML Signature dialog (*see screenshot below*), and the dialog will have a button that enables the Authentic View user to sign the 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.

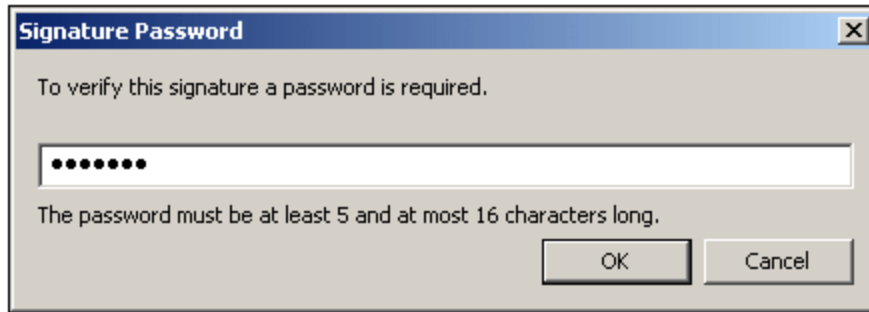
### Verification and authentication information

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.



### 13.6.6 Define XML Entities

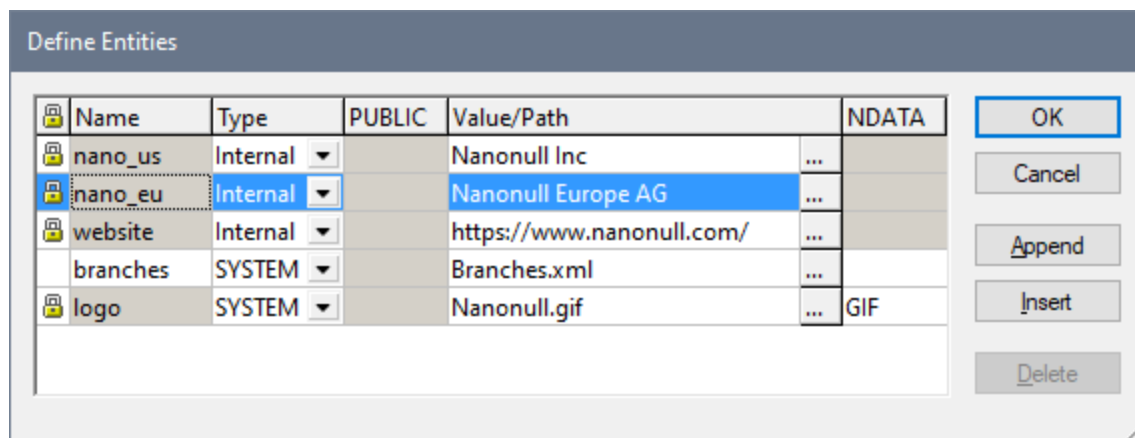
You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the **Insert Entity** submenu of the context menu. When you double-click on an entity in the Entities Entry Helper,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



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 is 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.

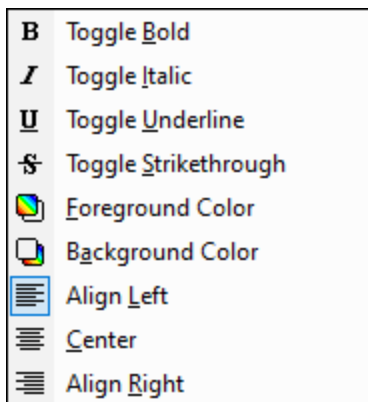
## 13.6.7 View Markup

The **View Markup** command has a submenu with options to control the display of markup in Authentic View. These options are described below.

	<b>Hide Markup</b> hides all markup symbols.
	<b>Show Small Markup</b> shows markup as small symbols.
	<b>Show Large Markup</b> shows markup as large symbols.
	<b>Show Mixed Markup:</b> The person who designs the StyleVision Power Stylesheet can specify either large markup, small markup, or no markup for individual elements/attributes of the document. Mixed markup shows this customized markup.

## 13.6.8 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.







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.

## 13.6.9 Append/Insert/Duplicate/Delete Row

The **Table Row** commands listed below enable you to structure tables in Authentic View.



	<b>Append Row</b> appends a row to the current table.
	<b>Insert Row</b> inserts a row in the current table .
	<b>Duplicate Row</b> duplicates the current table row below the current row.
	<b>Delete Row</b> deletes the current table row.

### 13.6.10 Collapse/Expand Markup

This command becomes enabled when Authentic markup has been switched on (see [View Markup](#)<sup>224</sup>) and a node's markup tag has been selected. Clicking the command when the node is expanded collapses the node. Clicking the command when the node is collapsed expands the node.

### 13.6.11 Move Row Up/Down

The **Table Row** commands listed below enable you to move rows within Authentic View tables.

- **Move Row Up** moves the current table row up by one row in Authentic View.
- **Move Row Down** moves the current table row down by one row in Authentic View.

### 13.6.12 Generate HTML, RTF, PDF, Word 2007+ Document

These buttons are enabled when a PXF file is opened in Authentic View. They 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**
- **Generate a Text Document**

They are also available in the Portable XML Form (PXF) toolbar (*screenshot below*).

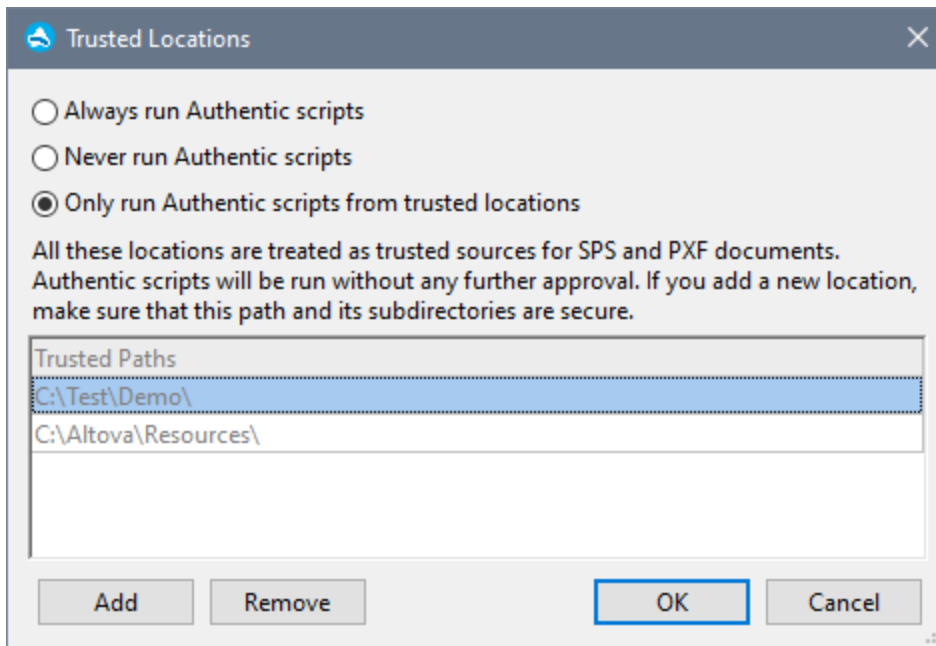


Clicking the individual command or buttons generates HTML, RTF, PDF, or DocX output, respectively.

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 Text, PDF and DocX (Word 2007+) output will be disabled.

### 13.6.13 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**.

## 13.7 View Menu

The **View** menu (*screenshot below*) controls the display of the active [Main window](#)<sup>15</sup> and allows you to change the way the document is displayed. This section provides a description of commands in the **View** menu.

### 13.7.1 Authentic View

This command switches the current document to [Authentic View](#)<sup>39</sup>.

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).

### 13.7.2 Browser View

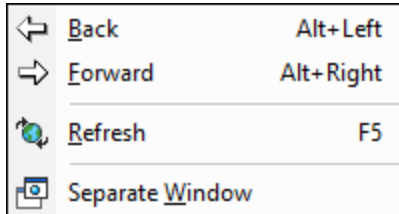


This command switches the current document to [Browser View](#)<sup>88</sup>. An XML-enabled browser renders the XML document using information from available CSS and/or XSL stylesheets.

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](#)<sup>254</sup> (**Tools | Options**) is checked. For more information, see the [Browser View](#)<sup>88</sup> section of this documentation.

## 13.8 Browser Menu

The **Browser** menu commands are enabled in [Browser View](#)<sup>88</sup> only.



### Back, Forward

The **Back** command (*shortcut: Alt + Left arrow*) 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.

The **Forward** command (*shortcut: Alt + Right arrow*) moves you forward through previously viewed pages in Browser View.

### 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).

### Separate Window

The **Separate Window** command is enabled in Browser View and undocks Browser View from the application window. 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 application window, make the Browser View window active and click the **Separate Window** command.

## 13.9 Tools Menu

The Tools menu allows you to:

- Check the [spelling](#) <sup>229</sup> of your XML documents
- Access the [scripting environment](#) <sup>283</sup> of Authentic Desktop. You can create, manage and store your own forms, macros and event handlers
- [View](#) <sup>235</sup> the currently assigned macros
- [Define and use global resources](#) <sup>235</sup>
- Access [Schema Manager](#) <sup>131</sup>, which enables you to install and manage the schemas you want to work with.
- Access customized commands that use external applications. These commands can be created in the [Tools tab of the Customize dialog](#) <sup>241</sup>.
- [Customize](#) <sup>237</sup> your version of Authentic Desktop: define your own toolbars, keyboard shortcuts, menus, and macros
- Define global Authentic Desktop [settings](#) <sup>253</sup>

### 13.9.1 Spelling

Authentic Desktop's spellchecker with built-in language dictionaries (*see note below*) is enabled in Authentic View.

**Note:** The built-in dictionaries that ship with Altova software do not indicate any language preferences by Altova. The selection of dictionaries is 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>. You should choose the dictionaries you want to use on the basis of their license and their usefulness to you.

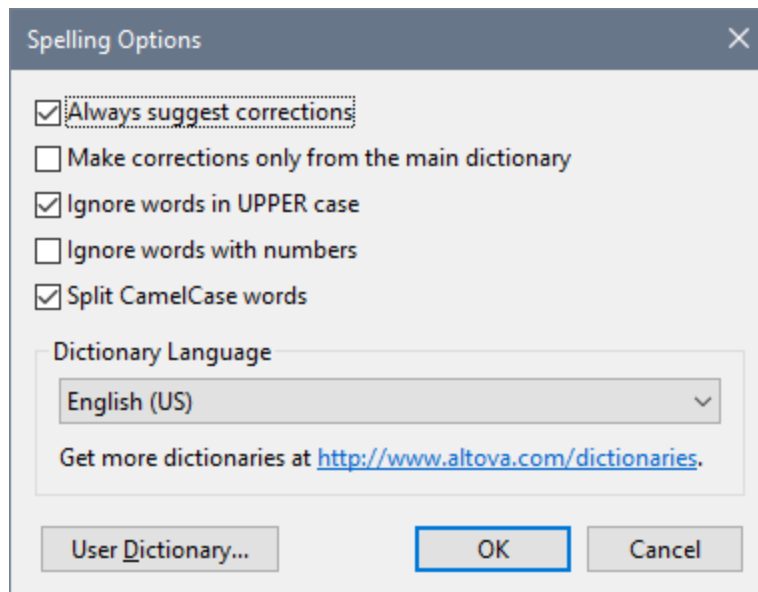
This section describes how to use the spellchecker. It is organized into the following sub-sections:

- [Select the spellchecker language](#) <sup>229</sup>
- [Run the spelling check](#) <sup>230</sup>

#### Select the spellchecker language

The spellchecker language can be set as follows:

1. Click the **Tools | Spelling Options** menu command.
2. In the Spelling Options dialog that appears (*screenshot below*), select one of the installed dictionaries from the dropdown list of the Dictionary Language combo box.

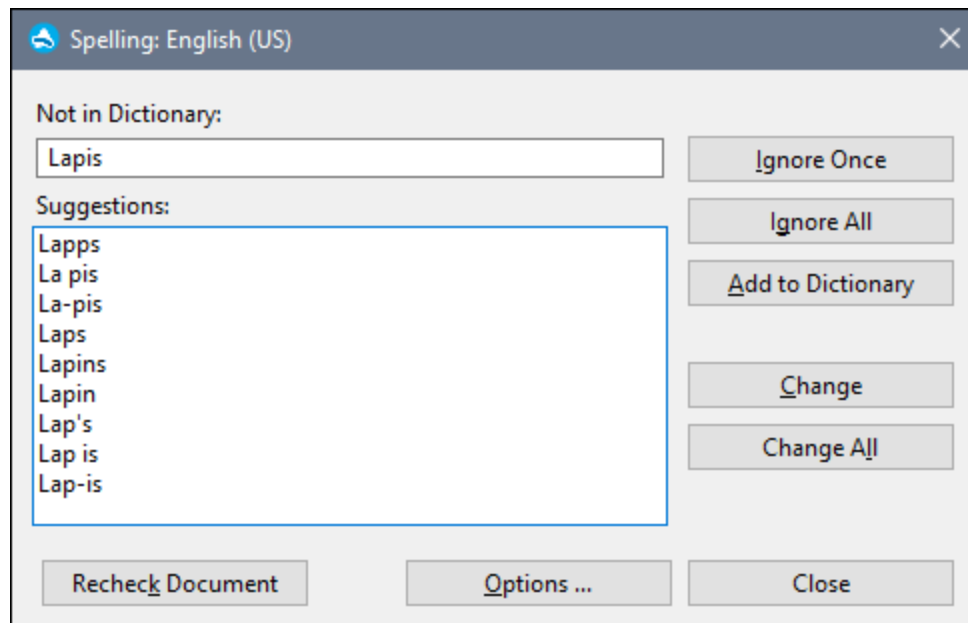


3. Click **OK** to finish.

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](#)<sup>233</sup>.

### Run 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.



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 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](#) <sup>234</sup> 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

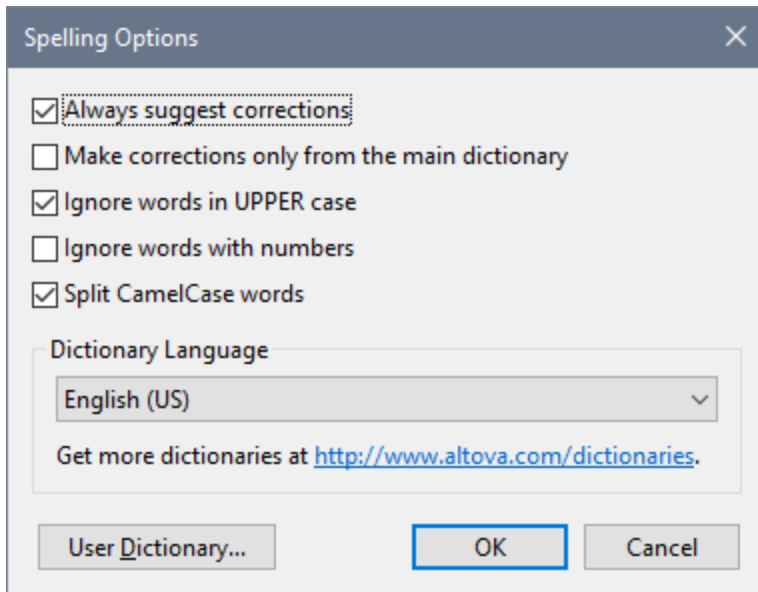
Clicking the **Options** button opens the [Spelling Options](#) <sup>232</sup> dialog box.

#### Close

This command closes the Spelling dialog box.

## 13.9.2 Spelling Options

The **Tools | Spelling Options** command opens the Spelling Options dialog (*screenshot below*), which 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](http://www.altova.com/dictionaries).



## 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 a 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:

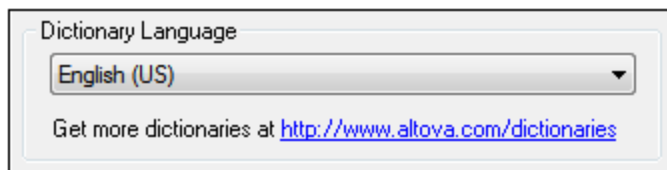
```
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.aff
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.aff
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (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:

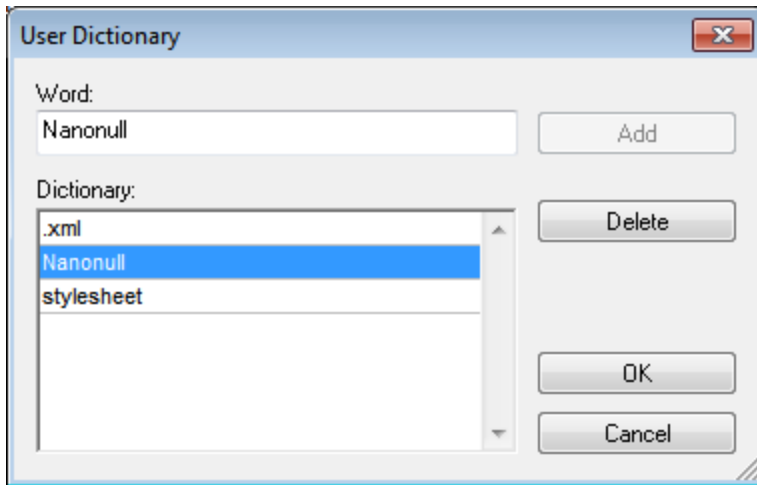
- By adding Hunspell dictionaries into a new subfolder of the `Lexicons` folder. Hunspell dictionaries can be downloaded, for example, from <https://wiki.openoffice.org/wiki/Dictionaryes> or <http://extensions.services.openoffice.org/en/dictionaries>. (Note that OpenOffice uses the zipped `OXT` format. 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 spellcheck, 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](#)<sup>229</sup> 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\\Documents\Altova\SpellChecker\Lexicons\user.dic

### 13.9.3 Scripting Editor

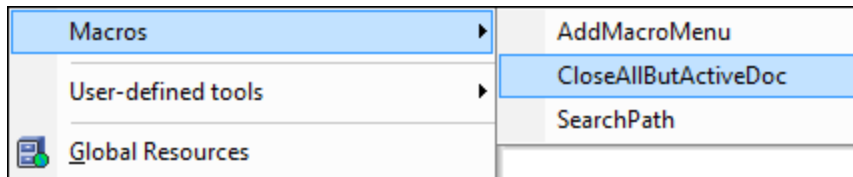
The **Scripting Editor** command opens the Scripting Editor window. How to work with the Scripting Editor is described in the [Scripting section](#)<sup>283</sup> of this documentation.

**Note:** For the Scripting Editor to run, .NET Framework version 2.0 or higher must be installed on your machine.

## 13.9.4 Macros

Run a macro as follows:

1. Placing your cursor over the **Macros** command rolls
2. The submenu that rolls out contains a list of macros in the Scripting Project that is currently active in Authentic Desktop (*screenshot below*). The active Scripting Project is specified in the [Scripting settings of the Options dialog](#)<sup>265</sup>.



3. Click a macro to run it.

## 13.9.5 User-Defined Tools

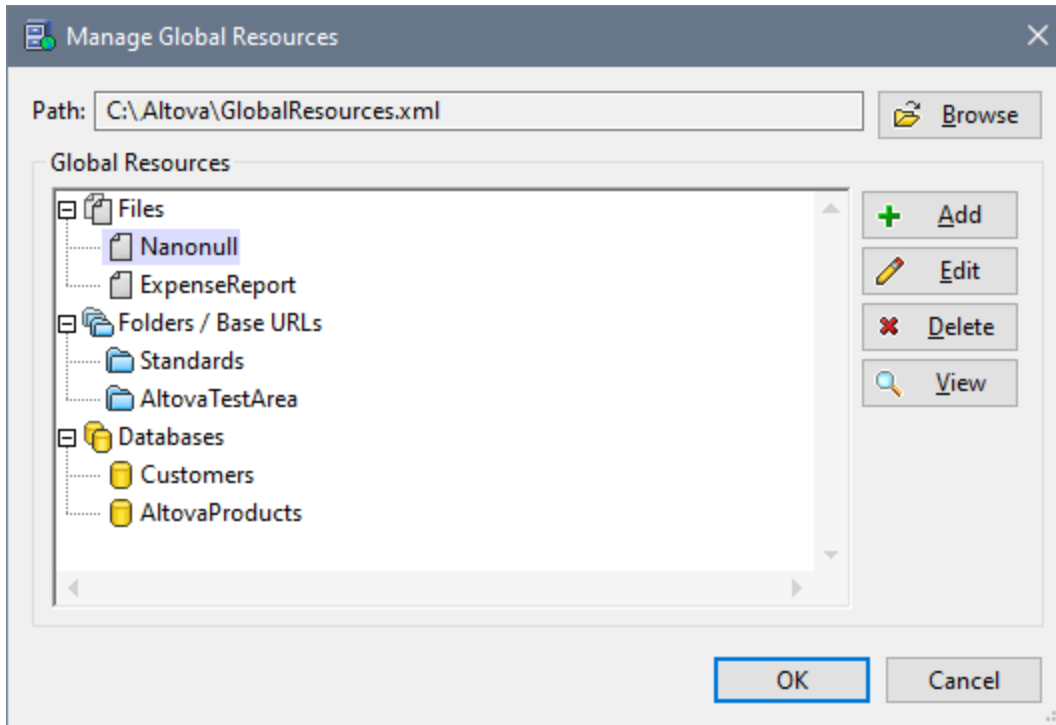
Placing the cursor over the **User-defined Tools** command rolls out a sub-menu containing custom-made commands that use external applications. You can create these commands in the [Tools tab of the Customize dialog](#)<sup>241</sup>. Clicking one of these custom commands executes the action associated with this command.

The **User-Defined Tools | Customize** command opens the [Tools tab of the Customize dialog](#)<sup>241</sup> (in which you can create the custom commands that appear in the menu of the **User-Defined Tools** command.)

## 13.9.6 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.

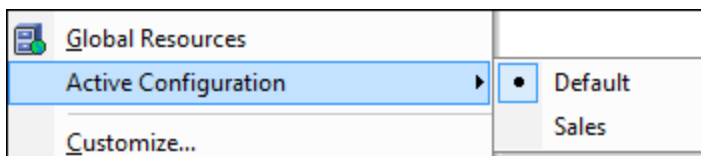


How to define global resources is described in detail in the section, [Defining Global Resources](#)<sup>90</sup>.

**Note:** The Altova Global Resources dialog can also be accessed via the [Global Resources toolbar](#)<sup>239</sup> (**Tools | Customize | Toolbars | Global Resources**).

## 13.9.7 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](#)<sup>235</sup> (screenshot below).



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](#)<sup>239</sup> (**Tools | Customize | Toolbars | Global Resources**).

## 13.9.8 XML Schema Manager

This command opens the XML Schema Manager dialog, from where you can install and manage your schemas for Altova products. See the [XML Schema Manager](#)<sup>131</sup> section for a description of how to use Schema Manager.

## 13.9.9 Customize

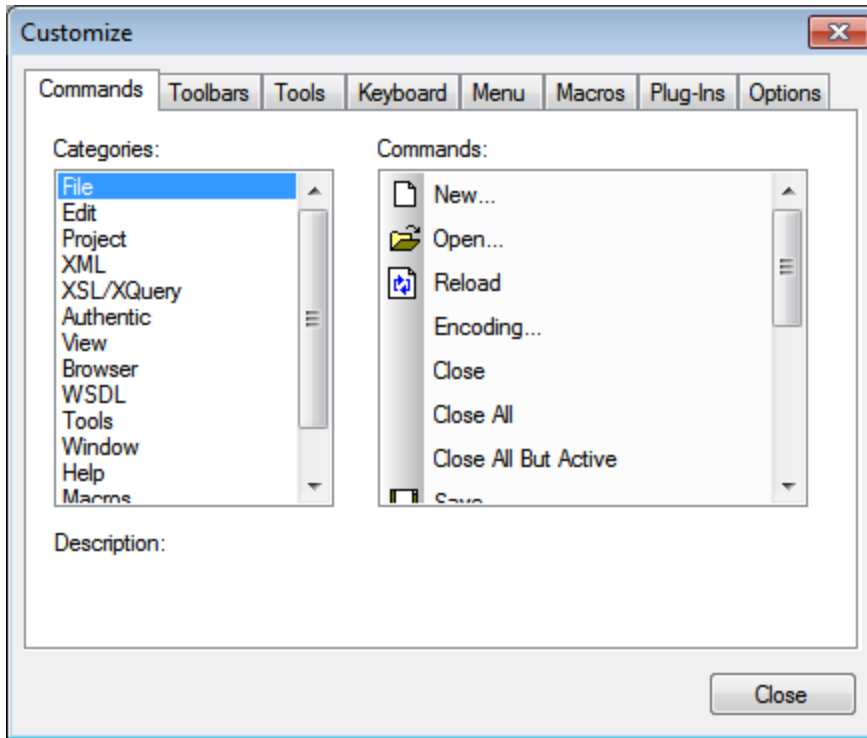
The **Customize** command lets you customize application menus and toolbars to suit your personal needs. Clicking the command pops up the Customize dialog, which has the following tabs:

- [Commands](#)<sup>237</sup>: All application and macro commands can be dragged from this tab into menu bars, menus and toolbars.
- [Toolbars](#)<sup>239</sup>: Toolbars can be activated, deactivated, and reset individually.
- [Tools](#)<sup>229</sup>: Commands that open external programs from within the interface can be added to the interface.
- [Keyboard](#)<sup>242</sup>: Keyboard shortcuts can be created for individual application and macro commands.
- [Menu](#)<sup>245</sup>: Menu bars and context menus to be customized are selected and made active in this tab. Works together with the Commands tab.
- [Macros](#)<sup>246</sup>: Macros can have new commands associated with them.
- [Plug-ins](#)<sup>248</sup>: Plug-ins can be activated and integrated in the interface.
- [Options](#)<sup>253</sup>: Display options for toolbars are set in this tab.

This section also describes the [context menu](#)<sup>250</sup> that appears when the Customize dialog is open and menu bar, menu, or tool bar items are right-clicked.

### 13.9.9.1 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 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](#)<sup>239</sup>, 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**<sup>245</sup> **tab**<sup>245</sup>.

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.

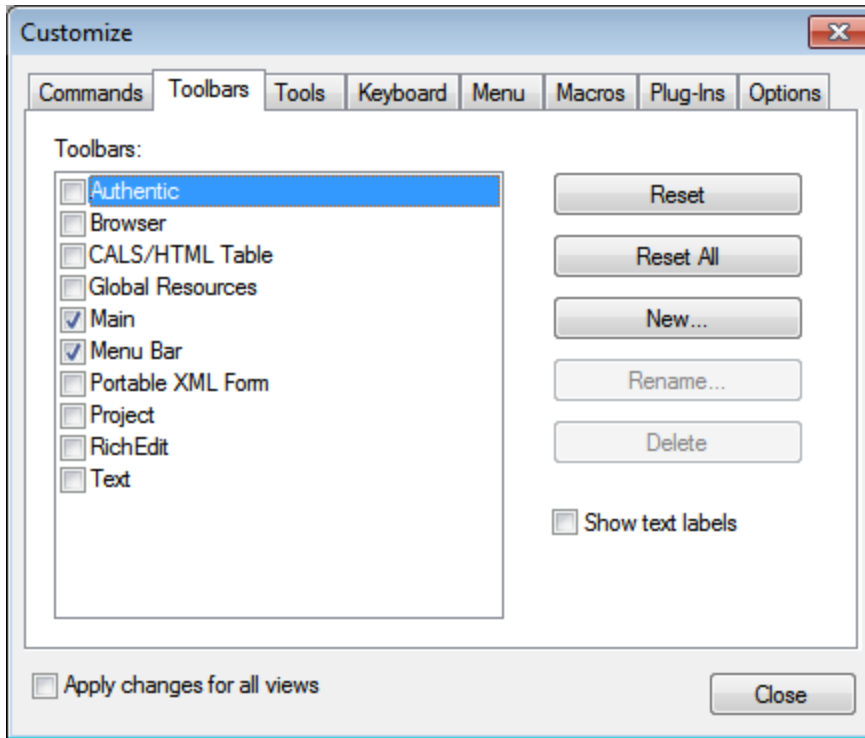
## 13.9.9.2 Toolbars

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](#)<sup>237</sup> 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](#)<sup>237</sup> 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.



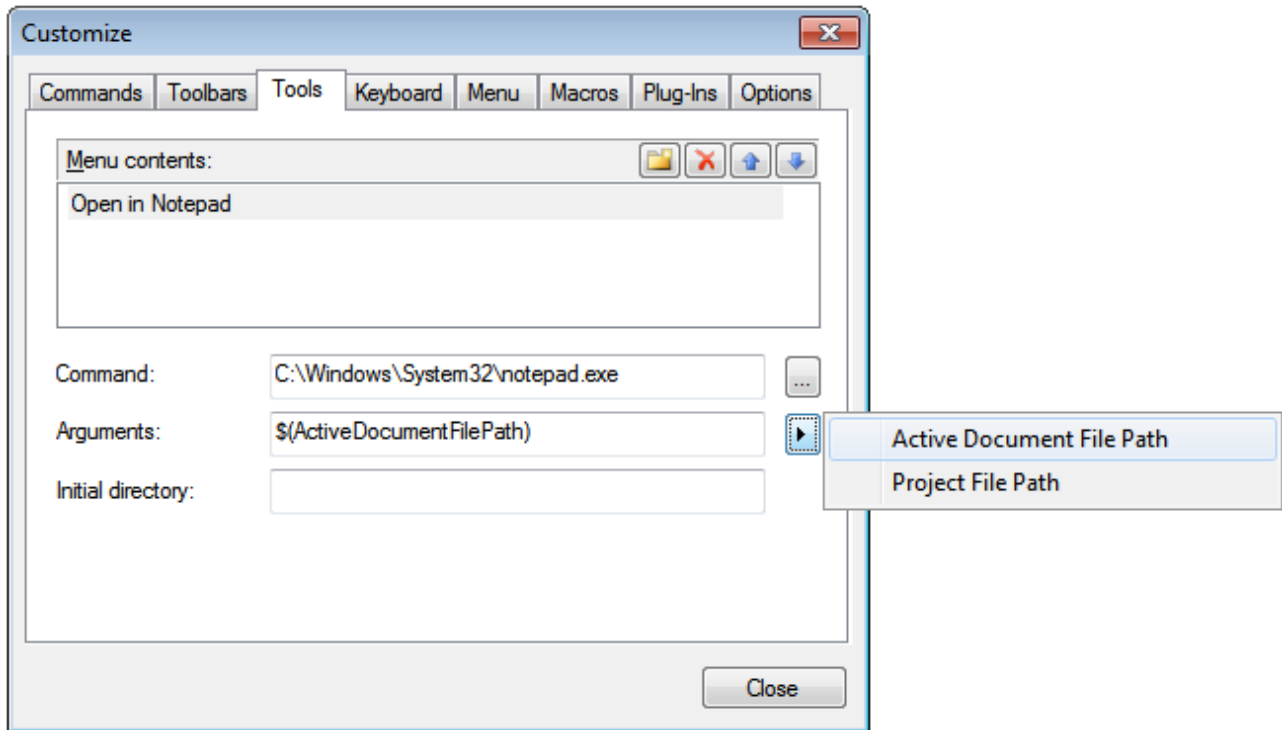
The following functionality is available:

- *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](#)<sup>237</sup> 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.



### 13.9.9.3 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.

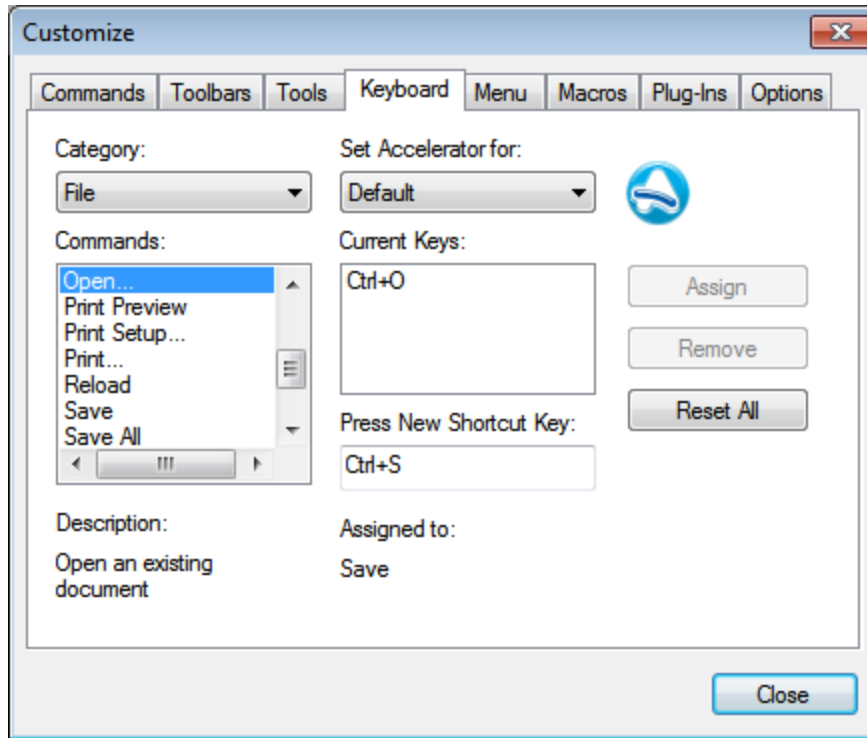
- *Active Document File Path*: The command in the **Tools | User-defined Tools** menu opens the document that is active in Authentic Desktop in the external application. The command in the context menu of a file in the Project window opens the selected file in the external application.
- *Project File 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.

## 13.9.9.4 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](#)<sup>246</sup>, 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](#)<sup>274</sup>).

### Function-key shortcuts (incl. for validation and transformation)

F1	Help Menu
F1 + Alt	Open Last File
F3	Find Next
F4 + CTRL	Close Active Window
F4 + Alt	Close Authentic Desktop
F5	Refresh
F6 + CTRL	Cycle through Open Windows
F7	Check Well-formedness
F8	Validate
F10	XSL Transformation
F10 + CTRL	XSL:FO Transformation

### File and Application commands

Alt + F1	Open Last File
CTRL + O	File Open
CTRL + N	File New
CTRL + P	File Print
CTRL + S	File Save
CTRL + F4	Close Active Window
CTRL + F6	Cycle through Open Windows
CTRL + TAB	Switch between Open Documents
Alt + F4	Close Authentic Desktop

### Miscellaneous keys

Up/Down Arrow Keys	Move Cursor or Selection Bar
Esc	Abandon Edits or Close Dialog Box
Return	Confirm Selection
Del	Delete Character or Selected
Shift + Del	Cut

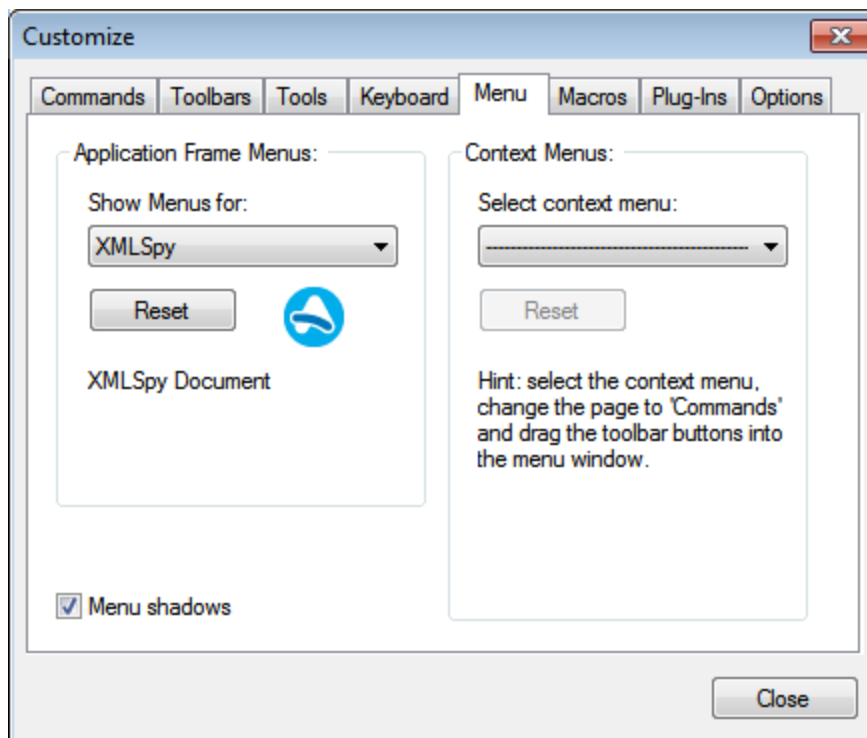
### Editing commands

CTRL + A	Select All
----------	------------

CTRL + F	Find
CTRL + G	Go to Line/Char
CTRL + H	Replace
CTRL + V	Paste
CTRL + X	Cut
CTRL + Y	Redo
CTRL + Z	Undo

### 13.9.9.5 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](#)<sup>237</sup> 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, do the following:

1. In the Application Frame Menus pane, select either *Default* (which shows available menus when no document is open) or *Authentic* (which shows available menus when one or more documents are open).
2. With the Customize dialog open, select (i) the menu you want to delete from the application's menu bar, or (ii) the command you want to delete from one of these menus.
3. Either (i) drag the menu from the menu bar or the menu command from the menu, or (ii) right-click the menu or menu command and select **Delete**.

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](#)<sup>237</sup>.
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.

### 13.9.9.6 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](#)<sup>237</sup>. As application commands, they can also be assigned shortcuts in the [Keyboard tab of the Customize dialog](#)<sup>242</sup>.

## How macros work in Authentic Desktop

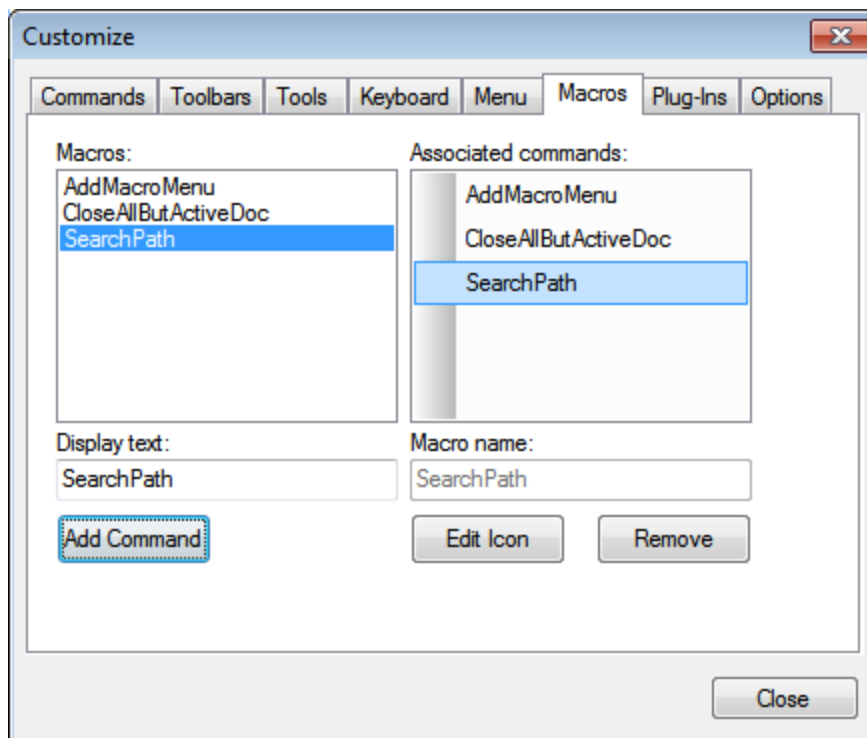
Macros in Authentic Desktop work as follows:

- Altova scripting projects (.asprj files) are created in Authentic Desktop's [Scripting Editor](#)<sup>283</sup>. 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](#)<sup>265</sup>, and (ii) The scripting project of the active [Authentic Desktop project](#)<sup>17</sup>, which is specified in the [Script Settings dialog](#)<sup>203</sup> ([Project | Script Settings](#)<sup>203</sup>).
- The macros in these two scripting projects are available in the application: in the **Tools | 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](#)<sup>283</sup> ([Tools | Scripting Editor](#)<sup>283</sup>) 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](#))<sup>265</sup> or the active application project's scripting project (via the application project's [Script Settings dialog](#)<sup>203</sup> ([Project | Script Settings](#)<sup>203</sup>)). 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:

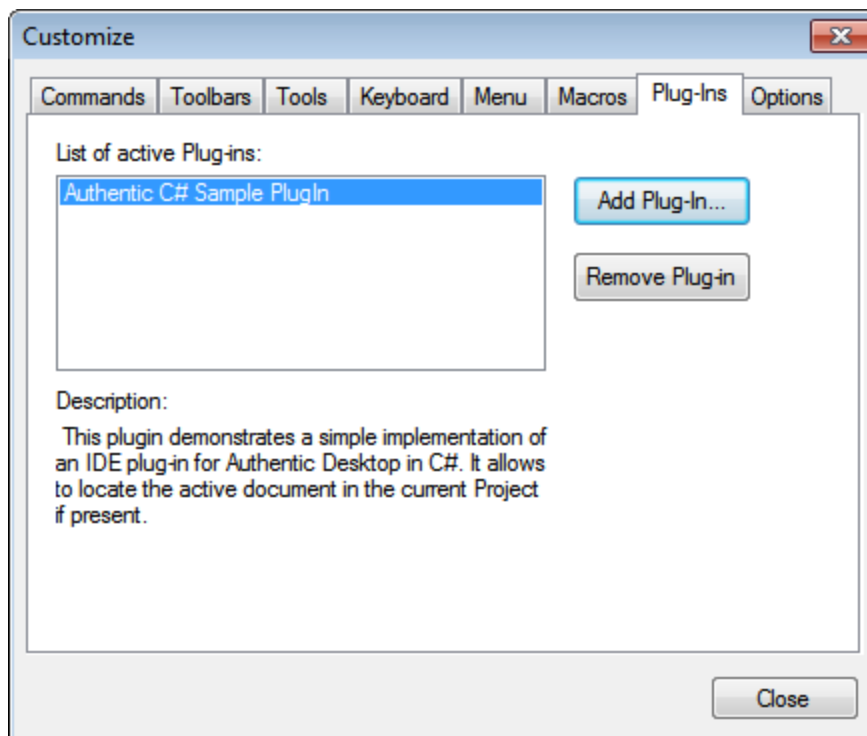
- Drag the command from the Associated Commands list box to the desired location in the menu or toolbar.
- Use the mechanisms available in the [Commands tab of the Customize dialog](#)<sup>237</sup>.

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](#)<sup>242</sup>. In the [Keyboard tab of the Customize dialog](#)<sup>242</sup>, 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.

### 13.9.9.7 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

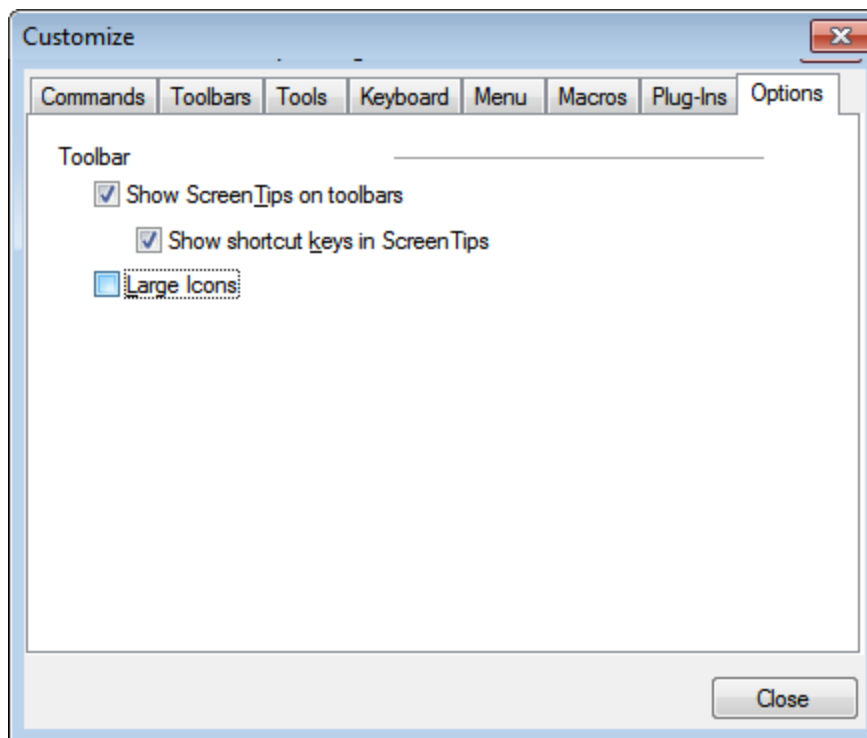
Source code for sample plug-ins has been provided in the application's [\(My\) Documents folder](#)<sup>11</sup>: `Examples\IDEPlugin` folder. To build a plug-in from such source code, do the following:

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 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](#)<sup>310</sup>.

### 13.9.9.8 Options

The **Options** tab allows you to define general environment settings.



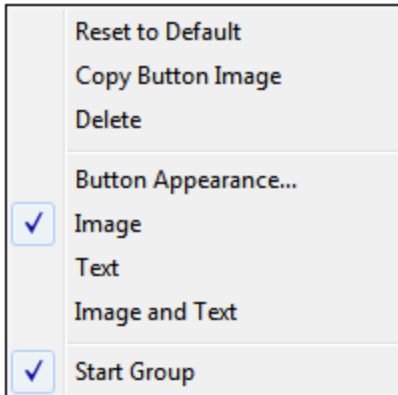
Click the check boxes to toggle on the following options:

- *Show Screen Tips 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 Screen Tips*: Defines whether shortcut information will be shown in screen tips.

- *Large icons*: Toggles the size of toolbar icons between standard and large.

### 13.9.9.9 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, a menu command, or a toolbar icon.

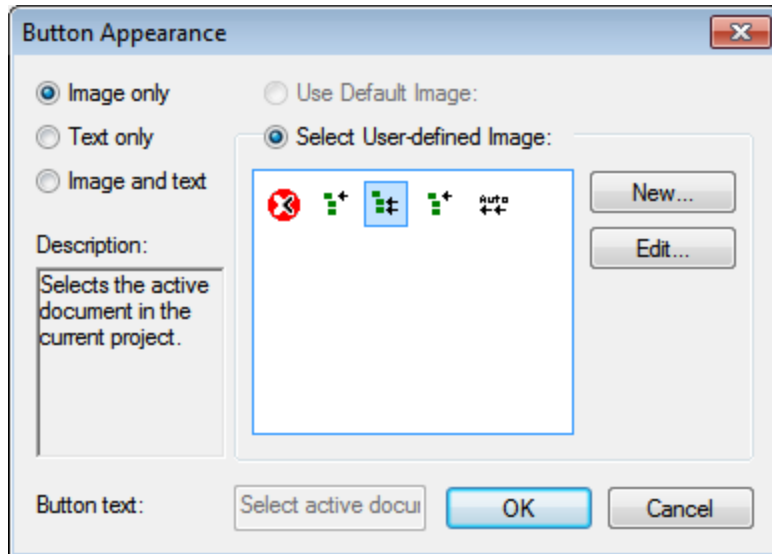


The following functionality is available:

- *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.

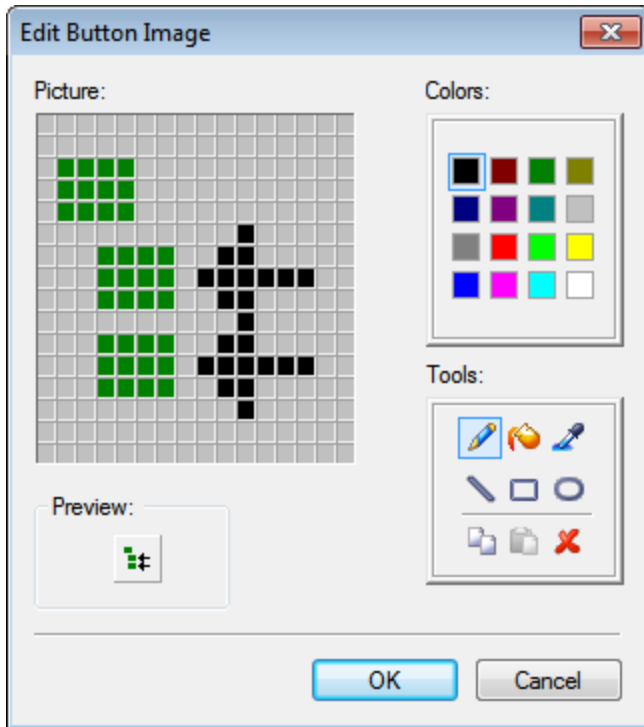
### The Button Appearance dialog

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.

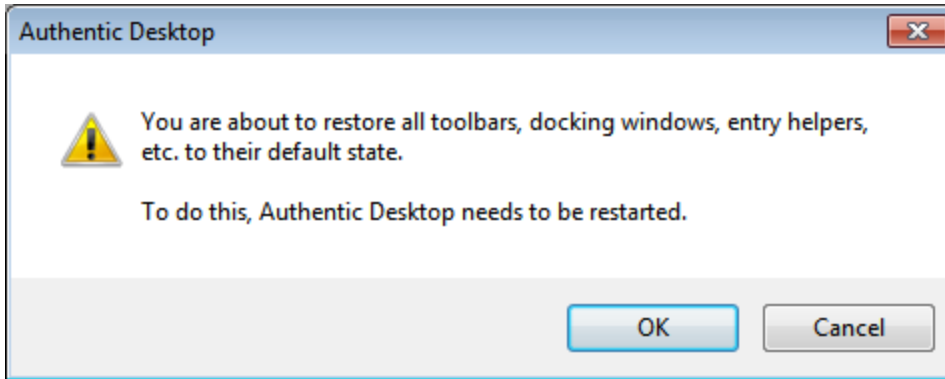
### Restoring deleted menus, menu commands, and toolbar icons

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](#) <sup>245</sup>, and click the **Reset** button in the *Application Frame Menus* pane. Alternatively, go to [Tools | Customize | Toolbars](#) <sup>239</sup>, select *Menu Bar*, and click the **Reset** button.
- *Menu commands:* Go to [Tools | Customize | Commands](#) <sup>237</sup>, and drag the command from the *Commands* list box into the menu.
- *Toolbar icons:* Go to [Tools | Customize | Commands](#) <sup>237</sup>, and drag the command from the *Commands* list box into the toolbar.

### 13.9.10 Restore Toolbars and Windows

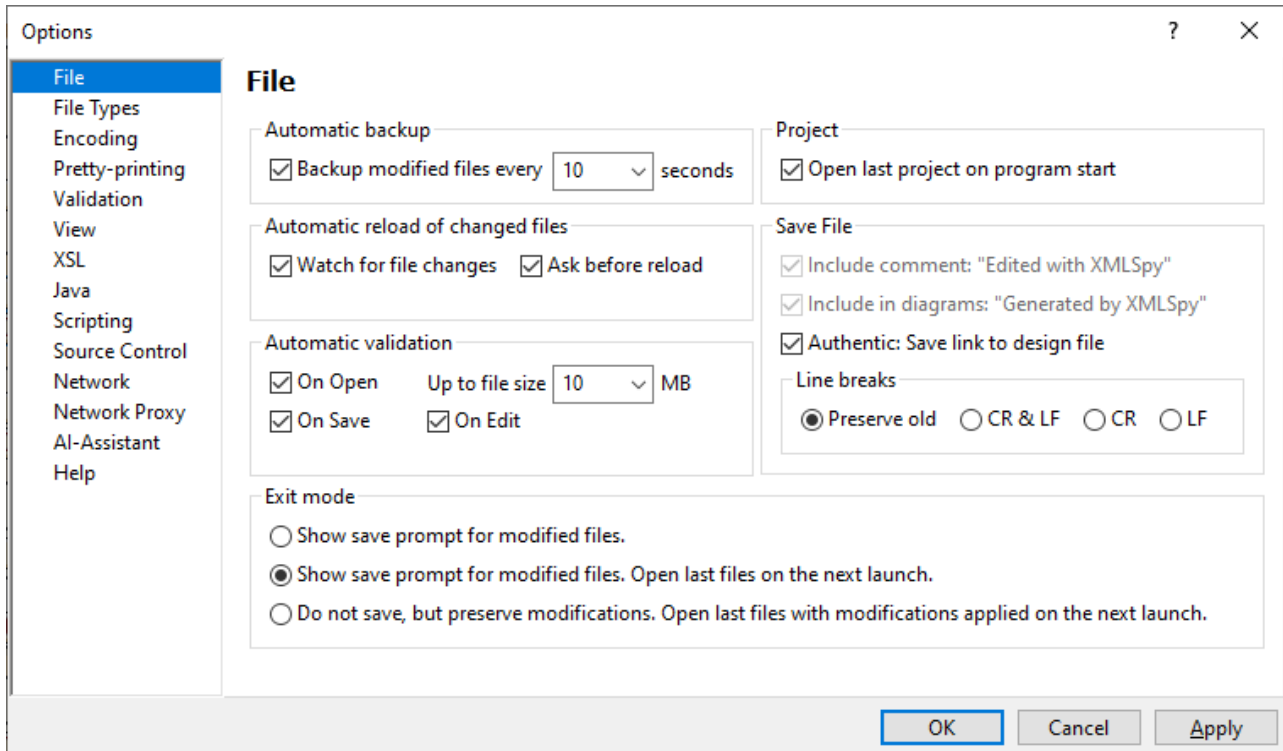
The **Restore Toolbars and Windows** command closes down Authentic Desktop and re-starts it with the default settings. Before it closes down a dialog pops up asking for confirmation about whether Authentic Desktop should be closed (*screenshot below*).



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.

### 13.9.11 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](#)<sup>254</sup> (*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.

### 13.9.11.1 File

The **File** section defines the way Authentic Desktop opens and saves documents. Related settings are in the [Encoding section](#) <sup>258</sup>.

### File

**Automatic backup**

Backup modified files every  seconds

**Automatic reload of changed files**

Watch for file changes  Ask before reload

**Automatic validation**

On Open Up to file size  MB

On Save  On Edit

**Exit mode**

Show save prompt for modified files.

Show save prompt for modified files. Open last files on the next launch.

Do not save, but preserve modifications. Open last files with modifications applied on the next launch.

**Project**

Open last project on program start

**Save File**

Include comment: "Edited with XMLSpy"

Include in diagrams: "Generated by XMLSpy"

Authentic: save link to design file

**Line breaks**

Preserve old  CR & LF  CR  LF

## Automatic backup

Files that you are currently editing will be automatically backed up if this option is enabled. You can select a backup frequency from between 5 seconds to 60 seconds in the combo box or enter a custom value up to 300 seconds. For more information, see the section [Automatic Backup of Files](#)<sup>56</sup>.

## 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.

## Automatic Validation

If you are using DTDs or XML Schemas to define the structure of your XML documents, you can automatically validate your instance documents in the following situations:

- On opening the file if the file has a size below a size you specify in MB
- On saving the file
- While editing the file.

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.

## 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).

## Exit mode

These options determine how to handle files that are open when Authentic Desktop is exited. The following options are available:

- *Show save prompt for modified files:* If an open file contains unsaved modifications, a prompt will appear asking whether you want to save the file modifications. Depending on your response, the file is saved or not saved, and the program is subsequently exited.
- *Show save prompt for modified files. Open last files on the next launch:* The Save dialog appears for open files that contain unsaved modifications. The user can save one or more modified files or not. When the program is relaunched after the exit, all the files that were open on exit will be opened on the relaunch. (If modifications had not been saved, then they would be lost.)
- *Do not save, but preserve modifications. Open last files with modifications applied on the next launch:* The program exits directly without saving unsaved modifications. On relaunch of the program, all files that were open on exit will be opened on relaunch, and they will contain the unsaved modifications. It would be as if you were continuing where you left off.

When you exit the program for the first time, the Exit Mode options are presented so that you can choose the exit behavior you want. Thereafter, the options are available in the File section of the Options dialog.

## Save and exit

After making the settings, click **OK** to finish.

## 13.9.11.2 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. Note that there are two special entries in the File Types list:

- `<default>` can be used to specify the treatment of files which have any extension that is not in the file-type list.
- `<none>` can be used to specify the treatment of files that have no extension at all.



## 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

The *Syntax Coloring* check box lets you set syntax-coloring on or off for different 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.

## 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.

### 13.9.11.3 Encoding

The **Encoding** section specifies options for file encodings.

**Encoding**

Default encoding for new XML files

Unicode UTF-8

Little-endian byte order

Big-endian byte order

Open XML files with unknown encoding as

Unicode UTF-8

Open non-XML files in

Codepage 1252 (Western)

BOM

Always create BOM if not UTF-8

Preserve detected BOM on saving

#### Default encoding for new XML files

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](#)<sup>166</sup> 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](#)<sup>166</sup> 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.

### 13.9.11.4 Pretty Printing

The **Pretty Printing** section (see screenshots below) enables you to specify how text is displayed in the XML document represented in Authentic View. Although Authentic Desktop does not provide a text view of the XML document, the settings that you choose here will be applied to the raw text of the XML file that you are editing in Authentic View.

The settings are:

- For whether to use tabs or spaces for pretty printing.
- For how empty elements are to be written. A self-closing element is one in which the opening and ending tag are combined in one, like this: `<element/>` or `<element />`.

## Save and exit

After making the settings, click **OK** to finish.

### 13.9.11.5 Validation

The **Validation** section enables you to specify options for validating XML documents.

#### Validation

XML

Cache DTD/Schema files in memory

Schema Version

v1.1 if <xs:schema vc:minVersion="1.1" ... >  
v1.0 otherwise

Always v1.1

Always v1.0

Message limits

Errors:

Inconsistencies:

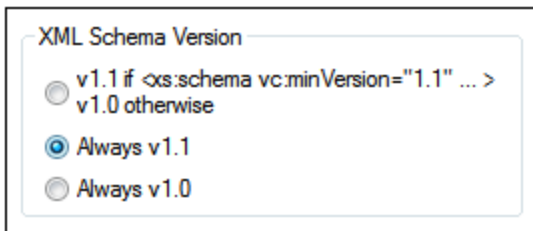
Warnings:

## XML

Authentic Desktop can cache DTD and XML Schema files in memory to save unnecessary reloading (for example, when the schema is not local but is accessed via a URL). Note, however, that if you use cached versions of schemas, changes you make to your schema will not be immediately reflected when you validate; in this case, you would need to reload the XML file or restart Authentic Desktop.

### 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*).



The following situations are possible. *XML Schema Version* in the table below refers to the selection in the XML Schema Version pane shown above. The `vc:minVersion` values in the table refer to the value of the `xs:schema/@vc:minVersion` attribute in the XML Schema document.

XML Schema Version	<code>vc:minVersion</code> attribute	XSD mode
<i>Always v1.0</i>	Is absent, or is present with any value	1.0
<i>Always v1.1</i>	Is absent, or is present with any value	1.1
<i>Value of @vc:minVersion</i>	Attribute has value of 1.1	1.1
<i>Value of @vc:minVersion</i>	Attribute is absent, or attribute is present with a value other than 1.1	1.0

## Message limits

These options enable you to set separate limits for the number of errors, XBRL inconsistencies, and warnings that are displayed. The default number for each category is 100. Change it to the number you want.

## Save and exit

After making the settings, click **OK** to finish.

### 13.9.11.6 View

The **View** section enables you to customize the XML documents presentation in Authentic Desktop.

#### 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.

#### Browser engine

The browser engine that is used in Authentic View and Browser View is currently Internet Explorer (IE), and IE is therefore the default browser engine for these two views. Alternatively, you can use Microsoft Edge Web View 2 as the engine for Browser View. If Edge is not installed on your machine, go to the [WebView2 download page](#), from where you can install the Evergreen Bootstrapper. This will enable you to use Microsoft Edge WebView2 as the engine for Browser View.

See the topic [Browser View](#)<sup>88</sup> for more information.

#### Save and exit

After making the settings, click **OK** to finish.

### 13.9.11.7 XSL

The **XSL** section (*screenshot below*) enables you to define options for [XSLT transformations](#)<sup>262</sup> and [XSL-FO transformations](#)<sup>264</sup> carried out from within the application.

### XSL

Engine: Built-in RaptorXML XSLT engine ▼

Validate XML files used in transformation

---

Output file

Default file extension: .html

Reuse output window

Use file extension from `<xsl:output method="">` attribute if provided

---

XSL-FO transformation

Path to engine (if using FOP, path to fop.bat):

C:\ProgramData\Altova\SharedBetweenVersions\Apache FOP 2.7\fop.bat Browse...

For the XSLT part of the transformation use  selected XSLT engine  XSL-FO engine

## Engine settings

You can set up an XSLT processor to carry out XSLT transformations when the [XSLT Transformation](#)<sup>210</sup> command is invoked.

You can select one of the following XSLT engine options:

- Built-in RaptorXML XSLT engine
- Microsoft XML Parser (MSXML)
- External XSLT engine

**Note:** For XSLT debugging in Authentic Desktop, the built-in RaptorXML XSLT engine is always used—even if another XSLT engine is selected here for transformations.

### Altova RaptorXML XSLT Engine

Authentic Desktop contains the Altova RaptorXML XSLT 1.0, XSLT 2.0, and XSLT 3.0 engines, which you can use for XSLT 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). This applies both for XSLT transformations as well as for XSLT debugging using XMLSpy's XSLT/XQuery Debugger.

If you wish to validate the XML files used in transformations, select the *Validate* option (see screenshot above).

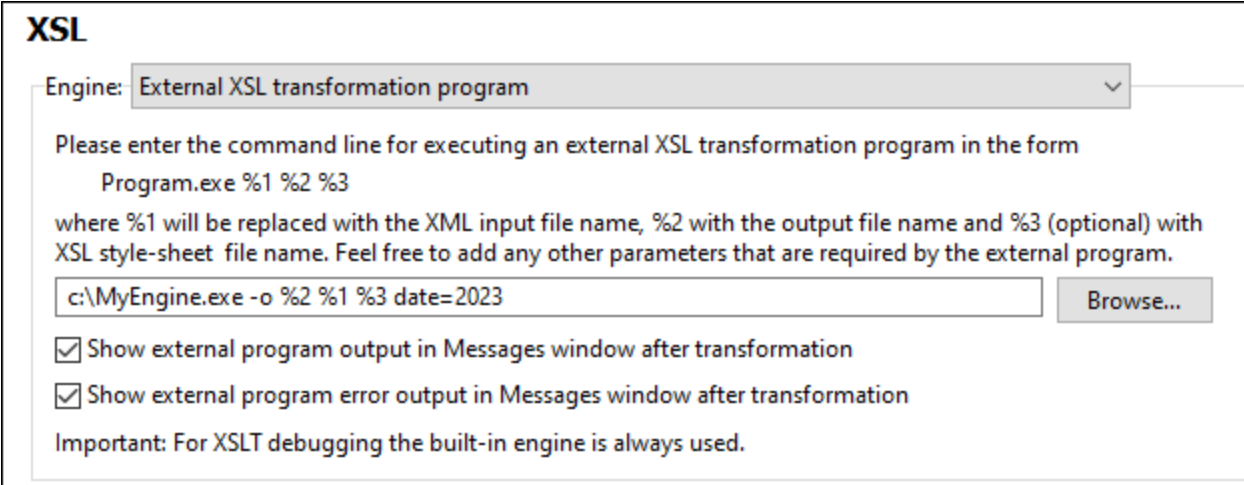
### Microsoft XML Parser (MSXML)

One or more of the MSXML 3.0, 4.0, or 6.0 parsers will be pre-installed on your machine. If you know which installed version you want to use, you could select it. Otherwise, you should let Authentic Desktop select the

version automatically. (The *Choose version automatically* option is active by default.) In this case, Authentic Desktop tries to select the most recent available version.

### External XSLT engine

Choose an external XSLT processor of your choice by entering the path to its executable file.



You must specify the command line string that the external XSLT processor uses to run a transformation. You can build the command line string with the following components:

- %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 pattern to run an XSLT transformation:

```
myxsltengine.exe -o <output.xml> <input.xml> <stylesheet.xslt> <param-name>=<param-value>?
```

Then, in Authentic Desktop, build the command line using the corresponding variables in the correct locations. For example:

```
c:\MyEngine.exe -o %2 %1 %3 date=2023
```

Authentic Desktop will send the correct input files to the external engine for processing and return the output file/s to an output location if one is specified and/or to an application window.

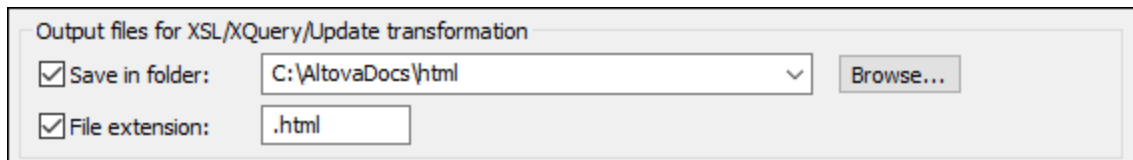
Check the respective check boxes to show the output and error messages of the external program in the Messages Window of Authentic Desktop.

**Note:** The parameters set in the [XSLT Input Parameters dialog](#)<sup>212</sup> (accessed via the **XSL** menu) are passed to the internal Altova XSLT Engines only. They are not passed to any other XSLT Engine that is set up as the default XSLT processor.

## Output File settings

The following options are available:

- *Default file extension:* Sets a default file extension for output files, which can be overridden by the file extension named in the XSLT element `xs1:output` (see *last list item*).
- *Reuse output window:* Causes subsequent transformations to display the result document in the same output window. If the input 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](#)<sup>204</sup> is **also** disabled.



- *Use file extension of xsl:output element:* Selects whether the file extension specified in the `xs1:output` element of the XSLT stylesheet would override the default extension specified in the first option of this list.

## 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](#)<sup>211</sup> 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](#)<sup>262</sup>), 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 Desktop).

## Save and exit

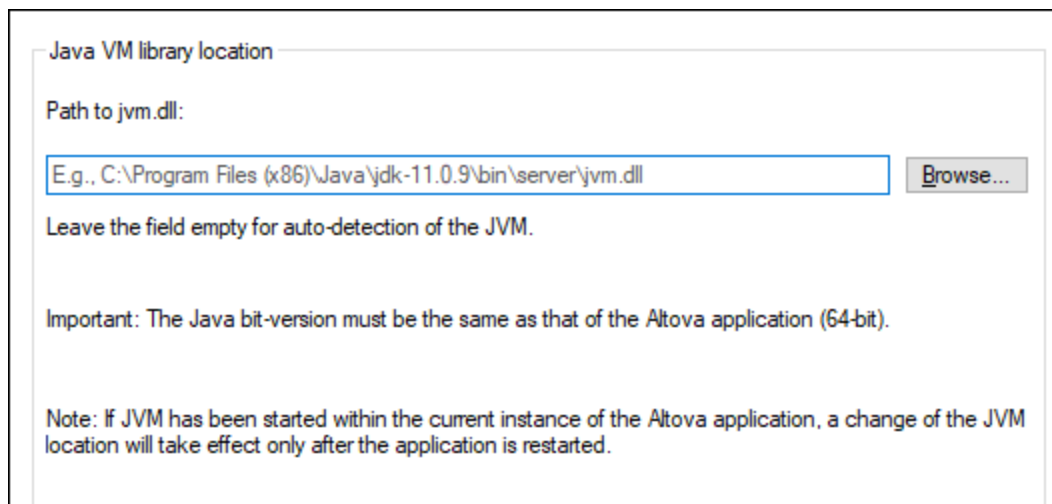
After making the settings, click **OK** to finish.



### 13.9.11.8 Java

In the *Java* section (see *screenshot below*), 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 in 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 (e.g., 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.

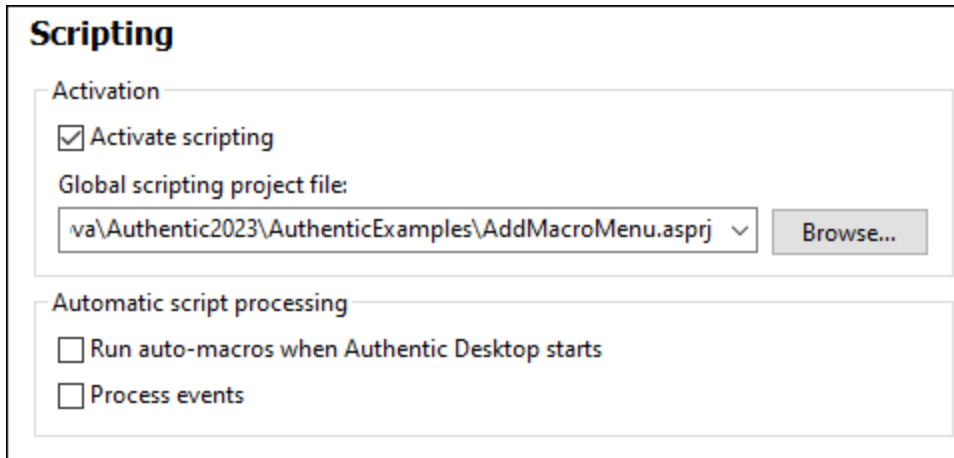


Note the following:

- 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.

### 13.9.11.9 Scripting

The **Scripting** section (*screenshot below*) allows you to enable the [Scripting Environment](#)<sup>234</sup> on application startup. Check the *Activate Scripting* check box to do this. You can then specify the Global Scripting Project file (*see screenshot below*).



The screenshot shows a dialog box titled "Scripting". It is divided into two sections. The first section, "Activation", contains a checked checkbox for "Activate scripting". Below it is a label "Global scripting project file:" followed by a text box containing the path "va\Authentic2023\AuthenticExamples\AddMacroMenu.asprj" and a "Browse..." button. The second section, "Automatic script processing", contains two unchecked checkboxes: "Run auto-macros when Authentic Desktop starts" and "Process events".

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.

## 13.9.11.10 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**

Current source control plug-in:  
Microsoft Visual SourceSafe

Logon ID (SourceSafe):  
MYFAVID

Perform background status updates every 500 ms  
 Display output messages from plug-in  
 Get everything when opening a project  
 Check in everything when closing a project  
 Don't show Check Out dialog box when checking out items  
 Don't show Check In dialog box when checking in items  
 Keep items checked out when checking in or adding items

If dialogs were hidden using Don't show this again, click Reset to view them again.

## 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:

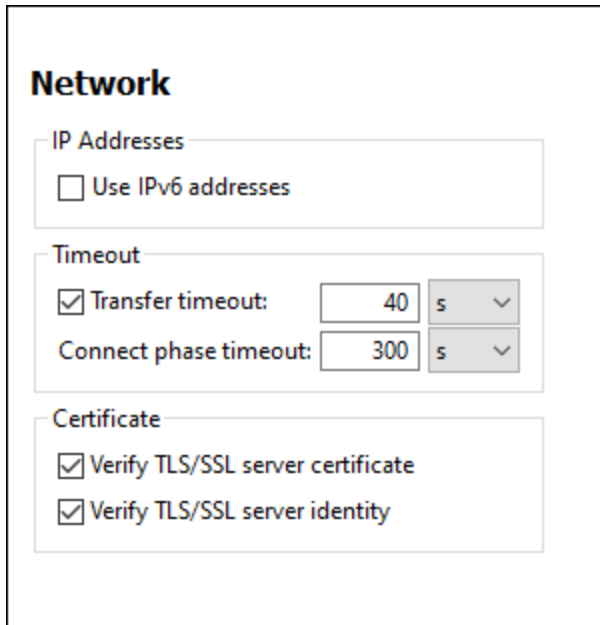
- Status updates can be performed in the background after a user-defined interval of time, or they can be switched off entirely. Very large source control databases could consume considerable CPU and network resources. The system can be speeded up, however, by disabling background status updates or increasing 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.

### 13.9.11.11 Network

The **Network** section (*screenshot below*) enables you to configure important network settings.



The screenshot shows a 'Network' configuration panel with three sections:

- IP Addresses:** A checkbox labeled 'Use IPv6 addresses' is currently unchecked.
- Timeout:** Two checked checkboxes are present: 'Transfer timeout' and 'Connect phase timeout'. The 'Transfer timeout' is set to 40 seconds, and the 'Connect phase timeout' is set to 300 seconds. Each has a dropdown menu for units, currently set to 's'.
- Certificate:** Two checked checkboxes are present: 'Verify TLS/SSL server certificate' and 'Verify TLS/SSL server identity'.

#### IP addresses

When host names resolve to more than one address in mixed IPv4/IPv6 networks, selecting this option causes the IPv6 addresses to be used. If the option is not selected in such environments and IPv4 addresses are available, then IPv4 addresses are used.

#### Timeout

- *Transfer timeout:* If this limit is reached for the transfer of any two consecutive data packages of a transfer (sent or received), then the entire transfer is aborted. Values can be specified in seconds [s] or milliseconds [ms], with the default being 40 seconds. If the option is not selected, then there is no time limit for aborting a transfer.
- *Connection phase timeout:* This is the time limit within which the connection has to be established, including the time taken for security handshakes. Values can be specified in seconds [s] or milliseconds [ms], with the default being 300 seconds. This timeout cannot be disabled.

#### Certificate

- *Verify TLS/SSL server certificate:* If selected, then the authenticity of the server's certificate is checked by verifying the chain of digital signatures until a trusted root certificate is reached. This option is enabled by default. If this option is not selected, then the communication is insecure, and attacks (for example, a man-in-the-middle attack) would not be detected. Note that this option does not verify that the certificate is actually for the server that is communicated with. To enable full security, both the certificate and the identity must be checked (*see next option*).
- *Verify TLS/SSL server identity:* If selected, then the server's certificate is verified to belong to the server we intend to communicate with. This is done by checking that the server name in the URL is the same as the name in the certificate. This option is enabled by default. If this option is not selected, then the server's identity is not checked. Note that this option does not enable verification of the server's certificate. To enable full security, both the certificate as well as the identity must be checked (*see*

*previous option*).

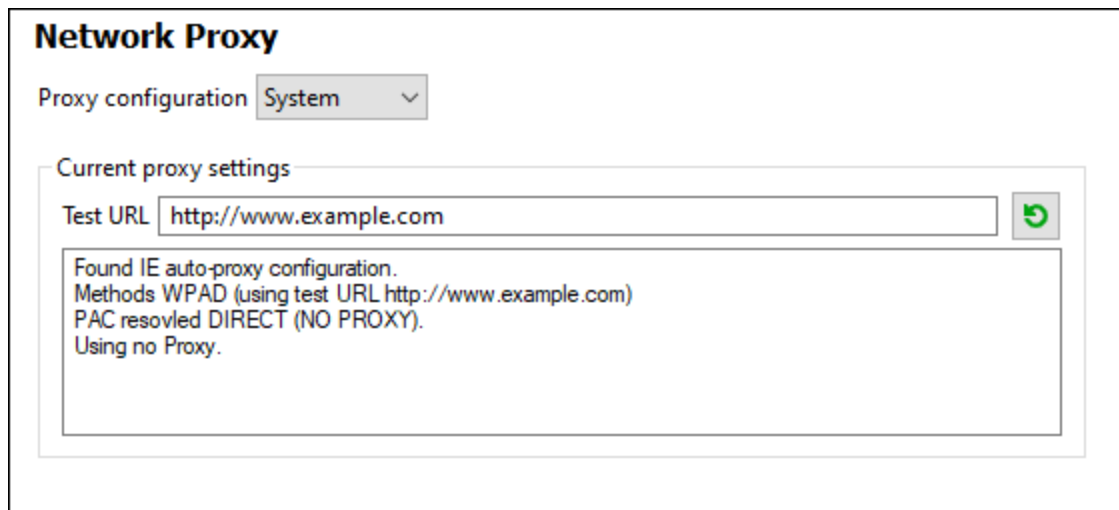
## Save and exit

After making the settings, click **OK** to finish.

### 13.9.11.12 Network Proxy

The *Network Proxy* section enables you to configure custom proxy settings. These settings affect how the application connects to the Internet (for XML validation purposes, for example). By default, the application uses the system's proxy settings, so you should not need to change the proxy settings in most cases. If necessary, however, you can set an alternative network proxy by selecting, in the *Proxy Configuration* combo box, either *Automatic* or *Manual* to configure the settings accordingly.

**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.

### Network Proxy

Proxy configuration Manual v

HTTP Proxy  Port

Use this proxy server for all protocols

SSL Proxy  Port

No Proxy for

Do not use the proxy server for local addresses

---

Current proxy settings

Test URL  ↻

(using test URL `http://www.example.com`)  
Using no Proxy.

The following options are provided:

- *HTTP Proxy*: Uses the specified host name and port for the HTTP protocol. If *Use this proxy server for all protocols* is selected, then the specified HTTP proxy is used for all protocols.
- *SSL Proxy*: Uses the specified host name and port for the SSL protocol.
- *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 *Automatic proxy configuration*).

### 13.9.11.13 AI-Assistant

Enter your OpenAI API key in the AI-Assistant options section. This enables you to use Authentic Desktop's AI-Assistant (accessible via the menu) directly—that is, without having to enter your OpenAI API key each time you open the assistant or make an OpenAI request.

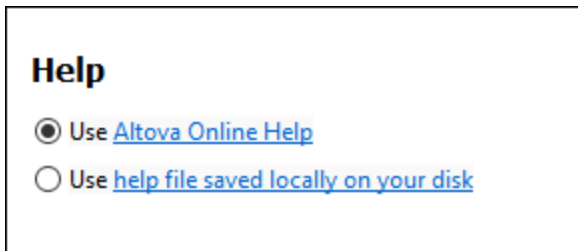
In order to create an OpenAI API key, you will need to first open an OpenAI account and then generate the key. Instructions for how to do this are given in the Options dialog.

### 13.9.11.14 Help

Authentic Desktop provides Help (the user manual) in two formats:

- Online Help, in HTML format, which is available at the Altova website. In order to access the Online Help you will need Internet access.
- A Help file in PDF format, which is installed on your machine when you install Authentic Desktop. It is named `Authentic Desktop.pdf` and is located in the application folder (in the Program Files folder). If you do not have Internet access, you can always open this locally saved Help file.

The Help option (*screenshot below*) enables you to select which of the two formats is opened when you click the **Help (F1)** command in the **Help** menu.



You can change this option at any time for the new selection to take effect. The links in this section (see *screenshot above*) open the respective Help format.

## 13.10 Window Menu

The **Window** menu contains commands that let you organize individual application and document windows within the GUI. You can cascade or tile open document windows, and you can arrange entry helper and output windows as well as hide them.

### Cascade, Tile Horizontally/Vertically

The **Cascade** command arranges document windows so that they are staggered in a sequence from back to forward.

The **Tile Horizontally** and **Tile Vertically** arranges the windows of open and non-minimized documents so that they are re-sized as tiles that are all visible within the application window.

### Project Window, Info Window, Entry Helpers, Output Windows

These commands switch the display of, respectively, the [Project Window](#)<sup>17</sup>, [Info Window](#)<sup>19</sup>, [Entry Helpers](#)<sup>19</sup>, and [Output Windows](#)<sup>19</sup> on or off.

Each of these windows is a dockable window. Dragging on the window's 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.

### Project and Entry Helpers

This command toggles on and off the display of the [Project Window](#)<sup>17</sup> and the [Entry Helpers](#)<sup>19</sup> together. It saves you the trouble of switching on/off the display of these windows individually.

### All On/Off

This command lets you switch all dockable windows (*listed below*) on or off.

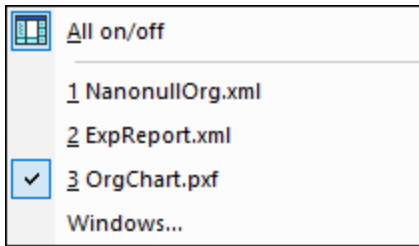
- [Project Window](#)<sup>17</sup>
- [Info Window](#)<sup>19</sup>
- [Entry Helpers](#)<sup>19</sup>
- [Output Windows](#)<sup>19</sup>

This is useful if you want to hide all non-document windows quickly, to get the maximum viewing area for the document/s you are working on.

### Currently open window list

This list shows all currently open windows, and lets you quickly switch between them.





You can also use **CTRL+F6** keyboard shortcuts to cycle through the open windows.

## 13.11 Help Menu

The **Help** menu contains commands to get help and information about Authentic Desktop, as well as links to information and support pages on the Altova web server.

The **Help** menu also contains the [Registration dialog](#)<sup>275</sup>, which lets you enter your license key-code once you have purchased the product.

### 13.11.1 Help

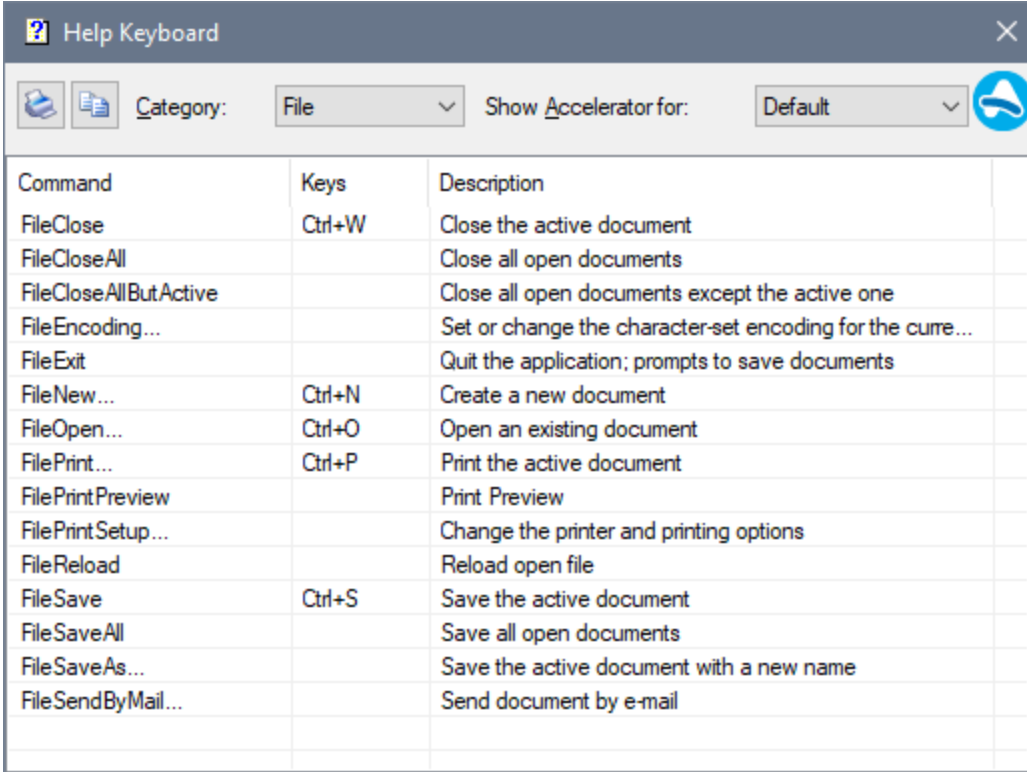
The **Help (F1)** command opens the application's Help documentation (its user manual). By default, the Online Help in HTML format at the Altova website will be opened.

If you do not have Internet access or do not want, for some other reason, to access the Online Help, you can use the locally stored version of the user manual. The local version is a PDF file named **Authentic Desktop.pdf** that is stored in the application folder (in the Program Files folder).

If you want to change the default format to open (Online Help or local PDF), do this in the Help section of the Options dialog (menu command **Tools | Options**).

### 13.11.2 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.



Command	Keys	Description
FileClose	Ctrl+W	Close the active document
FileCloseAll		Close all open documents
FileCloseAllButActive		Close all open documents except the active one
FileEncoding...		Set or change the character-set encoding for the curre...
FileExit		Quit the application; prompts to save documents
FileNew...	Ctrl+N	Create a new document
FileOpen...	Ctrl+O	Open an existing document
FilePrint...	Ctrl+P	Print the active document
FilePrintPreview		Print Preview
FilePrintSetup...		Change the printer and printing options
FileReload		Reload open file
FileSave	Ctrl+S	Save the active document
FileSaveAll		Save all open documents
FileSaveAs...		Save the active document with a new name
FileSendByMail...		Send document by e-mail

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.

### 13.11.3 Activation, Order Form, Registration, Updates

#### ☐ Software Activation

##### License your product

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 license.** 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 license. Click it to get your license. When you click this button, your machine-ID will be hashed and sent to Altova via HTTPS. The license information will be sent back to the machine via an HTTP response. If the license is created successfully, a dialog to this effect will appear in your Altova application. On clicking **OK** in this dialog, the software will be activated for a period of 30 days **on this particular machine**.
- **Permanent license key.** The **Software Activation** dialog allows you 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. Your license will be sent to you by e-mail in the form of a license file, which contains your license-data.

There are three types of permanent license: *installed*, *concurrent user*, and *named user*. An installed license unlocks the software on a single computer. If you buy an installed license for  $N$  computers, then the license allows use of the software on up to  $N$  computers. A concurrent-user license for  $N$  concurrent users allows  $N$  users to run the software concurrently. (The software may be installed on  $10N$  computers.) A named-user license authorizes a specific user to use the software on up to 5 different computers. To activate your software, click **Upload a New License**, and, in the dialog that appears, enter the path to the license file, and click **OK**.

**Note:** For multi-user licenses, each user will be prompted to enter his or her own name.

*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 file as an attachment. The license file has a `.altova_licenses` file extension.

To activate your Altova product, you can do one of the following:

- 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 a suitable location. In your Altova product, select the menu command **Help | Software Activation**, and then **Upload a New License**. Browse for or enter the path to the license file, and click **OK**.
- 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 (see *below*) or (ii) assign the license to the product from Altova LicenseServer. *For more information about licensing via LicenseServer, read the rest of this topic.*

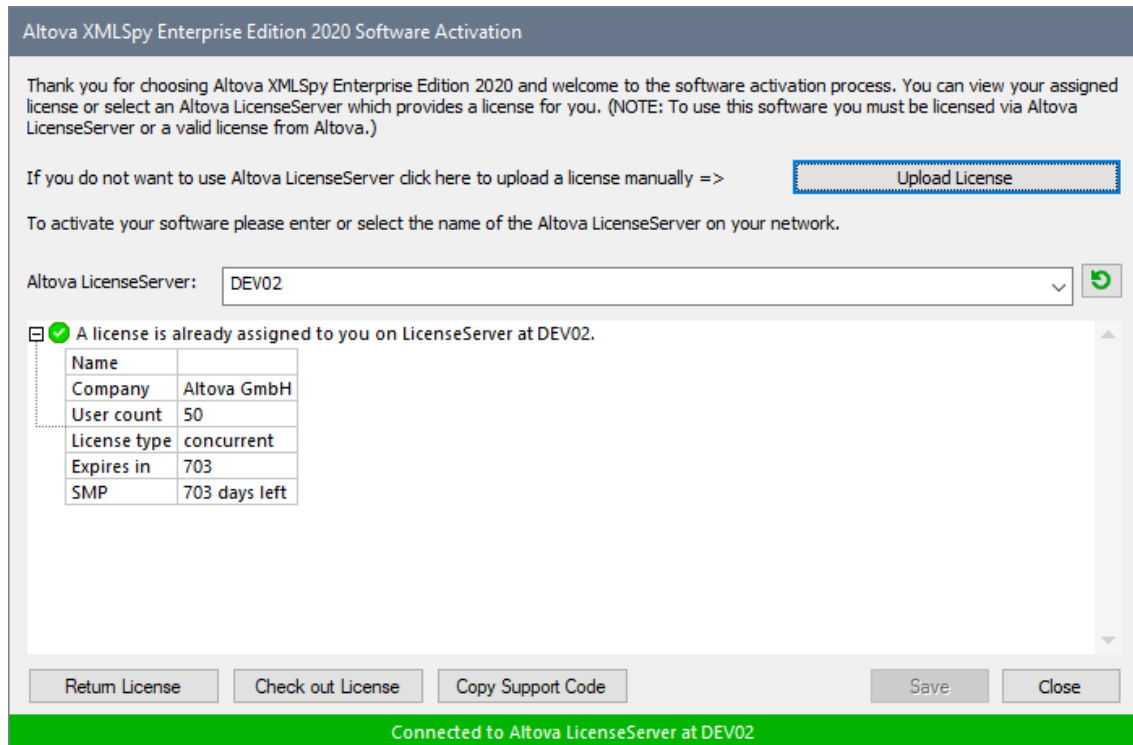
You can access the **Software Activation** dialog (*screenshot below*) at any time by clicking the **Help | Software Activation** command.

*Activate your software*

You can activate the software by registering the license in the Software Activation dialog or by licensing via [Altova LicenseServer](#) (see *details below*).

- *Registering the license in the Software Activation dialog.* In the dialog, click **Upload a New License** and browse for the license file. Click **OK** to confirm the path to the license file and to confirm any data you entered (your name in the case of multi-user licenses). Finish by clicking **Save**.
- *Licensing via Altova LicenseServer on your network:* To acquire 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 (see *screenshot below showing the dialog in Altova XMLSpy*).

Click **Save** to acquire the license.



After a machine-specific (aka installed) license has been acquired from LicenseServer, it cannot be returned to 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. After checking out a license, two things happen: (i) The **Software Activation** dialog will display the check-out information, including the time when the check-out period ends; (ii) 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 after the check-out period elapses, make sure that the check-out period you select adequately covers the period during which you will be working offline.

If the license being checked out is a Installed User license or Concurrent User license, then the license is checked out to the machine and is available to the user who checked out the license. If the license being checked out is a Named User license, then the license is checked out to the Windows account of the named user. License check-out will work for virtual machines, but not for virtual desktop (in a VDI). Note that, when a Named User license is checked out, the data to identify that license check-out is stored in the user's profile. For license check-out to work, the user's profile must be stored on the local machine that will be used for offline work. If the user's profile is stored at a non-local location (such as a file-share), then the checkout will be reported as invalid when the user tries to start the Altova application.

License check-ins must be to the same major version of the Altova product from which the license was checked out. So make sure to check in a license before you upgrade your Altova product to the next major version.

**Note:** For license check-outs to be possible, the check-out functionality must be enabled on LicenseServer. If this functionality has not been enabled, you will get an error message to this effect when you try to check out. 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

When you are ready to order a licensed version of the software product, you can use either the **Purchase a Permanent License Key** button in the **Software Activation** dialog (see *previous section*) or the **Order Form** command to proceed to the secure Altova Online Shop.

#### ☐ Registration

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

Checks with the Altova server whether a newer version than yours is currently available and displays a message accordingly.

## 13.11.4 Other Commands

### Support Center

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.

### Download Components and Free Tools

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

A link to the [Altova website](#) on the Internet. You can learn more about Authentic Desktop, related technologies and products on the [Altova website](#).

### About Authentic Desktop

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.

## 13.12 Command Line

Certain Authentic Desktop actions can be carried out from the command line. These commands are listed below:

### Open a file

```
authentic.exe file.xml
```

Opens the file, `file.xml`, in Authentic Desktop

### Open multiple files

```
authentic.exe file1.xml file2.xml
```

Opens the files, `file1.xml` and `file2.xml`, in Authentic Desktop

### Assign an SPS file to an XML file for Authentic View editing

```
authentic.exe myxml.xml /sps mysps.sps
```

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

```
authentic.exe mysps.sps
```

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.



## 14 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](#)<sup>325</sup>).

The API can be accessed from within the following environments:

- [Scripting Editor](#)<sup>283</sup>
- [IDE Plug-ins](#)<sup>310</sup>
- [External programs](#)<sup>325</sup>
- [ActiveX Integration](#)<sup>603</sup>

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](#)<sup>283</sup> 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](#)<sup>310</sup>.

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](#)<sup>310</sup>.

### 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](#)<sup>325</sup>.

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](#)<sup>327</sup>, 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](#)<sup>347</sup> and [VBScript](#)<sup>333</sup> 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#](#)<sup>336</sup> 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](#)<sup>347</sup>: 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](#)<sup>603</sup> 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](#)<sup>355</sup>. 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](#)<sup>603</sup>.

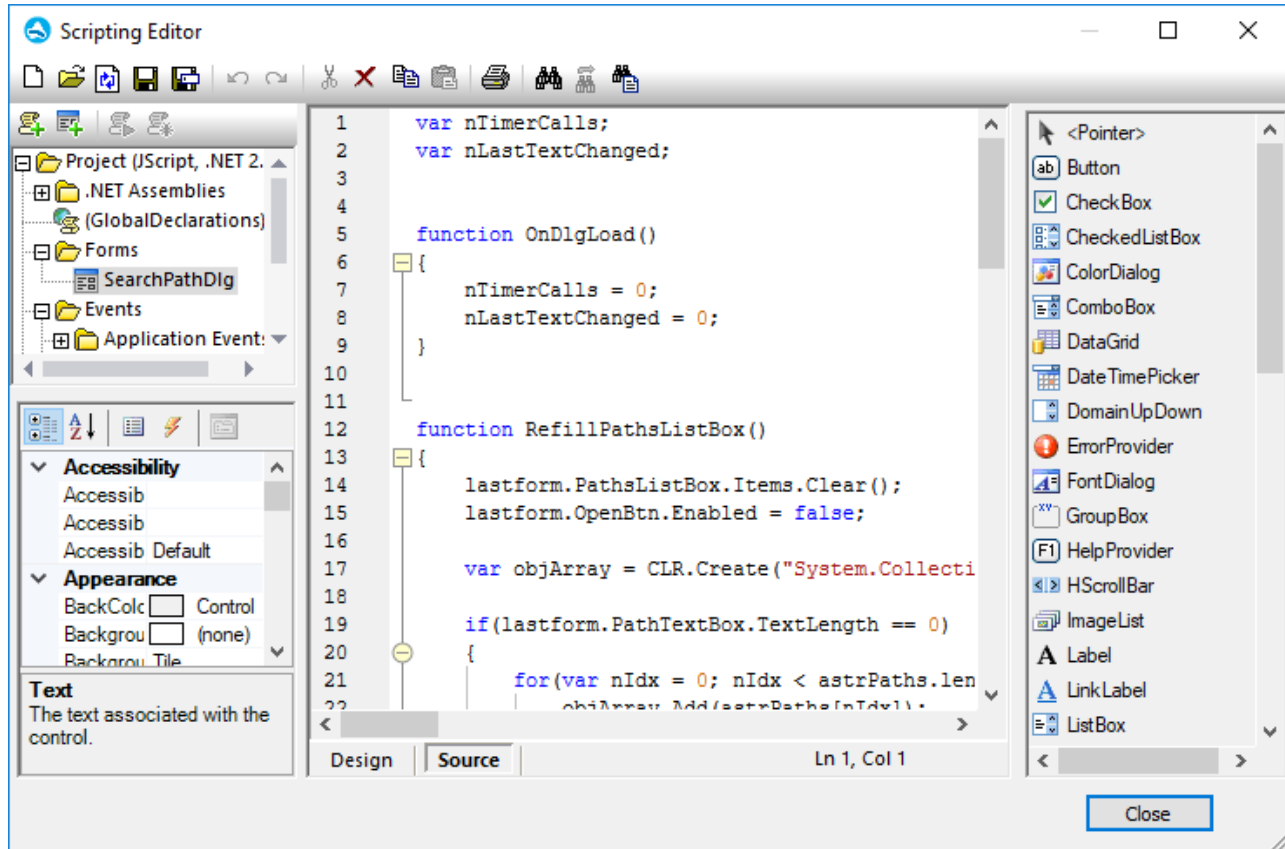
## About Programmers' Reference

The documentation contained in the Programmers' Reference for Authentic Desktop consists of the following sections:

- [Scripting Editor](#)<sup>283</sup>: a user reference for the Scripting Environment available in Authentic Desktop
- [IDE Plug-ins](#)<sup>310</sup>: a description of how to create plug-ins for Authentic Desktop
- [Application API](#)<sup>325</sup>: a reference for the Application API
- [ActiveX Integration](#)<sup>603</sup>: a guide and reference for how to integrate the Authentic Desktop GUI and Authentic Desktop functionality using an ActiveX control

## 14.1 Scripting Editor

Scripting Editor is a development environment built into Authentic Desktop from where you can customize the functionality of Authentic Desktop with the help of JScript or VBScript scripts. For example, you can add a new menu item to perform a custom project task, or you can have Authentic Desktop trigger some behavior each time when a document is opened or closed. To make this possible, you create scripting projects—files with .asprj extension (Altova Scripting Project).



Scripting Editor

Scripting projects typically include one or several macros—these are programs that perform miscellaneous custom tasks when invoked. You can run macros either explicitly from a menu item (or a toolbar button, if configured), or you can set up a macro to run automatically whenever Authentic Desktop starts. The scripting environment also integrates with the Authentic Desktop COM API. For example, your VBScript or JScript scripts can handle application or document events such as starting or shutting down Authentic Desktop, opening or closing a project, and so on. Scripting projects can include Windows Forms that you can design visually, in a way similar to Visual Studio. In addition, several built-in commands are available that help you instantiate and use .NET classes from VBScript or JScript code.

Once your scripting project is complete, you can enable it either globally in Authentic Desktop, or only for specific projects.

Scripting Editor requires .NET Framework 2.0 or later to be installed before Authentic Desktop is installed.

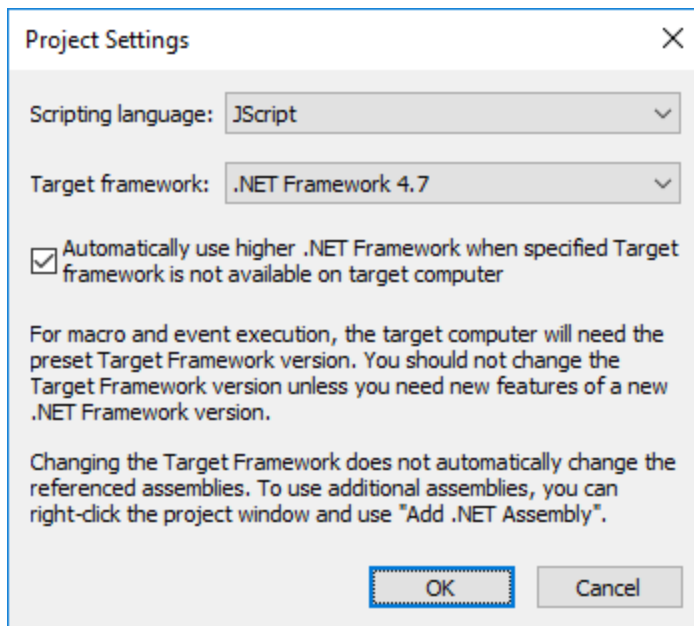
## 14.1.1 Creating a Scripting Project

All scripts and scripting information created in the Scripting Editor are stored in Altova Scripting Projects (.asprj files). A scripting project may contain macros, application event handlers, and forms (which can have their own event handlers). In addition, you can add global variables and functions to a "Global Declarations" script—this makes such variables and functions accessible across the entire project.

To start a new project, run the menu command **Tools | Scripting Editor**.

The languages supported for use in a scripting project are JScript and VBScript (not to be confused with Visual Basic, which is not supported). These scripting engines are available by default on Windows and have no special requirements to run. You can select a scripting language as follows:

1. Right-click the **Project** item in the upper-left pane, and select **Project settings** from the context menu.
2. Select a language (JScript or VBScript), and click **OK**.



From the Project settings dialog box above, you can also change the target .NET Framework version. This is typically necessary if your scripting project requires features available in a newer .NET Framework version. Note that any clients using your scripting project will need to have the same .NET Framework version installed (or a later compatible version).

By default, a scripting project references several .NET assemblies, like `System`, `System.Data`, `System.Windows.Forms`, and others. If necessary, you can import additional .NET assemblies, including assemblies from .NET Global Assembly Cache (GAC) or custom .dll files. You can import assemblies as follows:

1. Statically, by adding them manually to the project. Right-click **Project** in the top-left pane, and select **Add .NET Assembly** from the context menu.
2. Dynamically, at runtime, by calling the `CLR.LoadAssembly`<sup>300</sup> command from the code.

You can create multiple scripting projects if necessary. You can save a scripting project to the disk, and then load it back into the Scripting Editor later. To do this, use the standard Windows buttons available in the toolbar: **New**, **Open**, **Save**, **Save As**. Once the scripting project has been tested and is ready for deployment, you can load it into Authentic Desktop and run any of its macros or event handlers. For more information, see [Enabling Scripts and Macros](#) <sup>307</sup>.

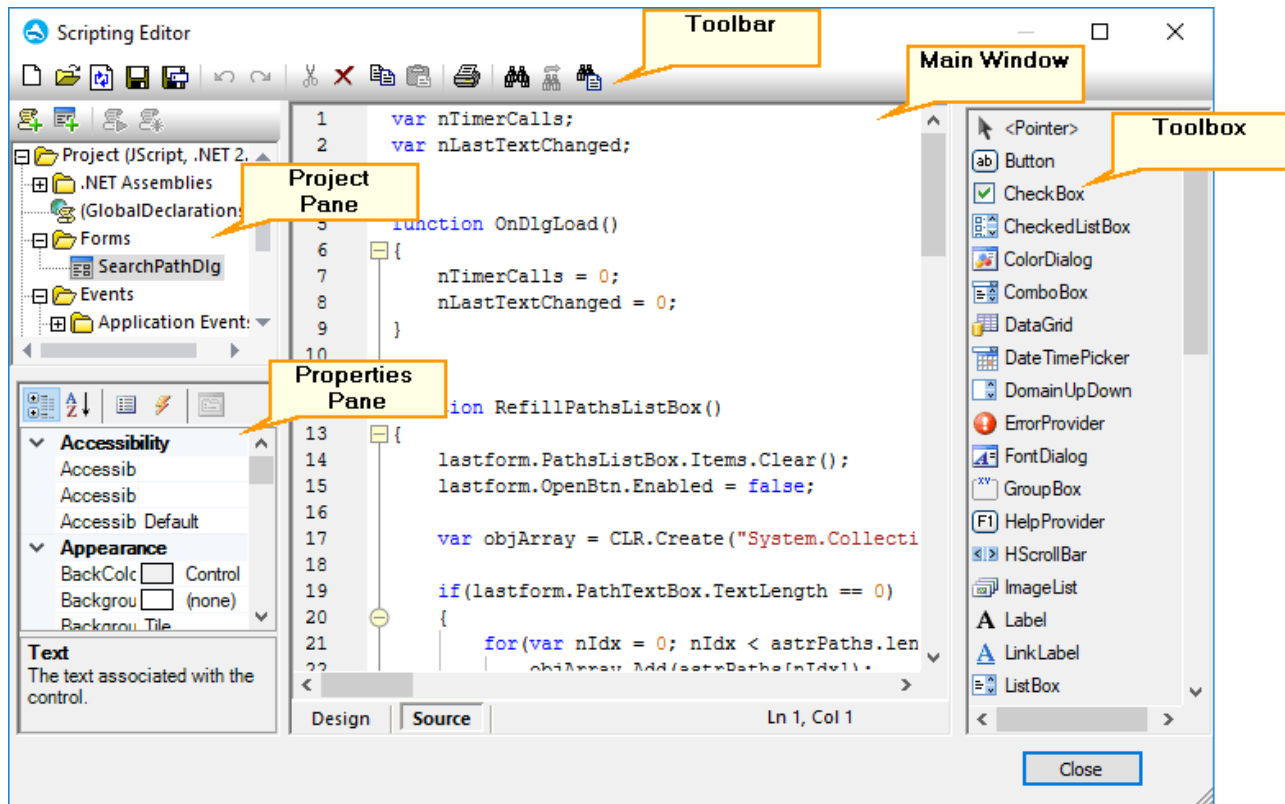
You can also find an example scripting project at the following path: **C:\Users\\Documents\Altova\Authentic2024\AuthenticExamples\SampleScripts.asprj**.

The next sections focus on the parts that your scripting project may need: global declarations, macros, forms, and events.

### 14.1.1.1 Overview of the Environment

The Scripting Editor consists of the following parts:

- Toolbar
- Project pane
- Properties pane
- Main window
- Toolbox



## Toolbar

The toolbar includes standard Windows file management commands (**New**, **Open**, **Save**, **Save As**) and editor commands (**Copy**, **Cut**, **Delete**, **Paste**). When editing source code, the **Find** and **Replace** commands are additionally available, as well as the **Print** command.





## Project pane

The project pane helps you view and manage the structure of the project. A scripting project consists of several components that can work together and may be created in any order:

- A *"Global Declarations"* script. As the name suggests, this script stores information available globally across the project. You can declare in this script any variables or functions that you need to be available in all forms, event handler scripts, and macros.
- *Forms*. Forms are typically necessary to collect user input, or provide some informative dialog boxes. A form is invoked by a call to it either within a function (in the Global Declarations script) or directly in a macro.
- *Events*. The "Events" folder displays Authentic Desktop application events provided by the COM API. To write a script that will be executed when an event occurs, double-click any event, and then type the handling code in the editor. The application events should not be confused with form events; the latter are handled at form level, as further detailed below.
- *Macros*. A macro is a script that can be invoked either on demand from a context menu or be executed automatically when Authentic Desktop starts. Macros do not have parameters or return values. A macro can access all variables and functions declared in the Global Declarations script and it can also display forms.

Right-click any of the components to see the available context menu commands and their shortcuts. Double-click any file (such as a form or a script) to open it in the main window.



The toolbar buttons provide the following quick commands:



-  **New macro** Adds a new macro to the project, in the **Macros** directory.
-  **New form** Adds a new form to the project, in the **Forms** directory.
-  **Run macro** Runs the selected macro.
-  **Debug macro** Runs the selected macro in debug mode.

## Properties pane

The Properties pane is very similar to the one in Visual Studio. It displays the following:

- Form properties, when a form is selected
- Object properties, when an object in a form is selected
- 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 the **Properties**  or **Events**  buttons, respectively.

The **Categorized**  and **Alphabetical**  icons 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 pane.

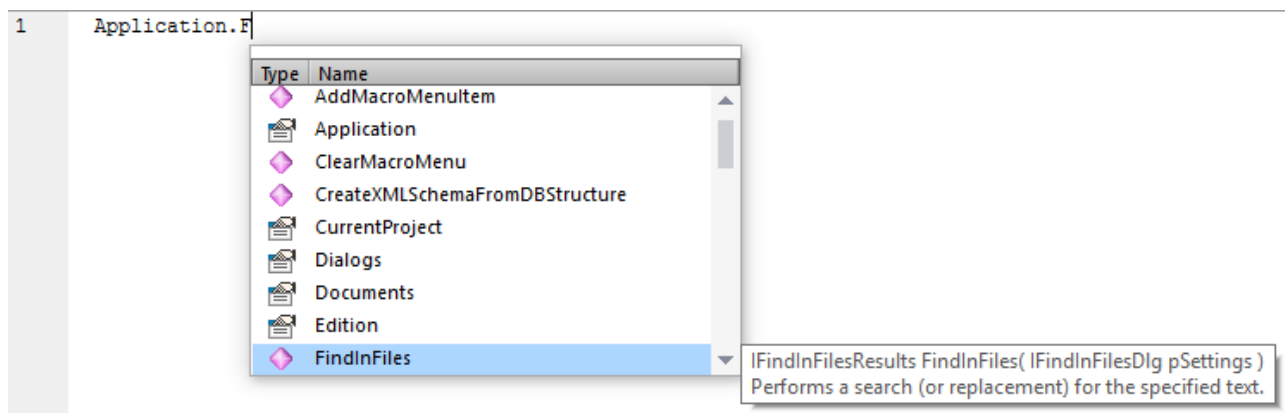
### Main window

The main window is the working area where you can enter source code or modify the design of the form. When editing forms, you can work in two tabs: the **Design** tab and the **Source** tab. The **Design** tab shows the layout of the form, while the **Source** tab contains the source code such as handler methods for the form events.

The source code editor provides code editing aids such as syntax coloring, source code folding, highlighting of starting and ending braces, zooming, autocompletion suggestions, bookmarks.

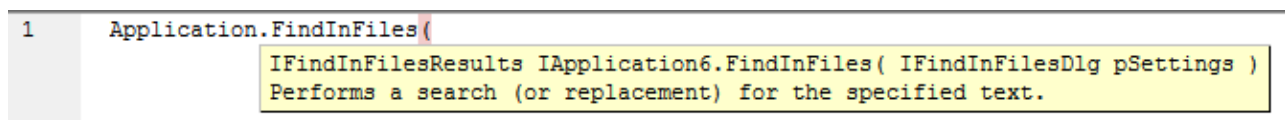
### Autocompletion suggestions

JScript and VBScript are untyped languages, so autocompletion is limited to COM API names and Authentic Desktop built-in [commands](#) <sup>297</sup>. The full method or property signature is shown next to the autocompletion entry helper.



If names start with `objDocument`, `objProject`, `objXMLData`, or `objAuthenticRange`, members of the corresponding interface will be shown.

Placing the mouse over a known method or property displays its signature (and documentation if available), for example:



The auto-completion entry helper is normally shown automatically during editing, but it can also be obtained on demand by pressing **Ctrl+Space**.

### Bookmarks

- To set or remove a bookmark, click inside a line, and then press **Ctrl+F2**
- To navigate to the next bookmark, press **F2**
- To navigate to the previous bookmark, press **Shift+F2**
- To delete all bookmarks, press **Ctrl+Shift+F2**

### Zooming in/out

- To zoom in or out, hold the **Ctrl** key pressed and then press the "+" or "-" keys or rotate the mouse wheel.

### Text view settings

To trigger text settings, right-click inside the editor, and select **Text View Settings** from the context menu.

### Font settings

To change the font, right-click inside the editor, and select **Text View Font** from the context menu.

## Toolbox

The Toolbox contains all the objects that are available for designing forms, such as buttons, text boxes, combo boxes, and so on.

### To add a Toolbox item to a form:

1. Create or open a form and make sure that the **Design** tab is selected.
2. Click the Toolbox object (for example, **Button**), and then click at the location in the form where you wish to insert it. Alternatively, drag the object directly onto the form.

Some objects such as `Timer` are not added to the Form but are created 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 from the Properties pane. For an example of handling tray components from the code, see [Handling form events](#) <sup>290</sup>.

You can also add registered ActiveX controls to the form. To do this, right-click the Toolbox area and select **Add ActiveX Control** from the context menu.

## 14.1.1.2 Global Declarations

The "Global Declarations" script is present by default in any scripting project; you do not need to create it explicitly. Any variables or functions that you add to this script are considered global across the entire project. Consequently, you can refer to such variables and functions from any of the project's macros and events. The following is an example of a global declarations script that imports the `System.Windows.Forms` namespace into the project. To achieve that, the code below invokes the `CLR.Import` command built into Scripting Editor.

```
// import System.Windows.Forms namespace for all macros, forms and events:  
CLR.Import( "System.Windows.Forms" );
```

**Note:** Every time a macro is executed or an event handler is called, the global declarations are re-initialized.

## 14.1.1.3 Macros

Macros are scripts that contain JScript (or VBScript, depending on your project's language) statements, such as variable declarations and functions.



If your projects should use macros, you can add them as follows: right-click inside the Project pane, select **Add Macro** from the context menu, and then enter the macro's code in the main form. The code of a macro could be as simple as an alert, for example:

```
alert("Hello, I'm a macro!");
```



More advanced macros can contain variables and local functions. Macros can also contain code that invokes forms from the project. The listing below illustrates an example of a macro that shows a form. It is assumed that this form has already been created in the "Forms" folder and has the name "SampleForm", see also [Forms](#) <sup>289</sup>.

```
// display a form  
ShowForm( "SampleForm" );
```

In the code listing above, `ShowForm` is a command built into Scripting Editor. For reference to other similar commands that you can use to work with forms and .NET objects, see the [Built-in Commands](#) <sup>297</sup>.

You can add multiple macros to the same project, and you can designate any macro as "auto-macro". When a macro is designated as "auto-macro", it runs automatically when Authentic Desktop starts. To designate a macro as auto-macro, right-click it, and select **Set as Auto-Macro** from the context menu.

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.

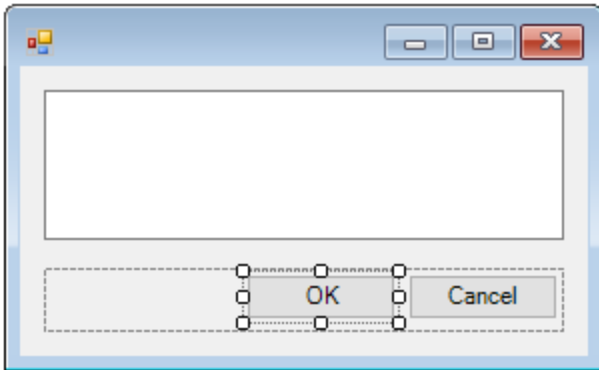
To run a macro directly in Script Editor, click **Run Macro** . To debug a macro using the Visual Studio debugger, click **Debug Macro** . For information about enabling and running macros in Authentic Desktop, see [Enabling Scripts and Macros](#) <sup>307</sup>.

#### 14.1.1.4 Forms

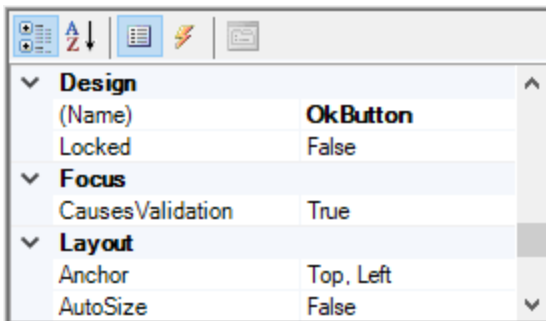
Forms are particularly useful if you need to collect input data from users or display data to users. A form can contain miscellaneous controls to facilitate this, such as buttons, check boxes, combo boxes, and so on.

To add a form, right-click inside the Project pane, and then select **Add Form** from the context menu. To add a control to a form, drag it from the Toolbox available to the right side of Scripting Editor and drop it onto the form.

You can change the position and size of the controls directly on the form, by using the handles that appear when you click any control, for example:




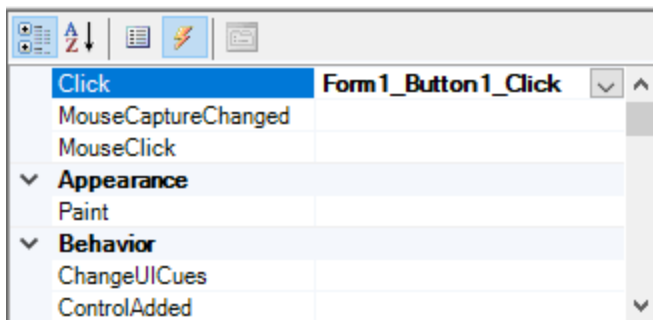
All form controls have properties that you can easily adjust in the Properties pane. To do this, first select the control on the form, and then edit the required properties in the Properties pane.



## Handling form events

Each form control also exposes various events to which your scripting project can bind. For example, you might want to invoke some Authentic Desktop COM API method whenever a button is clicked. To create a function that binds to a form event, do the following:

1. In the Properties pane, click **Events** .
2. In the **Action** column, double-click the event where you need the method (for example, in the image below, the handled event is "Click").



You can also add handler methods by double-clicking a control on the form. For example, double-clicking a button in the form design generates a handler method for the "Click" event of that button.

Once the body of the handler method is generated, you can type code that handles this event, for example:

```
//Occurs when the component is clicked.
function MyForm_ButtonClick( objSender, e_EventArgs )
{
    alert("A button was clicked");
}
```

To display a work-in-progress form detached from the Scripting Editor, right-click the form in the Project window, and select **Test Form** from the context menu. Note that the **Test Form** command just displays the form; the form's events (such as button clicks) are still disabled. To have the form react to events, call it from a macro, for example:

```
// Instantiate and display a form
ShowForm( "SampleForm" );
```

## Accessing form controls

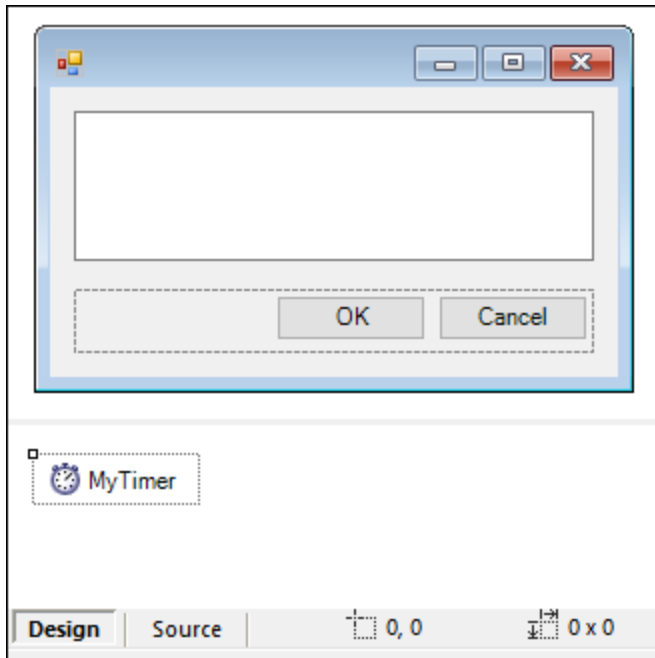
You can access any components on a form from your code by using field access syntax. For example, suppose there is a form designed as follows:

```
// MyForm
//   ButtonPanel
//     OkButton
//     CancelButton
//   TextEditor
//     AxMediaPlayer1
// TrayComponents
//   MyTimer
```

The code below shows how to instantiate the form, access some of its controls using field access syntax, and then display the form:

```
// Instantiate the form
var objForm = CreateForm("MyForm");
// Disable the OK button
objForm.ButtonPanel.OkButton.Enabled = false;
// Change the text of TextEditor
objForm.TextEditor.Text = "Hello";
// Show the form
objForm.ShowDialog();
```

When you add certain controls such as timers to the form, they are not displayed on the form; instead, they are shown as tray components at the base of the form design, for example:



To access controls from the tray, use the `GetTrayComponent` method on the form object, and supply the name of the control as argument. In this example, to get a reference to `MyTimer` and enable it, use the following code:

```
var objTimer = objForm.GetTrayComponent("MyTimer");
objTimer.Enabled = true;
```

For ActiveX Controls, you can access the underlying COM object via the `OCX` property:

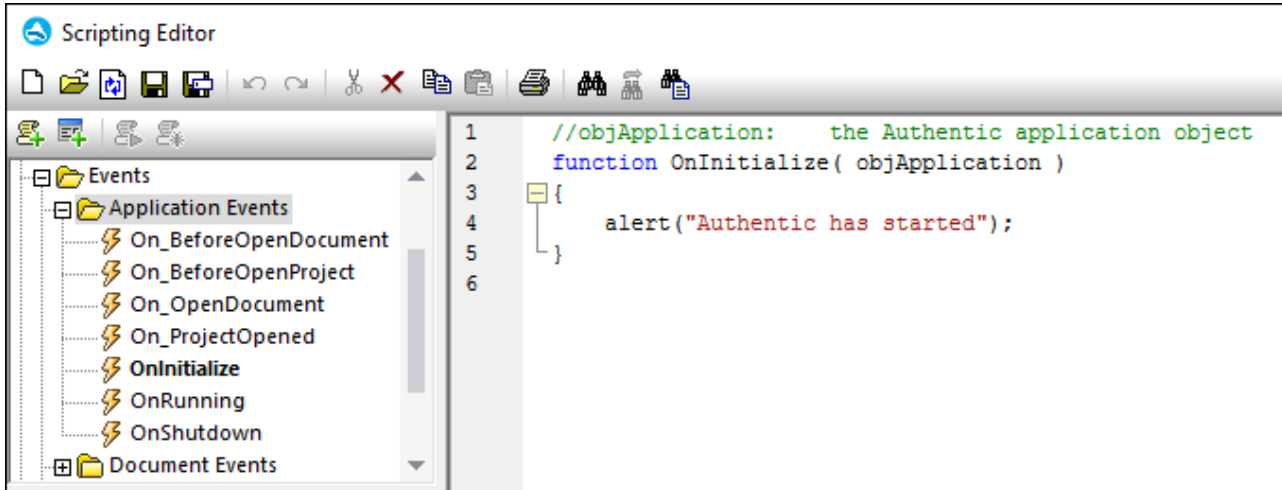
```
var ocx = lastform.AxMediaPlayer1.OCX; // get underlying COM object
ocx.enableContextMenu = true;
ocx.URL = "mms://apasf.apa.at/fm4_live_worldwide";
```

### 14.1.1.5 Events

Your scripting project may optionally include scripts that handle Authentic Desktop events such as opening, closing, or saving a document, starting or closing Authentic Desktop, adding an element to a diagram, and others. These events are provided by the Authentic Desktop COM API, and you can find them in the "Events" folder of your scripting project. Note that these events are Authentic Desktop-specific, as opposed to form events. Events are organized into folders as follows:

- Application Events
- Document Events
- AuthenticView Events
- GridView Events
- TextView Events

To create an event handler script, right-click an event, and select **Open** from the context menu (or double-click the event). The event handler script is displayed in the main window, where you can start editing it. For example, the event handler illustrated below displays an alert each time Authentic Desktop starts:



Note the following:

- The `alert` command is applicable to JScript. The VBScript equivalent is `MsgBox`. See also [alert](#)<sup>298</sup>.
- The name of the event handler function must not be changed; otherwise, the event handler script will not be called.
- In order for events to be processed, the **Process Events** check box must be selected when you enable the scripting project in Authentic Desktop. For more information, see [Enabling Scripts and Macros](#)<sup>307</sup>.

You can optionally define local variables and helper functions within event handler scripts, for example:

```

var local;

function OnInitialize( objApplication )
{
    local = "OnInitialize";
    Helper();
}

function Helper()
{
    alert("I'm a helper function for " + local);
}
    
```

### 14.1.1.6 JScript Programming Tips

Below are a few JScript programming tips that you may find useful while developing a scripting project in Authentic Desktop Scripting Editor.

## Out parameters

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
```

## Integer arguments

.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:

```
var objCustomColor =
CLR.Static("System.Drawing.Color").FromArgb(Math.floor(128),Math.floor(128),Math.floor(128));
```

## Iterating .NET collections

To iterate .NET collections, the JScript Enumerator as well as the .NET iterator technologies can be used, for example:

```
// 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

.NET templates can be instantiated as shown below:

```
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
```

or

```
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary<String,Dictionary<String,String>>" );
```

### .NET enumeration values

.NET enumeration values are accessed as shown below:

```
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch;
```

### Enumeration literals

The enumeration literals from the Authentic Desktop API can be accessed as shown below (there is no need to know their numerical value).

```
objExportXMIFileDialog.XMIType = eXMI21ForUML23;
```

## 14.1.1.7 Example Scripting Project

A demo project that illustrates scripting with Authentic Desktop is available at the following path: **C:\Users\<user>\Documents\Altova\Authentic2024\AuthenticExamples\SampleScripts.asprj**. This scripting project consists of a few macros and a Windows form.

#### To load the scripting project into Scripting Editor:

1. On the **Tools** menu, click **Scripting Editor**.
2. Click **Open** and browse for the **SampleScripts.asprj** file from the path above.

The project contains several macros in the "Macros" directory.

Macro	Description
<b>AddMacroMenu</b>	<p>This macro adds a new menu item to Authentic Desktop, by invoking the <code>Application.AddMacroMenuItem</code> method of the COM API. The first argument of the <code>AddMacroMenuItem</code> method is the name of the macro to be added (in this example, "CloseAllButActiveDoc") and the second argument is the display text for the menu item.</p> <p>Whenever this macro is run, a new menu command called "CloseAllButActiveDoc" is added under the <b>Tools</b> menu. To clear macro menu items created previously, either restart Authentic Desktop or create a macro that calls the <code>Application.ClearMacroMenu</code> API method.</p>
<b>CloseAllButActiveDocument</b>	<p>When executed, the macro iterates though the currently open documents in Authentic Desktop and closes all of them, except for the active</p>

	document.
<b>SearchPath</b>	<p>This macro displays a form that lets users perform search for files within the current project. The form is available in the "Forms" directory, where you can view its design and the associated event handlers.</p> <p>The <code>GetAllPathsFromProject()</code> method returns all the file paths that belong to the currently opened project, as an array. The definition of this method is in the <b>GlobalDeclarations</b> script of the project. The <code>InsertStringInArrayUnique</code> method ensures that only unique paths are added to the array. Next, the form is initialized with <a href="#">CreateForm</a><sup>303</sup>. Finally, the array is converted to a .NET type with the help of the <a href="#">CLR.Create</a><sup>299</sup> method and the form is populated with the resulting <code>ArrayList</code> collection.</p> <p>The <b>Open</b> button of the form has a handler that calls the <code>Application.Documents.OpenFile</code> API method to open the currently selected file.</p>

#### To enable the scripting project as global Authentic Desktop scripting project:

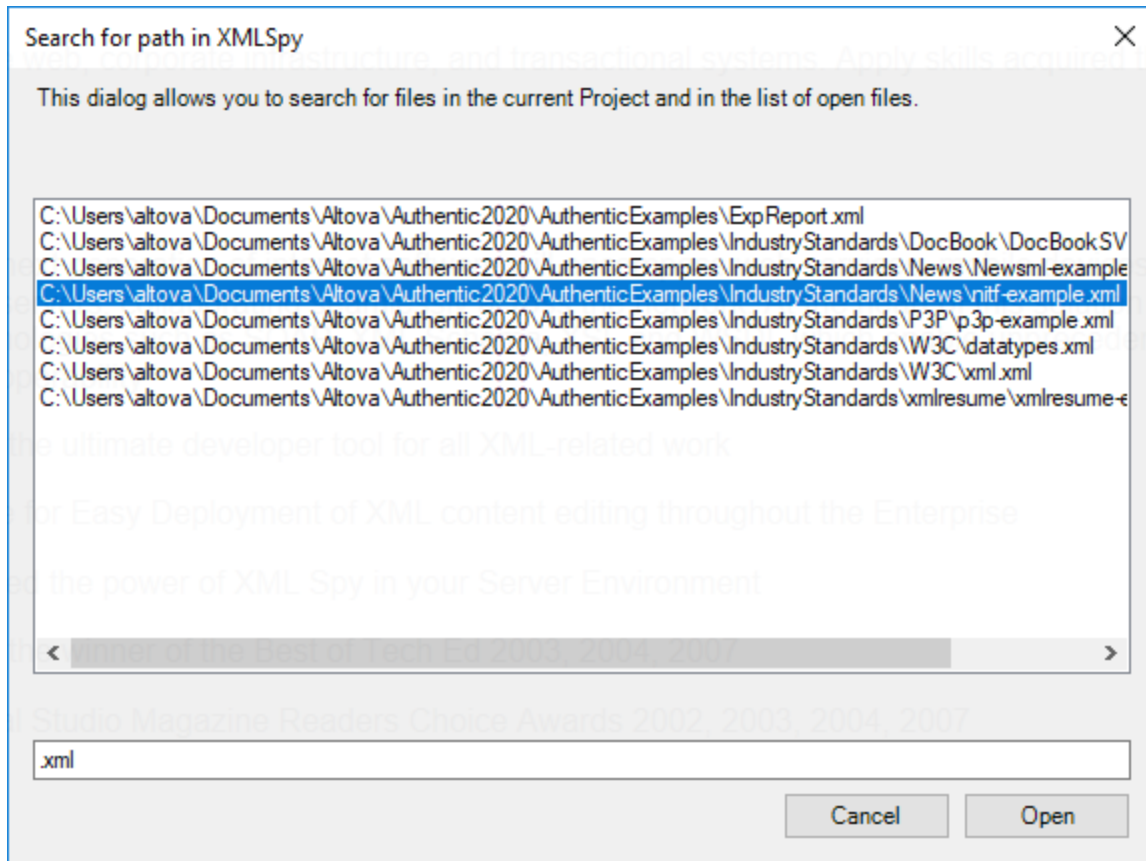
1. On the **Tools** menu, click **Options**.
2. Click the **Scripting** tab.
3. Under "Global scripting project file", click **Browse** and select the **SampleScripts.asprj** file from the path above.
4. This scripting project does not have auto-macros and application event handlers; therefore, you don't need to select either the **Run auto-macros...** or **Process events** check boxes.
5. Click **Apply**.

At this stage, several new menu items (one for each macro) become available under the **Tools | Macros** menu.

#### To run the "SearchPath" macro:

1. Open an Authentic Desktop project that contains several files (in this example, **C:\Users\<user>\Documents\Altova\Authentic2024\AuthenticExamples\Examples.spp**).
2. On the **Tools** menu, click **Macros**, and then click **Search Path**.
3. Type the search term (in this example, ".xml").





As shown above, all file names that contain the search term are now listed. You can click any element in the list, and then click **Open** to display it in the main editor.

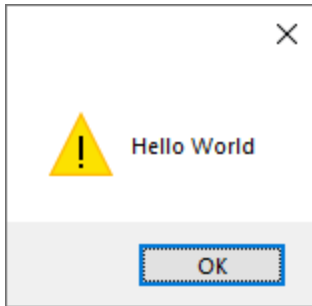
### 14.1.2 Built-in Commands

This section provides reference to all the commands you can use in the Authentic Desktop Scripting Editor.

- [alert](#) <sup>298</sup>
- [confirm](#) <sup>298</sup>
- [CLR.Create](#) <sup>299</sup>
- [CLR.Import](#) <sup>300</sup>
- [CLR.LoadAssembly](#) <sup>300</sup>
- [CLR.ShowImports](#) <sup>301</sup>
- [CLR.ShowLoadedAssemblies](#) <sup>302</sup>
- [CLR.Static](#) <sup>302</sup>
- [createForm](#) <sup>303</sup>
- [doevents](#) <sup>304</sup>
- [lastform](#) <sup>304</sup>
- [prompt](#) <sup>305</sup>
- [ShowForm](#) <sup>306</sup>
- [watchdog](#) <sup>306</sup>

### 14.1.2.1 alert

Displays a message box that shows a given message and the "OK" button. To proceed, the user will have to click "OK".



#### Signature

For JScript, the signature is:

```
alert(strMessage : String) -> void
```

For VBScript, the signature is:

```
MsgBox(strMessage : String) -> void
```

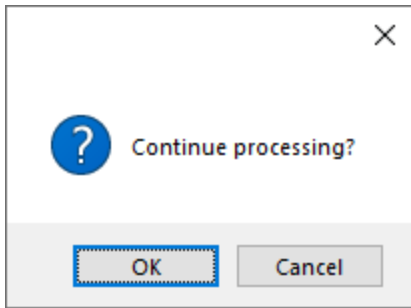
#### Example

The following JScript code displays a message box with the text "Hello World".

```
alert("Hello World");
```

### 14.1.2.2 confirm

Opens a dialog box that shows a given message, a confirmation button, and a cancel button. The user will have to click either "OK" or "Cancel" to proceed. Returns a Boolean that represents the user's answer. If the user clicked "OK", the function returns **true**; if the user clicked "Cancel", the function returns **false**.



### Signature

```
confirm(strMessage : String) -> result : Boolean
```

### Example (JScript)

```
if ( confirm( "Continue processing?" ) == false )
    alert("You have cancelled this action");
```

### Example (VBScript)

```
If ( confirm( "Continue processing?" ) = false ) Then
    MsgBox ("You have cancelled this action")
End If
```

## 14.1.2.3 CLR.Create

Creates a new .NET object instance of the type name supplied as argument. If more than one argument is passed, the successive arguments are interpreted as the arguments for the constructor of the .NET object. The return value is a reference to the created .NET object

### Signature

```
CLR.Create(strTypeNameCLR : String, constructor arguments ... ) -> object
```

### Example

The following JScript code illustrates how to create instances of various .NET classes.

```
// Create an ArrayList
var objArray = CLR.Create("System.Collections.ArrayList");
// Create a ListViewItem
var newItem = CLR.Create( "System.Windows.Forms.ListViewItem", "NewItemText" );
// Create a List<string>
```

```
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
// Import required namespaces and create a Dictionary object
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary<String, Dictionary<String, String >>" );
```

### 14.1.2.4 CLR.Import

Imports a namespace. This is the scripting equivalent of C# `using` and VB.Net `imports` keyword. Calling `CLR.Import` makes it possible to leave out the namespace part in subsequent calls like `CLR.Create()` and `CLR.Static()`.

**Note:** Importing a namespace does not add or load the corresponding assembly to the scripting project. You can add assemblies to the scripting project dynamically (at runtime) in the source code by calling [CLR.LoadAssembly](#)<sup>300</sup>.

#### Signature

```
CLR.Import(strNamespaceCLR : String) -> void
```

#### Example

Instead of having to use fully qualified namespaces like:

```
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
    CLR.Static( "System.Windows.Forms.MessageBox" ).Show( "Hello " + sName );
}
```

One can import namespaces first and subsequently use the short form:

```
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 );
}
```

### 14.1.2.5 CLR.LoadAssembly

Loads the .NET assembly with the given long assembly name or file path. Returns Boolean **true** if the assembly could be loaded; **false** otherwise.

## Signature

```
CLR.LoadAssembly(strAssemblyNameCLR : String, showLoadErrors : Boolean) -> result : Boolean
```

## Example

The following JScript code attempts to set the clipboard text by loading the required assembly dynamically.

```
// 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" );
}
```

### 14.1.2.6 CLR.ShowImports

Opens a message box that shows the currently imported namespaces. The user will have to click "OK" to proceed.

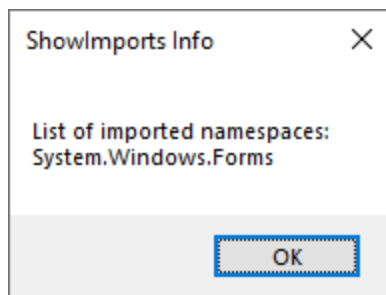
## Signature

```
CLR.ShowImports() -> void
```

## Example

The following JScript code first imports a namespace, and then displays the list of imported namespaces:

```
CLR.Import( "System.Windows.Forms" );
CLR.ShowImports();
```



### 14.1.2.7 CLR.ShowLoadedAssemblies

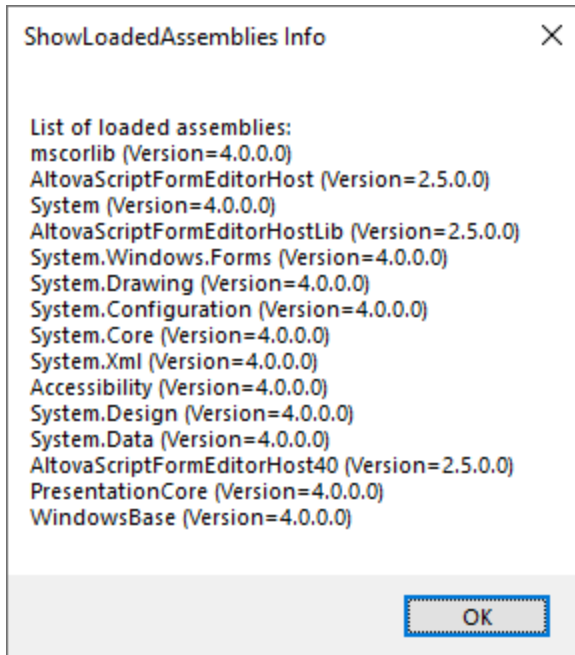
Opens a message box that shows the currently loaded assemblies. The user will have to click "OK" to proceed.

#### Signature

```
CLR.ShowLoadedAssemblies() -> void
```

#### Example

```
CLR.ShowLoadedAssemblies();
```



### 14.1.2.8 CLR.Static

Returns a reference to a static .NET object. You can use this function to get access to .NET types that have no instances and contain only static members.

#### Signature

```
CLR.Static(strTypeNameCLR : String) -> object
```

## Example (JScript)

```
// Get the value of a .NET Enum into a variable
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch

// Set the value of the Windows clipboard
var clipboard = CLR.Static( "System.Windows.Clipboard" );
clipboard.SetText( "HelloClipboard" );

// Check the buttons pressed by the user on a dialog box
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

### 14.1.2.9 CreateForm

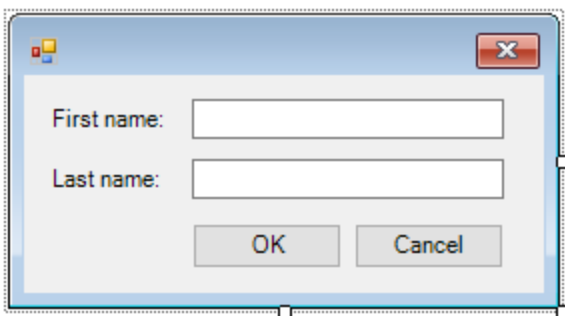
Instantiates the `Form` object identified by the name supplied as argument. The form must exist in the "Forms" folder of the scripting project. Returns the form object (`System.Windows.Forms.Form`) corresponding to the given name, or `null` if no form with such name exists.

#### Signature

```
CreateForm (strFormName : String) -> System.Windows.Forms.Form | null
```

#### Example

Let's assume that a form called "FormName" exists in the scripting project.



The following JScript code instantiates the form with some default values and displays it to the user.

```
var myForm = CreateForm( "FormName" );
if ( myForm != null )
{
    myForm.textboxFirstName.Text = "Daniela";
    myForm.textboxLastName.Text = "Heidegger";
}
```

```
var dialogResult = myForm.ShowDialog();  
}
```

The `dialogResult` can subsequently be evaluated as follows:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )  
    alert( "ok" );  
else  
    alert( "cancel" );
```

**Note:** The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

### 14.1.2.10 `doevents`

Processes all Windows messages currently in the message queue.

#### Signature

```
doevents() -> void
```

#### Example (JScript)

```
for ( i=0; i < nLongLastingProcess; ++i )  
{  
    // do long lasting process  
  
    doevents(); // process Windows messages; give UI a chance to update  
}
```

### 14.1.2.11 `lastform`

This is a global field that returns a reference to the last form object that was created via `CreateForm()` or `ShowForm()`.

#### Signature

```
lastform -> formObj : System.Windows.Forms.Form
```

#### Example

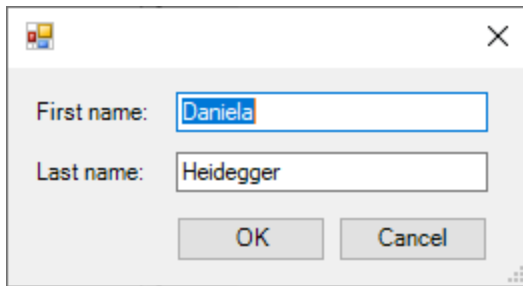
The following JScript code shows the form "FormName" as a dialog box.

```
CreateForm( "FormName" );  
if ( lastform != null )
```



```
{
  lastform.textboxFirstName.Text = "Daniela";
  lastform.textboxLastName.Text = "Heidegger";
  var dialogResult = lastform.ShowDialog();
}
```

The values of both textbox controls are initialized with the help of `lastform`.



### 14.1.2.12 prompt

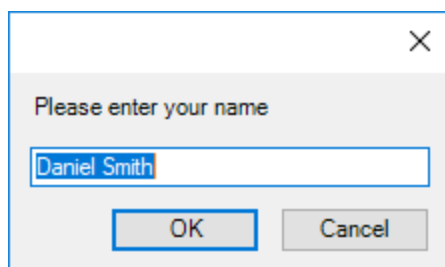
Opens a dialog box that shows a message and a textbox control with a default answer. This can be used to let the user input a simple string value. The return value is a string that contains the textbox value or null if the user selected "Cancel".

#### Signature

```
prompt(strMessage : String, strDefault : String) -> val : String
```

#### Example

```
var name = prompt( "Please enter your name", "Daniel Smith" );
if ( name != null )
  alert( "Hello " + name + "!" );
```



### 14.1.2.13 ShowForm

Instantiates a new form object from the given form name and immediately shows it as dialog box. The return value is an integer that represents the generated `DialogResult` (`System.Windows.Forms.DialogResult`). For the list of possible values, refer to the documentation of the `DialogResult` Enum (<https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.dialogresult?view=netframework-4.8>).

#### Signature

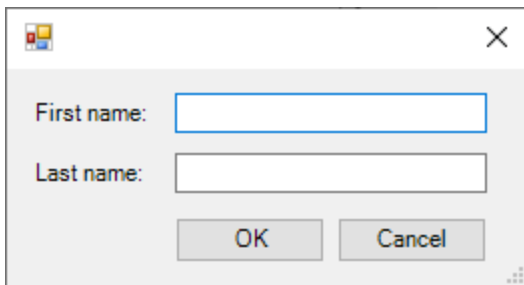
```
ShowForm(strFormName : String) -> result : Integer
```

#### Example

The following JScript code

```
var dialogResult = ShowForm( "FormName" );
```

Shows the form "FormName" as a dialog box:



The `DialogResult` can subsequently be evaluated, for example:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )  
    alert( "ok" );  
else  
    alert( "cancel" );
```

**Note:** The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

### 14.1.2.14 watchdog

Long running CPU-intensive scripts may ask the user if the script should be terminated. The `watchdog()` method is used to disable or enable this behavior. By default, the watchdog is enabled.

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.

## Signature

```
watchdog(bEnable : boolean) -> void
```

## 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
```

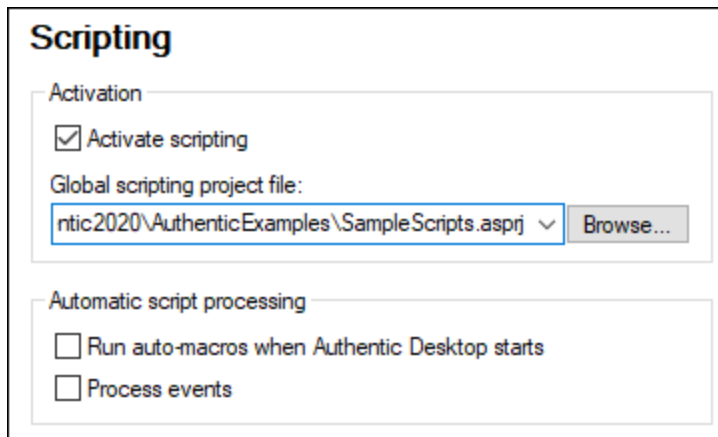
### 14.1.3 Enabling Scripts and Macros

Once a scripting project is complete and tested, you can use it in the following ways:

1. As the global scripting project for Authentic Desktop. This means that all the scripts and macros from the scripting project are available to Authentic Desktop.
2. At Authentic Desktop project level. This means that a reference to the .asprj file is saved together with the Authentic Desktop project. When the Authentic Desktop project is opened, its associated scripts and macros can be called.

**To set a scripting project as global:**

1. On the **Tools** menu, click **Options**.
2. Click the **Scripting** tab.
3. Select the **Activate scripting** check box and browse for the .asprj file to be used as global scripting project.



You can optionally enable the following additional script processing options:

<p><b>Run auto-macros when Authentic Desktop starts</b></p>	<p>If you select this check box, any macros that were set as "Auto-macro" in the project will be triggered automatically when Authentic Desktop starts.</p>
---	---

<b>Process events</b>	Select this check box if your scripts bind to any application events. Clear the check box to prevent the scripts from reacting to events.
-----------------------	---

#### To enable a scripting project at project level:

1. Open the project.
2. On the **Project** menu, click **Script Settings**.
3. Select the **Activate project scripts** check box and browse for the .asprj file.

The **Run-auto macros...** check box has the same meaning as already described above.

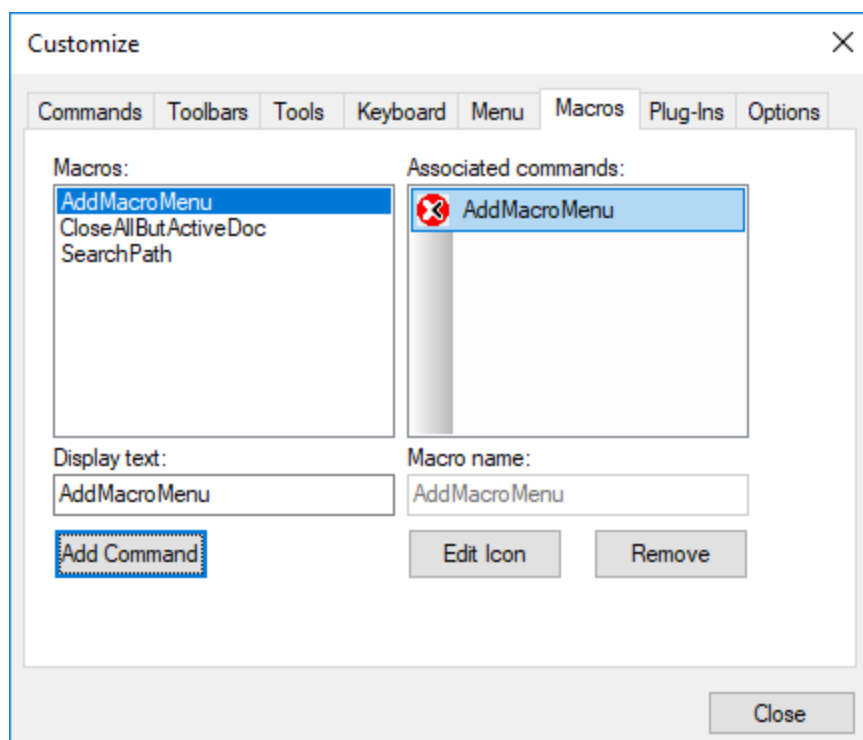
### 14.1.3.1 Running Macros

When a scripting project is active in Authentic Desktop, any macros available in that project are displayed in the **Tools | Macros** menu. Therefore, you can run a macro at any time, by triggering the respective menu command, for example **Tools | Macros | <SomeMacro>**.

Macros that were configured as auto-macros will run automatically whenever Authentic Desktop starts, provided that this behavior is enabled from options, as described in [Enabling Scripts and Macros](#)<sup>307</sup>.

For convenience, you can create toolbar buttons for macros, as follows:

1. On the **Tools** menu, click **Customize**.
2. Click the **Macros** tab. Any macros that are available at application level (in the *global* scripting project) are listed.
3. Click **Add Command**.



4. Optionally, click **Edit icon** and draw a new icon for the new macro. You can also assign a shortcut to the macro, from the **Keyboard** tab.
5. Drag the macro from the **Associated commands** pane onto the toolbar where you would like it to appear.

#### To remove a macro from a toolbar:

1. On the **Tools** menu, click **Customize**.
2. Click the **Macros** tab.
3. Drag the macro from the toolbar where it appears back into the **Associated commands** pane.

## 14.2 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](#)<sup>320</sup> interface. VB.NET, C# and C++ are the currently supported languages, and examples using these languages are included with your installation package and are located in the `Authentic2024\AuthenticExamples\IDEPlugin` folder of your Authentic Desktop installation.

Windows 7, 8, 10, 11	C:/Users/<username>/Documents
----------------------	-------------------------------

See [ATL sample files](#)<sup>314</sup> for an example using C++.

### 14.2.1 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\PluginIns
```

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 `PluginIns` 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 Authentic Desktop's **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 `PluginIns` key.

If you experience problems with manual registration, check whether the CLSID of your plug-in is correctly registered in the `PluginIns` key. If the registration is incorrect, then 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. (This can be done, for example, during plug-in setup.) 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](#)<sup>11</sup>: `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 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).

**Note:** VB.NET, C# and C++ are the currently supported languages.

## 14.2.2 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 `IDEPlugin` folder in the `Examples` folder of your application folder.

## 14.2.3 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](#)<sup>323</sup> method of the `IXMLSpyPlugIn` interface.

The XML file containing the UI modifications for the IDE PlugIn, must have the following structure:

```
<ConfigurationData>
  <ImageFile>Path to image file</ImageFile>
  <Modifications>
    <Modification>
      ...
    </Modification>
    ...
  </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for new menu items that 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 16x16 pixels using maximum 256 colors. The image references must be arranged from left to right in a single `<ImageFile>` element. The rightmost image index value is zero.

The **Modification** element can have any number of **Modification** child elements. Each **Modification** element defines a specific change to the standard UI of Authentic Desktop. The modifications you can carry out are described in the next section below.

## Structure of Modification elements

A **Modification** element has two child elements:

```
<Modification>
  <Action>Type of action</Action>
  <UIElement Type="Type of UI element" />
</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

Multiple modifications can be combined in an **Action** element, like this: "Add Hide"

The **UIElement** element defines any new or existing UI element and may be one of the the following types: toolbars, buttons, menus, or menu items. The **type** attribute specifies which of these types the UI element belongs to. The structure of **UIElement** is described in the sections below.

## Common UIElement children

The **ID** and **Name** elements are defined for all types of UI element. In the case of some types, however, one of these elements is ignored. For example, **Name** is ignored for a separator.

```
<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 example, menus and menu items that have sub menus. For new UI elements, the **Name** element sets the caption (for example, the title of a toolbar) or the text of a menu item.

## Toolbars and Menus

To define a toolbar it is necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name or ID (if the latter is known). To create a **new** toolbar, both values must be set. The **type** attribute must have a value of **ToolBar**.

```
<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:

```
<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:

```
<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:

```
<MacroName></MacroName>
<Info></Info>
<ImageID></ImageID>
```

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 description string that 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 in the image file. Note that all icons are stored in one image file.

To define a toolbar button, create an `UIElement` with this structure:

```
<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`. The `ToolBarID` and `ToolBarName` elements 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 be `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. The content of the `MenuID` element can be either 128 or 129. See the section "Toolbars and Menus" above for more information.

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. We do this by setting the content of the `Parent` element to be the path to the menu item. The steps in the path are indicated by a colon. The pattern would be `ParentMenu:SubMenu`. If the menu has no parent (because it is not a submenu), add a colon to the beginning of the name (see example below). The `type` attribute must be set to `MenuItem`.

The example below defines a menu item, where the containing menu is not a sub menu:

```
<UIElement Type="MenuItem">
  <!--the following element is a Local command ID-->
  <ID>3</ID>
  <Name>Open file from repository...</Name>
  <Place>-1</Place>
  <MenuID>129</MenuID>
  <Parent>:PlugIn Menu</Parent>
  <ImageID>0</ImageID>
</UIElement>
```

You can add toolbar separators and menus if the value of the `ID` element is set to 0.

## 14.2.4 ATL Sample Files

This section shows how to create a simple Authentic Desktop IDE plug-in DLL using ATL. You must know how to work with MS VisualStudio, ATL, and the wizards that generate new ATL objects. 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 of the code. In addition, the sample code uses the MFC class `CString` and ATL conversion macros such as `W2T`.

The broad steps to create an ATL DLL are as follows:

1. Open VisualStudio and select **File | New**.
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 **Insert | New ATL Object**.
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. See the following pages for information about how to modify the code and specify some basic functionality.

### 14.2.4.1 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. After creating the ATL object, you need to implement the IDE plug-in interface of Authentic Desktop.

```
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 OnEvent([in] long nEventID, [in] SAFEARRAY(VARIANT)*
arrayParameters, [in] IDispatch* pXMLSpy, [out, retval] VARIANT* pReturnValue);
        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 -----
    };
};

```

### 14.2.4.2 Class definition

In the class definition of the ATL object, the following changes must be made:

- The class has to derive from `IXMLSpyPlugIn`
- The "Interface Map" needs an entry for `IXMLSpyPlugIn`
- The methods of the IDE plug-in interface must be declared

These changes can be made as shown below:

```

#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_

```

### 14.2.4.3 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.

```

////////////////////////////////////
// CXMLSpyIDEPlugIn

#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(" \
        </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);
}

```

## 14.2.5 IXMLSpyPlugIn

### Methods

[OnCommand](#) <sup>320</sup>

[OnUpdateCommand](#) <sup>321</sup>

[OnEvent](#) <sup>322</sup>

[GetUIModifications](#) <sup>323</sup>

[GetDescription](#) <sup>324</sup>

### 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)`. This is required for it to be recognized as a plug-in.

### 14.2.5.1 OnCommand

#### 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
        End If
    End If
End Sub
```



```

    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

```

## 14.2.5.2 OnUpdateCommand

### 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

```

### 14.2.5.3 OnEvent

#### 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
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
On_DocEditDragOver	= 10
On_BeforeOpenProject	= 25
On_BeforeOpenDocument	= 26
On_BeforeSaveDocument	= 27
On_BeforeCloseDocument	= 28

On_ViewActivation	= 29
On_DocEditKeyboardEvent	= 30
On_DocEditMouseEvent	= 31
On_BeforeValidate	= 32
On_BeforeShowSuggestions	= 33
On_ProjectOpened	= 34
On_Char	= 35
On_Initialize	= 36
On_Running	= 37
On_Shutdown	= 38
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.

### 14.2.5.4 GetUIModifications

**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](#)<sup>311</sup> for a detailed description how to change the UI.

### Example

```
Public Function IXMLSpyPlugIn_GetUI Modifications() As String
    ' GetUI Modifications() 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_GetUI Modifications = strMods
        Else
            IXMLSpyPlugIn_GetUI Modifications = ""
        End If
    End If
End Function
```

## 14.2.5.5 GetDescription

### 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
```

## 14.3 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](#)<sup>576</sup> 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](#)<sup>325</sup> and in the [Overview](#)<sup>326</sup> 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](#)<sup>283</sup>.
- 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](#)<sup>310</sup>.
- Via an ActiveX Control, which is available if the [integration package](#)<sup>603</sup> is installed. For more information, see the section [ActiveX Integration](#)<sup>603</sup>.

### External programs

In the [Overview](#)<sup>326</sup> 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](#)<sup>327</sup>, 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](#)<sup>333</sup> 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#](#)<sup>336</sup> 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](#)<sup>347</sup>: 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](#)<sup>332</sup> (and in [C#](#)<sup>345</sup> and [Java](#)<sup>354</sup>) 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](#)<sup>326</sup>, which describes the object model for the API and explains how the API is accessed via various [programming languages](#)<sup>327</sup>.
- The second part is a reference section ([Interfaces](#)<sup>355</sup> and [Enumerations](#)<sup>588</sup>) that contains descriptions of the interface objects of the Application API.

## 14.3.1 Overview

This overview of the Application API is organized as follows:

- [The Object Model](#)<sup>326</sup> describes the relationships between the objects of the Application API.
- [Programming Languages](#)<sup>327</sup> 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.

### 14.3.1.1 Object Model

The starting point for every application which uses the Application API is the [Application](#)<sup>356</sup> 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](#)<sup>327</sup> 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:

- Document collection and reference to the active document.
- Reference to current project and methods for creating and opening projects.
- Methods to support the export to and import from databases, text files, and Word documents.
- URL management.
- 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](#)<sup>477</sup> interface.

### 14.3.1.2 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. The files in the API subfolder of the Examples folder can be used to test this functionality:

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\Examples
---	---

### JScript

The JScript listings demonstrate the following basic functionality:

- [Start application or attach to a running instance](#)<sup>329</sup>
- [Simple document access](#)<sup>330</sup>
- [Iteration](#)<sup>331</sup>
- [Error handling](#)<sup>332</sup>
- [Events](#)<sup>332</sup>

## 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](#)<sup>328</sup>.

- [Events](#)<sup>333</sup>: 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](#)<sup>342</sup>: Starts Authentic Desktop, which is registered as an automation server, or activates the application if it is already running.
- [Open OrgChart.pxf](#)<sup>343</sup>: 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](#)<sup>346</sup>: 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](#)<sup>343</sup>: Opens another example document.
- [Toggle View Mode](#)<sup>344</sup>: Changes the view of all open documents between Browser View and Authentic View. The code shows how to iterate through open documents.
- [Validate](#)<sup>345</sup>: 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](#)<sup>342</sup>: Stops Authentic Desktop.

## Java

The Authentic Desktop API can be accessed from Java code. [The Java sub-section of this section](#)<sup>347</sup> 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](#)<sup>347</sup>
- [Example Java Project](#)<sup>349</sup>
- [Application Startup and Shutdown](#)<sup>352</sup>
- [Simple Document Access](#)<sup>353</sup>
- [Iterations](#)<sup>353</sup>
- [Use of Out-Parameters](#)<sup>354</sup>
- [Event Handlers](#)<sup>354</sup>

### 14.3.1.2.1 JScript

This section contains listings of JScript code that demonstrate the following basic functionality:

- [Start application or attach to a running instance](#)<sup>329</sup>
- [Simple document access](#)<sup>330</sup>
- [Iteration](#)<sup>331</sup>
- [Error handling](#)<sup>332</sup>
- [Events](#)<sup>332</sup>



## 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` subfolder of the API Examples folder:

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\Examples
---	---

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>).

### 14.3.1.2.1 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`. Be aware, though, that the calling program will access the CLASSES registry entries in its own registry hive, or group (32-bit or 64-bit). Therefore, if you run scripts using the standard command prompt and Windows Explorer on 64-bit Windows, the 64-bit registry entries will be accessed, which point to the 64-bit Authentic Desktop. For this reason, if both Authentic Desktop 32-bit and 64-bit are installed, special handling is required in order to call the 32-bit Authentic Desktop. For example, assuming that Windows Scripting Host is the calling program, do the following:

1. Change the current directory to **C:\Windows\SysWOW64**.
2. At the command line, type **wscript.exe** followed by the path to the script that you would like to run, for example:

```
wscript.exe "C:\Users\...
\Documents\Altova\Authentic2024\AuthenticExamples\API\JScript\start.js"
```

```
// 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 {   objAuthentic = WScript.GetObject("", "AuthenticDesktop.Application");   }
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](#) <sup>329</sup>).

#### 14.3.1.2.1.2 Simple Document Access

After you have started the application as shown in [Start Application](#) <sup>329</sup>, 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;

// 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](#)<sup>329</sup>).

### 14.3.1.2.1.3 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](#)<sup>329</sup>).

#### 14.3.1.2.1.4 Error Handling

The Application API returns errors in two different ways:

- 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`.

```
try {
    var arr = new Array(-1);
}
catch (err) {
    WScript.Echo("Error : (" + (err.number & 0xffff) + ") " + err.description);
}
```

#### 14.3.1.2.1.5 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.

```
// 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();
```

### 14.3.1.2.2 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](#)<sup>333</sup> and an [example](#)<sup>335</sup>.

For information about other functionality, refer to the JScript examples listed below:

- [Start application or attach to a running instance](#)<sup>329</sup>
- [Simple document access](#)<sup>330</sup>
- [Iteration](#)<sup>331</sup>
- [Error handling](#)<sup>332</sup>

#### 14.3.1.2.2.1 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
Function DocEvent_OnBeforeCloseDocument(objDocument)
    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`. Be aware, though, that the calling program will access the CLASSES registry entries in its own registry hive, or group (32-bit or 64-bit). Therefore, if you run scripts using the standard command prompt and Windows Explorer on 64-bit Windows, the 64-bit registry entries will be accessed, which point to the 64-bit Authentic Desktop. For this reason, if both Authentic Desktop 32-bit and 64-bit are installed, special handling is required in order to call the 32-bit Authentic Desktop. For example, assuming that Windows Scripting Host is the calling program, do the following:

1. Change the current directory to **C:\Windows\SysWOW64**.

2. At the command line, type **wscript.exe** followed by the path to the script that you would like to run, for example:

```
wscript.exe "C:\Users\...\
\Documents\Altova\Authentic2024\AuthenticExamples\API\JScript\start.js"
```

#### 14.3.1.2.2.2 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." & vbNewLine & vbNewLine & "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

```

### 14.3.1.2.3 C#

The C# programming language can be used to access the Application API functionality. You could use Visual Studio 2012/2013/2015/2017/2019/2022 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 `API\C#` subfolder of the `Examples` folder :

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\Examples
---	---

You can compile and run the project from within Visual Studio 2012/2013/2015/2017/2019/2022. 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

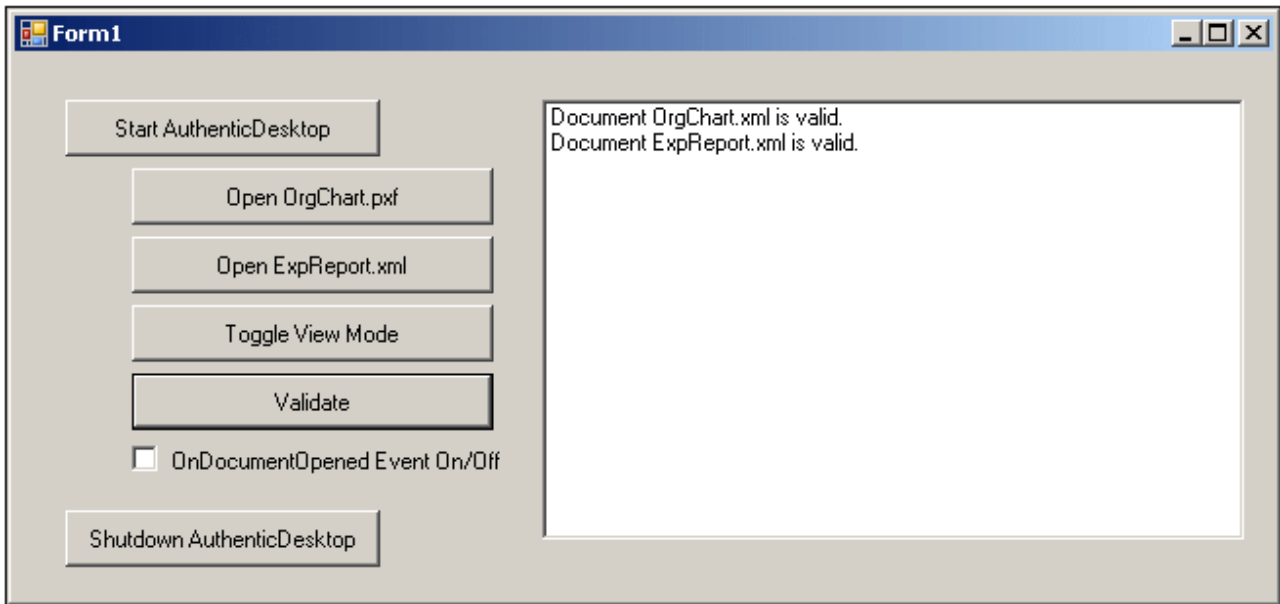
If you have a 64-bit operating system and are using a 32-bit installation of Authentic Desktop, you must add the x86 platform in the solution's Configuration Manager and build the sample using this configuration. A new



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](#)<sup>342</sup>: Starts Authentic Desktop, which is registered as an automation server, or activates the application if it is already running.
- [Open OrgChart.pxf](#)<sup>343</sup>: 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](#)<sup>343</sup>: Opens another example document.
- [Toggle View Mode](#)<sup>344</sup>: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- [Validate](#)<sup>345</sup>: 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](#)<sup>342</sup>: 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 2012/2013/2015/2017/2019/2022). 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.

```
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)
{
    if (AuthenticDesktop != null)
    {
        if (checkBoxEventOnOff.Checked)
            AuthenticDesktop.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
        else
            AuthenticDesktop.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
    }
}
}
```

### 14.3.1.2.3.1 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.tlb`, and is located in the Authentic Desktop application folder.

Then declare a variable to access the Authentic Desktop API:

```
// An instance of Authentic Desktop is accessed via its automation interface.
XMLSpyLib.Application Authentic Desktop;
```

### 14.3.1.2.3.2 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](#)<sup>336</sup> that, respectively, start up and shut down the application. This example is located in the C# subfolder of the API Examples folder (see *the file Form1.cs*):

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%>Examples
---	---

You can compile and run the project from within Visual Studio 2012/2013/2015/2017/2019/2022.

## Starting Authentic Desktop

The following code snippet from the [AutomateAuthenticDesktop example](#)<sup>336</sup> shows how to start up the application.

```
// 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 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](#)<sup>336</sup> 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;
        }
    }
}
```

#### 14.3.1.2.3.3 Opening Documents

The code snippets below (from the [AutomateAuthenticDesktop example](#)<sup>336</sup>) 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\(string, boolean\)](#)<sup>480</sup>.

The [AutomateAuthenticDesktop example](#)<sup>336</sup> (see the file *Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\>Examples
---	--

You can compile and run the project from within Visual Studio 2012/2013/2015/2017/2019/2022.

### Code snippet

```
// 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)
    {
visible
        // Make sure there's a running Authentic Desktop instance, and that it's
        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.

#### 14.3.1.2.3.4 Iterating through Open Documents

The code snippet below (from the [AutomateAuthenticDesktop example](#)<sup>336</sup>; 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 Browser View and Authentic View.

```

    // Handler for the "Toggle View Mode" button
    private void toggleView_Click(object sender, EventArgs e)
    {
visible
        // Make sure there's a running Authentic Desktop instance, and that it's
        StartAuthenticDesktop_Click(null, null);

        // Iterate through all open documents and toggle the current view between
Browser 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](#)<sup>336</sup> example is located in the C# subfolder of the API Examples folder:

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\Examples
---	---



You can compile and run the project from within Visual Studio 2012/2013/2015/2017/2019/2022.

### 14.3.1.2.3.5 Errors and COM Output Parameters

The code snippet below (from the [AutomateAuthenticDesktop example](#)<sup>336</sup>) shows how to handle errors and COM output parameters. The method [AuthenticDesktop.ActiveDocument.IsValid\(ref strErrorText, ref nErrorNumber, ref errorData\)](#)<sup>466</sup> uses output parameters that are used, in the code snippet below, to generate an error-message text.

The [AutomateAuthenticDesktop example](#)<sup>336</sup> (see the file *Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\Examples
---	---

You can compile and run the project from within Visual Studio 2012/2013/2015/2017/2019/2022.

### Code snippet

```
// 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);
}
}
}

```

### 14.3.1.2.3.6 Events

The code snippet below (from the [AutomateAuthenticDesktop example](#)<sup>336</sup>) lists the code for two event handlers. The [AutomateAuthenticDesktop example](#)<sup>336</sup> (see the file *Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\Examples
---	---

You can compile and run the project from within Visual Studio 2012/2013/2015/2017/2019/2022.

### Code snippet

```

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);
    }
}

```

#### 14.3.1.2.4 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](#)<sup>349</sup>.

## 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:  
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:

Interface	Java name
Document, method <code>SetEncoding</code>	<code>setFileEncoding</code>
AuthenticView, method <code>Goto</code>	<code>gotoElement</code>
AuthenticRange, method <code>Goto</code>	<code>gotoElement</code>
AuthenticRange, method <code>Clone</code>	<code>cloneRange</code>

## 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](#) <sup>349</sup>
- [Application Startup and Shutdown](#) <sup>352</sup>
- [Simple Document Access](#) <sup>353</sup>
- [Iterations](#) <sup>353</sup>
- [Use of Out-Parameters](#) <sup>354</sup>
- [Event Handlers](#) <sup>354</sup>

### 14.3.1.2.4.1 Example Java Project

The Authentic Desktop installation package contains an example Java project, located in the the `API\Java` subfolder of the `Examples` folder :

Windows 7, Windows 8, Windows 10, Windows 11	C:\Users\<<username>\Documents\ Altova\Authentic\2024\%APPNAME%\Examples
---	---

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)`.

<code>AltovaAutomation(_x64).dll</code>	Java-COM bridge: DLL part
<code>AltovaAutomation.jar</code>	Java-COM bridge: Java library part
<code>AuthenticAPI.jar</code>	Java classes of the Authentic Desktop API
<code>RunAuthenticDesktop.java</code>	Java example source code
<code>BuildAndRun.bat</code>	Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
<code>.classpath</code>	Eclipse project helper file
<code>.project</code>	Eclipse project file
<code>Authentic_JavaDoc.zip</code>	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](#)<sup>352</sup>: Starts Authentic Desktop, which is registered as an automation server, or activates Authentic Desktop if it is already running.
- [Open example files](#)<sup>353</sup>: Locates example documents installed with Authentic Desktop and opens them.
- [Iteration and Changing the View Mode](#)<sup>353</sup>: Changes the view of all open documents to Browser View. The code also shows how to iterate through open documents.
- [Iteration, validation, output parameters](#)<sup>354</sup>: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- [Event Handling](#)<sup>354</sup>: Shows how to handle Authentic Desktop events.

- [Shut down Authentic Desktop](#)<sup>352</sup>: 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 14 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.

```
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 view
10  * 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;
19
20         // Instead of COM error-handling, use Java exception mechanism.
21         try
22         {
23             // Start AuthenticDesktop as COM server.
24             authenticDesktop = new Application();
25             // COM servers start up invisible so we make it visible
```

```
25     authenticDesktop.setVisible(true);
26
27     // Locate samples installed with the product.
28     String strExamplesFolder = System.getenv("USERPROFILE") + "\\My Documents\\Altova\\
\\Authentic2012\\AuthenticExamples\\";
29
30     // Open two files from the product samples.
31     authenticDesktop.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf",
false);
32     authenticDesktop.getDocuments().openFile(strExamplesFolder + "ExpReport.xml",
false);
33
34     // Iterate through all open documents and set the View Mode to 'Text'.
35     for (Document doc:authenticDesktop.getDocuments())
36         if ( doc.getCurrentViewMode() != SPYViewModes.spyViewText)
37             doc.switchViewMode(SPYViewModes.spyViewText);
38
39     // An alternative iteration mode is index-based. COM indices are typically zero-
based.
40     Documents documents = authenticDesktop.getDocuments();
41     for (int i = 1; i <= documents.getCount(); i++)
42     {
43         Document doc = documents.getItem(i);
44
45         // Validation is one of the few methods that have output parameters.
46         // The class JVariant is the correct type for parameters in these cases.
47         // To get values back mark them with the by-reference flag.
48         JVariant validationErrorText = new JVariant.JStringVariant("");
validationErrorText.setByRefFlag();
49         JVariant validationErrorCount = new JVariant.JIntVariant(0);
validationErrorCount.setByRefFlag();
50         JVariant validationErrorXMLData = new JVariant.JIDispatchVariant(0);
validationErrorXMLData.setByRefFlag();
51         if (!doc.isValid(validationErrorText, validationErrorCount,
validationErrorXMLData))
52             System.out.println("Document " + doc.getName() + " is not wellformed - " +
validationErrorText.getStringValue());
53         else
54             System.out.println("Document " + doc.getName() + " is wellformed.");
55     }
56
57     // The following lines attach to the document events using a default
implementation
58     // for the events and override one of its methods.
59     // If you want to override all document events it is better to derive your
listener class
60     // from DocumentEvents and implement all methods of this interface.
61     Document doc = authenticDesktop.getActiveDocument();
62     doc.addListener(new DocumentEventsDefaultHandler()
63     {
64         @Override
65         public boolean onBeforeCloseDocument(Document i_ipDoc) throws
AutomationException
66         {
67             System.out.println("Document " + i_ipDoc.getName() + " requested closing.");
68
```

```
69         // Allow closing of document
70         return true;
71     }
72 });
73 doc.close(true);
74 doc = null;
75
76     System.out.println("Watch AuthenticDesktop!");
77 }
78 catch (AutomationException e)
79 {
80     // e.printStackTrace();
81 }
82 finally
83 {
84     // Make sure that AuthenticDesktop can shut down properly.
85     if (authenticDesktop != null)
86         authenticDesktop.dispose();
87
88     // Since the COM server was made visible and still is visible, it will keep
running
89     // and needs to be closed manually.
90     System.out.println("Now close AuthenticDesktop!");
91 }
92 }
93 }
```

#### 14.3.1.2.4.2 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*).

```
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 view
operations on it.
10  * Feel free to extend.
11  */
12 public class RunAuthenticDesktop
13 {
14     public static void main(String[] args)
15     {
16         // An instance of the application.
17         Application authenticDesktop = null;
```



```
18
19 // Instead of COM error-handling, use Java exception mechanism.
20 try
21 {
22     // Start AuthenticDesktop as COM server.
23     authenticDesktop = new Application();
24     // COM servers start up invisible so we make it visible
25     authenticDesktop.setVisible(true);
26
27 ...
28 }
29 }
30 }
```

## Application shutdown

The application can be shut down as shown below.

```
1 {
2     // Make sure that AuthenticDesktop can shut down properly.
3     if (authenticDesktop != null)
4         authenticDesktop.dispose();
5
6     // Since the COM server was made visible and still is visible, it will keep running
7     // and needs to be closed manually.
8     System.out.println("Now close AuthenticDesktop!");
9 }
```

### 14.3.1.2.4.3 Simple Document Access

The code listing below shows how to open a document.

```
1 // Locate samples installed with the product.
2 String strExamplesFolder = System.getenv("USERPROFILE") + "\\My Documents\\Altova\\
\\Authentic2012\\AuthenticExamples\\";
3
4 // Open two files from the product samples.
5 authenticDesktop.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
6 authenticDesktop.getDocuments().openFile(strExamplesFolder + "ExpReport.xml", false);
```

### 14.3.1.2.4.4 Iterations

The listing below shows how to iterate through open documents.

```
01 // Iterate through all open documents and set the View mode to 'Browser'.
02 for (Document doc:authenticDesktop.getDocuments())
03     if ( doc.getCurrentViewMode() != SPYViewModes.spyViewBrowser)
```

```
04         doc.switchViewMode(SPYViewModes.spyViewBrowser);
05
06 // 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     }
```

#### 14.3.1.2.4.5 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
06 // 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 // Validation is one of the few methods that have output parameters.
13 // The class JVariant is the correct type for parameters in these cases.
14 // To get values back, mark them with the by-reference flag.
15 JVariant validationErrorText = new JVariant.JStringVariant("");
validationErrorText.setByRefFlag();
16 JVariant validationErrorCount = new JVariant.JIntVariant(0);
validationErrorCount.setByRefFlag();
17 JVariant validationErrorXMLData = new JVariant.JIDispatchVariant(0);
validationErrorXMLData.setByRefFlag();
18     if (!doc.isValid(validationErrorText, validationErrorCount, validationErrorXMLData))
19         System.out.println("Document " + doc.getName() + " is not wellformed - " +
validationErrorText.getStringValue());
20     else
21         System.out.println("Document " + doc.getName() + " is wellformed.");
22 }
```

#### 14.3.1.2.4.6 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 AutomationException
10     {
11         System.out.println("Document " + i_ipDoc.getName() + " requested closing.");
12
13         // allow closing of document
14         return true;
15     }
16 });
17 doc.close(true);
18 doc = null;

```

## 14.3.2 Interfaces

### Object Hierarchy

- [Application](#) <sup>356</sup>
- [SpyProject](#) <sup>527</sup>
- [SpyProjectItems](#) <sup>531</sup>
- [SpyProjectItem](#) <sup>529</sup>
- [Documents](#) <sup>477</sup>
- [Document](#) <sup>445</sup>
- [GridView](#) <sup>510</sup>
- [AuthenticView](#) <sup>412</sup>
- [AuthenticRange](#) <sup>383</sup>
- [AuthenticDataTransfer](#) <sup>378</sup> (previously DocEditDataTransfer)
- [AuthenticDataTransfer](#) <sup>378</sup> (previously DocEditDataTransfer)
- [TextView](#) <sup>535</sup>
- [XMLData](#) <sup>576</sup>
- [Dialogs](#) <sup>362</sup>
- [CodeGeneratorDlg](#) <sup>429</sup>
- [FileSelectionDlg](#) <sup>490</sup>
- [SchemaDocumentationDlg](#) <sup>513</sup>
- [GenerateSampleXMLDlg](#) <sup>504</sup>
- [DTDSchemaGeneratorDlg](#) <sup>481</sup>
- [FindInFilesDlg](#) <sup>492</sup>
- [DatabaseConnection](#) <sup>435</sup>
- [ExportSettings](#) <sup>488</sup>
- [TextImportExportSettings](#) <sup>533</sup>
- [ElementList](#) <sup>486</sup>
- [ElementListItem](#) <sup>487</sup>

### Enumerations <sup>588</sup>

### 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.

### 14.3.2.1 Application

#### Methods

[GetDatabaseImportElementList](#) <sup>363</sup>

[GetDatabaseSettings](#) <sup>363</sup>

[GetDatabaseTables](#) <sup>364</sup>

[ImportFromDatabase](#) <sup>367</sup>

[CreateXMLSchemaFromDBStructure](#) <sup>361</sup>

[GetTextImportElementList](#) <sup>365</sup>

[GetTextImportExportSettings](#) <sup>366</sup>

[ImportFromText](#) <sup>368</sup>

[ImportFromWord](#) <sup>369</sup>

[ImportFromSchema](#) <sup>368</sup>

[GetExportSettings](#) <sup>364</sup>

[NewProject](#) <sup>370</sup>

[OpenProject](#) <sup>371</sup>

[AddMacroMenuItem](#) <sup>360</sup>

[ClearMacroMenu](#) <sup>361</sup>

[ShowForm](#) <sup>374</sup>

[ShowApplication](#) <sup>373</sup>

[URLDelete](#) <sup>375</sup>

[URLMakeDirectory](#) <sup>375</sup>

[AddXSLT\\_XQParameter](#) <sup>360</sup>

[GetXSLT\\_XQParameterCount](#) <sup>366</sup>

[GetXSLT\\_XQParameterName](#) <sup>366</sup>

[GetXSLT\\_XQParameterXPath](#) <sup>366</sup>

[RemoveXSLT\\_XQParameter](#) <sup>372</sup>

[FindInFiles](#) <sup>362</sup>

[Quit](#) <sup>372</sup>

#### Properties

[Application](#) <sup>360</sup>

[Parent](#) <sup>371</sup>

[ActiveDocument](#) <sup>359</sup>  
[Documents](#) <sup>362</sup>

[CurrentProject](#) <sup>361</sup>

[Dialogs](#) <sup>362</sup>

[WarningNumber](#) <sup>376</sup>  
[WarningText](#) <sup>376</sup>

[Status](#) <sup>374</sup>

[MajorVersion](#) <sup>370</sup>

[MinorVersion](#) <sup>370</sup>

[Edition](#) <sup>362</sup>

[IsAPISupported](#) <sup>370</sup>

[ServicePackVersion](#) <sup>373</sup>

### Description

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")
```

## 14.3.2.1.1 Events

### 14.3.2.1.1.1 OnBeforeOpenDocument

**Event:** OnBeforeOpenDocument(*objDialog* as [FileSelectionDlg](#) <sup>490</sup>)

### 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](#) <sup>491</sup> property of *io\_objDialog* at its default value [spyDialogOK](#) <sup>591</sup>. To abort the opening of the document set this property to [spyDialogCancel](#) <sup>591</sup>.

### 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

**14.3.2.1.1.2 OnBeforeOpenProject**

**Event:** OnBeforeOpenProject(*objDialog* as [FileSelectionDlg](#)<sup>490</sup>)

**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](#)<sup>491</sup> property of *io\_objDialog* at its default value [spyDialogOK](#)<sup>591</sup>. To abort the opening of the project set this property to [spyDialogCancel](#)<sup>591</sup>.

**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

**14.3.2.1.1.3 OnDocumentOpened**

**Event:** OnDocumentOpened(*objDocument* as [Document](#)<sup>445</sup>)

**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)
```

```
{
}
```

**XMLSpy IDE Plugin:**

IXMLSpyPlugIn.OnEvent (7, ...) // nEventId = 7

### 14.3.2.1.1.4 OnProjectOpened

**Event:** OnProjectOpened(*objProject* as [SpyProject](#)<sup>527</sup>)

**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:**

```
Function On_OpenProject(objProject)
End Function
```

**XMLSpy scripting environment - JScript:**

```
function On_OpenProject(objProject)
{
}
```

**XMLSpy IDE Plugin:**

IXMLSpyPlugIn.OnEvent (6, ...) // nEventId = 6

### 14.3.2.1.2 ActiveDocument

**Property:** ActiveDocument as [Document](#)<sup>445</sup>

**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.

### 14.3.2.1.3 AddMacroMenuItem

**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 [Example Scripting Project](#)<sup>295</sup>.

#### 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.

### 14.3.2.1.4 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.

### 14.3.2.1.5 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.



### 14.3.2.1.6 ClearMacroMenu

**Method:** ClearMacroMenu()

**Return Value**

None

**Description**

Removes from the **Tools** menu those menu items that were added by calling [AddMenuItem](#)<sup>360</sup>. See also [Example Scripting Project](#)<sup>295</sup>.

**Errors**

1111	The application object is no longer valid.
------	--

### 14.3.2.1.7 CreateXMLSchemaFromDBStructure

**Method:** CreateXMLSchemaFromDBStructure(pImportSettings as [DatabaseConnection](#)<sup>435</sup>, pTables as [ElementList](#)<sup>486</sup>)

**Description**

CreateXMLSchemaFromDBStructure creates from a database specified in pImportSettings for the defined tables in pTables new XML Schema document(s) describing the database tables structure.

The parameter pTables 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](#)<sup>364</sup>.

**Errors**

1112	Invalid database specified.
1120	Database import failed.

### 14.3.2.1.8 CurrentProject

**Property:** CurrentProject as [SpyProject](#)<sup>527</sup>

**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.

### 14.3.2.1.9 Dialogs

**Property:** Dialogs as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.1.10 Documents

**Property:** Documents as [Documents](#)<sup>477</sup>

#### Description

Collection of all open documents.

#### Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

### 14.3.2.1.11 Edition

**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.

### 14.3.2.1.12 FindInFiles

**Method:** FindInFiles(*pSettings* as [FindInFilesDlg](#)<sup>492</sup>) as [FindInFilesResults](#)<sup>503</sup>

#### Description

Returns a [FindInFilesResults](#)<sup>503</sup> 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.

### 14.3.2.1.13 GetDatabaseImportElementList

**Method:** GetDatabaseImportElementList(*plImportSettings* as [DatabaseConnection](#)<sup>435</sup>) as [ElementList](#)<sup>486</sup>

**Description**

The function returns a collection of [ElementListItem](#)s where the properties [ElementListItem.Name](#)<sup>487</sup> contain the names of the fields that can be selected for import and the properties [ElementListItem.ElementKind](#)<sup>487</sup> are initialized either to [spyXMLDataAttr](#) or [spyXMLDataElement](#), depending on the value passed in [DatabaseConnection.AsAttributes](#)<sup>436</sup>. This list serves as a filter to what finally gets imported by a future call to [ImportFromDatabase](#)<sup>367</sup>. Use [ElementList.RemoveElement](#)<sup>486</sup> to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of [DatabaseConnection.File](#)<sup>438</sup>, [DatabaseConnection.ADOConnection](#)<sup>436</sup> and [DatabaseConnection.ODBCConnection](#)<sup>439</sup>, as well as [DatabaseConnection.SQLSelect](#)<sup>440</sup>. Use the property [DatabaseConnection.AsAttributes](#)<sup>436</sup> to initialize [ElementListItem.ElementKind](#)<sup>487</sup> of the resulting element list to either [spyXMLDataAttr](#) or [spyXMLDataElement](#), respectively.

**Example**

See example at [ImportFromDatabase](#)<sup>367</sup>.

**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.

### 14.3.2.1.14 GetDatabaseSettings

**Method:** GetDatabaseSettings() as [DatabaseConnection](#)<sup>435</sup>

**Description**

GetDatabaseSettings creates a new object of database settings. The object is used to specify database connection parameters for the methods [GetDatabaseTables](#)<sup>364</sup>, [GetDatabaseImportElementList](#)<sup>363</sup>, [ImportFromDatabase](#)<sup>367</sup>, [ImportFromSchema](#)<sup>368</sup> and [ExportToDatabase](#)<sup>458</sup>.

**Example**

See example of [ImportFromDatabase](#)<sup>367</sup>.

**Errors**

1111	The application object is no longer valid.
------	--

1100	Invalid address for the return parameter was specified.
------	---

### 14.3.2.1.15 GetDatabaseTables

**Method:** GetDatabaseTables(*plmpSettings* as [DatabaseConnection](#)<sup>356</sup>) as [ElementList](#)<sup>486</sup>

#### Description

GetDatabaseTables reads the table names from the database specified in *plmpSettings*. Properties mandatory to be filled out for the database connection are one of [DatabaseConnection.File](#)<sup>438</sup>, [DatabaseConnection.ADOConnection](#)<sup>436</sup> and [DatabaseConnection.ODBCConnection](#)<sup>439</sup>. All other properties are ignored.

The function returns a collection of [ElementListItem](#)s where the properties [ElementListItem.Name](#)<sup>487</sup> contain the names of tables stored in the specified database. The remaining properties of [ElementListItem](#)<sup>487</sup> 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

```

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

```

### 14.3.2.1.16 GetExportSettings

**Method:** GetExportSettings() as [ExportSettings](#)<sup>488</sup> (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](#)<sup>445</sup>.

**Errors**

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

### 14.3.2.1.17 GetTextImportElementList

**Method:** GetTextImportElementList(plmportSettings as [TextImportExportSettings](#)<sup>533</sup>) as [ElementList](#)<sup>486</sup>

**Description**

GetTextImportElementList retrieves importing information about the text-file as specified in plmportSettings. The function returns a collection of ElementListItem where the properties [ElementListItem.Name](#)<sup>487</sup> 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](#)<sup>535</sup> 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 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
```

### 14.3.2.1.18 GetTextImportExportSettings

**Method:** GetTextImportExportSettings() as [TextImportExportSettings](#)<sup>533</sup> (read-only)

#### Description

GetTextImportExportSettings creates a new object of common import and export settings for text files. See also the example for [Application.GetTextImportElementList](#)<sup>365</sup>.

#### Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

### 14.3.2.1.19 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.

### 14.3.2.1.20 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.

### 14.3.2.1.21 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.

14.3.2.1.22 ImportFromDatabase

**Method:** ImportFromDatabase(plmportSettings as [DatabaseConnection](#)<sup>435</sup>, pElementList as [ElementList](#)<sup>486</sup>) as [Document](#)<sup>445</sup>

**Return Value**

Creates a new document containing the data imported from the database.

**Description**

ImportFromDatabase imports data from a database as specified in plmportSettings and creates a new document containing the data imported from the database. Properties mandatory to be filled out are one of [DatabaseConnection.File](#)<sup>438</sup>, [DatabaseConnection.ADOConnection](#)<sup>436</sup> or [DatabaseConnection.ODBCConnection](#)<sup>439</sup> and [DatabaseConnection.SQLSelect](#)<sup>440</sup>. Additionally, you can use [DatabaseConnection.AsAttributes](#)<sup>436</sup>, [DatabaseConnection.ExcludeKeys](#)<sup>438</sup>, [DatabaseConnection.IncludeEmptyElements](#)<sup>439</sup> and [NumberDateTimeFormat](#)<sup>439</sup> 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](#)<sup>363</sup> and [GetDatabaseImportElementList](#)<sup>363</sup> 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**

```
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
```

### 14.3.2.1.23 ImportFromSchema

**Method:** ImportFromSchema(*pImportSettings* as [DatabaseConnection](#)<sup>435</sup>, *strTable* as String, *pSchemaDoc* as [Document](#)<sup>445</sup>) as [Document](#)<sup>445</sup>

#### 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](#)<sup>438</sup>, [DatabaseConnection.ADOConnection](#)<sup>436</sup> or [DatabaseConnection.ODBCConnection](#)<sup>439</sup>. Additionally, you can use [DatabaseConnection.AsAttributes](#)<sup>436</sup>, [DatabaseConnection.ExcludeKeys](#)<sup>438</sup> and [NumberDateTimeFormat](#)<sup>439</sup> to further parameterize import. All other properties get ignored.

ImportFromSchema does not use an 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](#)<sup>363</sup> and [GetDatabaseTables](#)<sup>364</sup> 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.

### 14.3.2.1.24 ImportFromText

**Method:** ImportFromText(*pImportSettings* as [TextImportExportSettings](#)<sup>533</sup>, *pElementList* as [ElementList](#)<sup>486</sup>) as [Document](#)<sup>445</sup>



**Description**

ImportFromText imports the text file as specified in plmpSettings. The parameter pElementList can be used as import filter. Either pass the list returned by a previous call to [GetTextImportElementList](#)<sup>365</sup> or null to import all columns. To avoid import of unnecessary columns use [ElementList.RemoveElement](#)<sup>486</sup> 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
```

14.3.2.1.25 [ImportFromWord](#)

**Method:** ImportFromWord(*strFile* as String) as [Document](#)<sup>445</sup>

**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.

### 14.3.2.1.26 IsAPISupported

**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.

### 14.3.2.1.27 MajorVersion

**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.

### 14.3.2.1.28 MinorVersion

**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.

### 14.3.2.1.29 NewProject

**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 <i>bDiscardCurrent</i> is <i>true</i> .
1103	Creation of new project failed.

### 14.3.2.1.30 OpenProject

**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 possibly 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 <i>bDiscardCurrent</i> is <i>true</i> .

### 14.3.2.1.31 Parent

**Property:** Parent as [Application](#)<sup>356</sup> (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.

### 14.3.2.1.32 Quit

**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.
------	--

### 14.3.2.1.33 ReloadSettings

**Method:** ReloadSettings

**Return Value**

**Description**

The application settings are reloaded from the registry.

Available with TypeLibrary version 1.5

**Errors**

1111	The application object is no longer valid.
------	--

### 14.3.2.1.34 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.

### 14.3.2.1.35 RunMacro

**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.
------	--

### 14.3.2.1.36 ScriptingEnvironment

**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.

### 14.3.2.1.37 ServicePackVersion

**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.

### 14.3.2.1.38 ShowApplication

**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.
------	--

**14.3.2.1.39 ShowFindInFiles**

**Method:** ShowFindInFiles(pSettings as [FindInFilesDlg](#)<sup>492</sup>) 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.

**14.3.2.1.40 ShowForm**

**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.

**14.3.2.1.41 Status**

**Property:** Status as [ENUMApplicationStatus](#)<sup>588</sup>

**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.

### 14.3.2.1.42 URLDelete

**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.

### 14.3.2.1.43 URLMakeDirectory

**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.

### 14.3.2.1.44 Visible

**Property:** Visible as VARIANT\_BOOL

**Description**

Sets or gets the visibility attribute of Authentic Desktop. This standard automation property makes usage of [ShowApplication](#) <sup>373</sup> obsolete.

**Errors**

1110	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

### 14.3.2.1.45 WarningNumber

**Property:** WarningNumber as integer

#### Description

Some methods fill the property WarningNumber with additional information if an error occurs.

Currently just [Documents.OpenFile](#)<sup>480</sup> fills this property.

#### Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

### 14.3.2.1.46 WarningText

**Property:** WarningText as String

#### Description

Some methods fill the property WarningText with additional information if an error occurs.

Currently just [Documents.OpenFile](#)<sup>480</sup> fills this property.

#### Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

## 14.3.2.2 AuthenticContextMenu

The context menu interface provides the means for the user to customize the context menus shown in Authentic. The interface has the methods listed in this section.

### 14.3.2.2.1 CountItems

**Method:** CountItems () nItems as long

#### Return Value

Returns the number of menu items.

#### Errors

2501	Invalid object.
------	-----------------



### 14.3.2.2 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

### 14.3.2.2.3 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

### 14.3.2.2.4 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

### 14.3.2.2.5 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

### 14.3.2.3 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.

#### Methods

[getData](#) <sup>379</sup>

#### Properties

[dropEffect](#) <sup>379</sup>

[ownDrag](#) <sup>379</sup>

[type](#) <sup>379</sup>

#### 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 IUnkown interface.

### 14.3.2.3.1 dropEffect

**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.
------	---

### 14.3.2.3.2 getData

**Method:** getData() as Variant

**Description**

Retrieve the data associated with the dragged object. Depending on [AuthenticDataTransfer.type](#)<sup>379</sup>, that data is either a string or a COM interface pointer of type IUnknown.

**Errors**

2101	Invalid address for the return parameter was specified.
------	---

### 14.3.2.3.3 ownDrag

**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.
------	---

### 14.3.2.3.4 type

**Property:** type as String (read-only)

**Description**

Holds the type of data you get with the [DocEditDataTransfer.getData](#)<sup>379</sup> 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.
------	---

### 14.3.2.4 AuthenticEventContext

The `EventContext` interface gives access to many properties of the context in which a macro is executed.

#### 14.3.2.4.1 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 a 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.

#### 14.3.2.4.2 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.

### 14.3.2.4.3 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.

### 14.3.2.4.4 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

### 14.3.2.4.5 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.

### 14.3.2.4.6 IsAvailable

**Method:** IsAvailable () as Boolean

**Return Value**

Returns true if `EventContext` is set, false otherwise.

**Errors**

2201	Invalid object.
------	-----------------

### 14.3.2.4.7 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

### 14.3.2.5 AuthenticRange

The first table lists the properties and methods of AuthenticRange that can be used to navigate through the document and select specific portions.

Properties	Methods	
<a href="#">Application</a> <sup>385</sup>	<a href="#">Clone</a> <sup>387</sup>	<a href="#">MoveBegin</a> <sup>405</sup>
<a href="#">FirstTextPosition</a> <sup>390</sup>	<a href="#">CollapsToBegin</a> <sup>387</sup>	<a href="#">MoveEnd</a> <sup>405</sup>
<a href="#">FirstXMLData</a> <sup>391</sup>	<a href="#">CollapsToEnd</a> <sup>387</sup>	<a href="#">NextCursorPosition</a> <sup>396</sup>
<a href="#">FirstXMLDataOffset</a> <sup>392</sup>	<a href="#">ExpandTo</a> <sup>390</sup>	<a href="#">PreviousCursorPosition</a> <sup>397</sup>
<a href="#">LastTextPosition</a> <sup>402</sup>	<a href="#">Goto</a> <sup>395</sup>	<a href="#">Select</a> <sup>408</sup>
<a href="#">LastXMLData</a> <sup>403</sup>	<a href="#">GotoNext</a> <sup>395</sup>	<a href="#">SelectNext</a> <sup>408</sup>
<a href="#">LastXMLDataOffset</a> <sup>404</sup>	<a href="#">GotoPrevious</a> <sup>397</sup>	<a href="#">SelectPrevious</a> <sup>409</sup>
<a href="#">Parent</a> <sup>406</sup>	<a href="#">IsEmpty</a> <sup>400</sup>	<a href="#">SetFromRange</a> <sup>411</sup>
	<a href="#">IsEqual</a> <sup>400</sup>	

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
<a href="#">Text</a> <sup>411</sup>	<a href="#">Copy</a> <sup>387</sup>	<a href="#">AppendRow</a> <sup>385</sup>
	<a href="#">Cut</a> <sup>388</sup>	<a href="#">DeleteRow</a> <sup>388</sup>
	<a href="#">Delete</a> <sup>388</sup>	<a href="#">DuplicateRow</a> <sup>389</sup>
	<a href="#">IsCopyEnabled</a> <sup>399</sup>	<a href="#">InsertRow</a> <sup>399</sup>
	<a href="#">IsCutEnabled</a> <sup>400</sup>	<a href="#">IsFirstRow</a> <sup>401</sup>
	<a href="#">IsDeleteEnabled</a> <sup>400</sup>	<a href="#">IsInDynamicTable</a> <sup>401</sup>
	<a href="#">IsPasteEnabled</a> <sup>401</sup>	<a href="#">IsLastRow</a> <sup>401</sup>
	<a href="#">Paste</a> <sup>406</sup>	<a href="#">MoveRowDown</a> <sup>406</sup>

		<a href="#">MoveRowUp</a> <sup>406</sup>
--	--	--

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows		
Elements	Attributes	Entities
<a href="#">CanPerformActionWith</a> <sup>386</sup>	<a href="#">GetElementAttributeValue</a> <sup>393</sup>	<a href="#">GetEntityNames</a> <sup>394</sup>
<a href="#">CanPerformAction</a> <sup>386</sup>	<a href="#">GetElementAttributeNames</a> <sup>393</sup>	<a href="#">InsertEntity</a> <sup>398</sup>
<a href="#">PerformAction</a> <sup>407</sup>	<a href="#">GetElementHierarchy</a> <sup>394</sup>	
	<a href="#">HasElementAttribute</a> <sup>398</sup>	
	<a href="#">IsTextStateApplied</a> <sup>402</sup>	
	<a href="#">SetElementAttributeValue</a> <sup>410</sup>	

## 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](#)<sup>427</sup>, 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](#)<sup>422</sup>, [AuthenticView.DocumentEnd](#)<sup>422</sup>, and [AuthenticView.WholeDocument](#)<sup>429</sup>. The most flexible method is [AuthenticView.Goto](#)<sup>424</sup>, 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](#)<sup>390</sup> and [LastTextPosition](#)<sup>402</sup>. 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](#)<sup>391</sup>, [FirstXMLDataOffset](#)<sup>392</sup>, [LastXMLData](#)<sup>403</sup>, and [LastXMLDataOffset](#)<sup>404</sup>. This method is very efficient but requires knowledge of 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](#)<sup>411</sup> 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](#)<sup>388</sup>, [Copy](#)<sup>387</sup>, [Paste](#)<sup>406</sup> and [Delete](#)<sup>388</sup>.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the [XMLData](#)<sup>576</sup> objects of the underlying document to modify them directly.

### 14.3.2.5.1 AppendRow

**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
```

### 14.3.2.5.2 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.5.3 CanPerformAction

**Method:** CanPerformAction (*eAction* as [SPYAuthenticActions](#)<sup>589</sup>, *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](#)<sup>386</sup>.

#### 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](#)<sup>407</sup>.

### 14.3.2.5.4 CanPerformActionWith

**Method:** CanPerformActionWith (*eAction* as [SPYAuthenticActions](#)<sup>589</sup>, *out\_arrElementNames* as Variant)

#### Description

PerformActionWith 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.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use [CanPerformActionWith](#)<sup>386</sup>.

#### 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](#)<sup>407</sup>.

### 14.3.2.5.5 Clone

**Method:** Clone() as [AuthenticRange](#)<sup>383</sup>

**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.

### 14.3.2.5.6 CollapsToBegin

**Method:** CollapsToBegin() as [AuthenticRange](#)<sup>383</sup>

**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.

### 14.3.2.5.7 CollapsToEnd

**Method:** CollapsToEnd() as [AuthenticRange](#)<sup>383</sup>

**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.

### 14.3.2.5.8 Copy

**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.

#### 14.3.2.5.9 Cut

**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.

#### 14.3.2.5.10 Delete

**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.

#### 14.3.2.5.11 DeleteRow

**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 - VBScript
' 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
```

### 14.3.2.5.12 DuplicateRow

**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 - VBScript
' 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
```

### 14.3.2.5.13 EvaluateXPath

**Method:** EvaluateXPath (strExpression as string) strValue as string

**Return Value**

The method returns a string

**Description**

`EvaluateXPath()` executes an XPath expression with the context node being the beginning of the range selection. The result is returned as a 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

#### 14.3.2.5.14 ExpandTo

**Method:** `ExpandTo` (*eKind* as [SPYAuthenticElementKind](#)<sup>590</sup>), as [AuthenticRange](#)<sup>383</sup>

#### 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.

#### 14.3.2.5.15 FirstTextPosition

**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](#)<sup>402</sup>. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the text position by 1, has the same effect as the cursor-right key. Decrementing the text position by 1 has the same effect as the cursor-left key.

If you set `FirstTextPosition` to a value greater than the current [LastTextPosition](#)<sup>402</sup>, [LastTextPosition](#)<sup>402</sup> 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

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
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
```

14.3.2.5.16 FirstXMLData

**Property:** FirstXMLData as [XMLData](#)<sup>576</sup>

**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](#)<sup>392</sup> attribute.

Whenever you set FirstXMLData to a new data object, [FirstXMLDataOffset](#)<sup>392</sup> 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](#)<sup>392</sup> selects a position greater then the current [LastXMLData](#)<sup>403</sup> / [LastXMLDataOffset](#)<sup>404</sup>, the latter gets moved to the new start position.

HINT: You can use the [FirstXMLData](#)<sup>391</sup> and [LastXMLData](#)<sup>403</sup> properties to directly access and manipulate the underlying XML document in those cases where the methods available with the [AuthenticRange](#)<sup>383</sup> 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
```

**14.3.2.5.17 FirstXMLDataOffset****Property:** FirstXMLDataOffset as Long**Description**

Set or get the cursor position offset inside [FirstXMLData](#)<sup>391</sup> element for the beginning of the range. Offset positions are based on the characters returned by the [Text](#)<sup>411</sup> 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](#)<sup>392</sup> selects a position after the current [LastXMLData](#)<sup>403</sup> / [LastXMLDataOffset](#)<sup>404</sup>, 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
    
```

### 14.3.2.5.18 GetElementAttributeNames

**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](#)<sup>393</sup> and [SetElementAttributeValue](#)<sup>410</sup>.

**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 [SetElementAttributeValue](#)<sup>410</sup>.

### 14.3.2.5.19 GetElementAttributeValue

**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](#)<sup>393</sup>, or [HasElementAttribute](#)<sup>398</sup>.

**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](#)<sup>410</sup>.

### 14.3.2.5.20 GetElementHierarchy

**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](#)<sup>393</sup>, 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.

### 14.3.2.5.21 GetEntityNames

**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](#)<sup>398</sup> 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](#)<sup>394</sup> and [InsertEntity](#)<sup>398</sup>.

### 14.3.2.5.22 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

### 14.3.2.5.23 Goto

**Method:** Goto (eKind as [SPYAuthenticElementKind](#)<sup>590</sup>, nCount as Long, eFrom as [SPYAuthenticDocumentPosition](#)<sup>589</sup>) as [AuthenticRange](#)<sup>383</sup>

**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. Invalid start position specified. Invalid address for the return parameter was specified.

### 14.3.2.5.24 GotoNext

**Method:** GotoNext (eKind as [SPYAuthenticElementKind](#)<sup>590</sup>) as [AuthenticRange](#)<sup>383</sup>

**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
Wend

```

**14.3.2.5.25 GotoNextCursorPosition**

**Method:** GotoNextCursorPosition() as [AuthenticRange](#) <sup>383</sup>

**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.

### 14.3.2.5.26 GotoPrevious

**Method:** GotoPrevious (eKind as [SPYAuthenticElementKind](#)<sup>590</sup>) as [AuthenticRange](#)<sup>383</sup>

**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.
2004	Target lies before beginning 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.GotoPrevious(spyAuthenticTag).Select
    If ((Err.number - vbObjecterror) = 2004) Then
        bBeginOfDocument = True
        Err.Clear
    ElseIf (Err.number <> 0) Then
        Err.Raise ' forward error
    End If
Wend
```

### 14.3.2.5.27 GotoPreviousCursorPosition

**Method:** GotoPreviousCursorPosition() as [AuthenticRange](#)<sup>383</sup>

**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.

### 14.3.2.5.28 HasElementAttribute

**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.

### 14.3.2.5.29 InsertEntity

**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](#)<sup>394</sup>.

#### 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

```

### 14.3.2.5.30 InsertRow

**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**

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
' 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

```

### 14.3.2.5.31 IsCopyEnabled

**Property:** IsCopyEnabled as Boolean (read-only)

**Description**

Checks if the copy 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.

### 14.3.2.5.32 IsCutEnabled

**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.

### 14.3.2.5.33 IsDeleteEnabled

**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.

### 14.3.2.5.34 IsEmpty

**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.

### 14.3.2.5.35 IsEqual

**Method:** IsEqual (*objCmpRange* as [AuthenticRange](#)<sup>383</sup>) 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.



### 14.3.2.5.36 IsFirstRow

**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 extent 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.

### 14.3.2.5.37 IsInDynamicTable

**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.

### 14.3.2.5.38 IsLastRow

**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 extent 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.

### 14.3.2.5.39 IsPasteEnabled

**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.

#### 14.3.2.5.40 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.

#### 14.3.2.5.41 IsTextStateApplied

**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.

#### 14.3.2.5.42 LastTextPosition

**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](#)<sup>390</sup>. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the text position by 1, has the same effect as the cursor-right key. Decreasing the text position by 1 has the same effect as the cursor-left key.

If you set LastTextPosition to a value less than the current [FirstTextPosition](#)<sup>390</sup>, [FirstTextPosition](#)<sup>390</sup> 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

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
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
```

14.3.2.5.43 LastXMLData

**Property:** LastXMLData as [XMLData](#)<sup>576</sup>

**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](#)<sup>404</sup> attribute.

Whenever you set LastXMLData to a new data object, [LastXMLDataOffset](#)<sup>404</sup> 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](#)<sup>404</sup>, select a position less then the current [FirstXMLData](#)<sup>391</sup> / [FirstXMLDataOffset](#)<sup>392</sup>, the latter gets moved to the new end position.

HINT: You can use the [FirstXMLData](#)<sup>391</sup> and [LastXMLData](#)<sup>403</sup> properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the [AuthenticRange](#)<sup>383</sup> 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.

#### 14.3.2.5.44 LastXMLDataOffset

**Property:** LastXMLDataOffset as Long

##### Description

Set or get the cursor position inside [LastXMLData](#)<sup>403</sup> element for the end of the range.

Offset positions are based on the characters returned by the [Text](#)<sup>411</sup> 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](#)<sup>403</sup> / [LastXMLDataOffset](#)<sup>404</sup> selects a position before [FirstXMLData](#)<sup>391</sup> / [FirstXMLDataOffset](#)<sup>392</sup>, 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
    objRange.Select()
Else
    MsgBox "Ooops"
End If
```

### 14.3.2.5.45 MoveBegin

**Method:** MoveBegin (eKind as [SPYAuthenticElementKind](#)<sup>590</sup>, nCount as Long) as [AuthenticRange](#)<sup>383</sup>

**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.

### 14.3.2.5.46 MoveEnd

**Method:** MoveEnd (eKind as [SPYAuthenticElementKind](#)<sup>590</sup>, nCount as Long) as [AuthenticRange](#)<sup>383</sup>

**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.
------	--

#### 14.3.2.5.47 MoveRowDown

**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.

#### 14.3.2.5.48 MoveRowUp

**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.

#### 14.3.2.5.49 Parent

**Property:** Parent as [AuthenticView](#)<sup>412</sup> (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.

#### 14.3.2.5.50 Paste

**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.

14.3.2.5.51 PerformAction

**Method:** PerformAction (eAction as [SPYAuthenticActions](#)<sup>589</sup>, 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](#)<sup>386</sup>.

**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).CollapsesToEnd().Select
Else
    MsgBox "Can't insert any elements at current position"
```

End If

### 14.3.2.5.52 Select

**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()
```

### 14.3.2.5.53 SelectNext

**Method:** SelectNext (eKind as [SPYAuthenticElementKind](#)<sup>590</sup>) as [AuthenticRange](#)<sup>383</sup>

#### 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
    
```

### 14.3.2.5.54 SelectPrevious

**Method:** GotoPrevious (eKind as [SPYAuthenticElementKind](#)<sup>590</sup>) as [AuthenticRange](#)<sup>383</sup>

**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
    End If
Wend
    
```

```

        Err.Clear
    Elseif (Err.number <> 0) Then
        Err.Raise ' forward error
    End If
Wend

```

### 14.3.2.5.55 SetElementAttributeValue

**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](#)<sup>393</sup>, or [HasElementAttribute](#)<sup>398</sup>.

#### 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

```

### 14.3.2.5.56 SetFromRange

**Method:** SetFromRange (objSrcRange as [AuthenticRange](#)<sup>383</sup>)

**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.

### 14.3.2.5.57 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

### 14.3.2.5.58 Text

**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](#)<sup>388</sup>, [Delete](#)<sup>388</sup> or [PerformAction](#)<sup>407</sup> instead.

### Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for a return parameter was specified.

## 14.3.2.6 AuthenticView

Properties	Methods	Events
<a href="#">Application</a> <sup>420</sup>	<a href="#">Goto</a> <sup>424</sup>	<a href="#">OnBeforeCopy</a> <sup>413</sup>
<a href="#">AsXMLString</a> <sup>420</sup>	<a href="#">IsRedoEnabled</a> <sup>425</sup>	<a href="#">OnBeforeCut</a> <sup>413</sup>
<a href="#">DocumentBegin</a> <sup>422</sup>	<a href="#">IsUndoEnabled</a> <sup>425</sup>	<a href="#">OnBeforeDelete</a> <sup>414</sup>
<a href="#">DocumentEnd</a> <sup>422</sup>	<a href="#">Print</a> <sup>426</sup>	<a href="#">OnBeforeDrop</a> <sup>414</sup>
<a href="#">Event</a> <sup>423</sup>	<a href="#">Redo</a> <sup>427</sup>	<a href="#">OnBeforePaste</a> <sup>415</sup>
<a href="#">MarkupVisibility</a> <sup>426</sup>	<a href="#">Undo</a> <sup>428</sup>	<a href="#">OnDragOver</a> <sup>415</sup>
<a href="#">Parent</a> <sup>426</sup>	<a href="#">UpdateXMLInstanceEntities</a> <sup>428</sup>	<a href="#">OnKeyboardEvent</a> <sup>416</sup>
<a href="#">Selection</a> <sup>427</sup>		<a href="#">OnMouseEvent</a> <sup>417</sup>
<a href="#">XMLDataRoot</a> <sup>429</sup>		<a href="#">OnSelectionChanged</a> <sup>418</sup>
<a href="#">WholeDocument</a> <sup>429</sup>		

### Description

AuthenticView and its child objects [AuthenticRange](#)<sup>383</sup> 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](#)<sup>383</sup>)

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](#)<sup>383</sup> to make navigation inside the document straightforward, and to allow for the flexible selection of logical text elements. Use the properties [DocumentBegin](#)<sup>422</sup>, [DocumentEnd](#)<sup>422</sup>, or [WholeDocument](#)<sup>429</sup> for simple selections, while using the [Goto](#)<sup>424</sup> method for more complex selections. To navigate relative to a given document range, see the methods and properties of the [AuthenticRange](#)<sup>383</sup> object.

### 14.3.2.6.1 Events

#### 14.3.2.6.1.1 *OnBeforeCopy*

**Event:** OnBeforeCopy() as Boolean

**Scripting environment - VBScript:**

```
Function On_AuthenticBeforeCopy()
    ' On_AuthenticBeforeCopy = False ' to disable operation
End Function
```

**Scripting environment - JScript:**

```
function On_AuthenticBeforeCopy()
{
    // return false; /* to disable operation */
}
```

**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.

#### 14.3.2.6.1.2 *OnBeforeCut*

**Event:** OnBeforeCut() as Boolean

**Scripting environment - VBScript:**

```
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.

**14.3.2.6.1.3 OnBeforeDelete**

**Event:** OnBeforeDelete() as Boolean

**Scripting environment - VBScript:**

```
Function On_AuthenticBeforeDelete()
    ' On_AuthenticBeforeDelete = False ' to disable operation
End Function
```

**Scripting environment - JScript:**

```
function On_AuthenticBeforeDelete()
{
    // return false; /* to disable operation */
}
```

**IDE Plugin:**

```
IXMLSpyPlugIn.OnEvent (22, ...) // nEventId = 22
```

**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.

**14.3.2.6.1.4 OnBeforeDrop**

**Event:** OnBeforeDrop (*i\_nXPos* as Long, *i\_nYPos* as Long, *i\_ipRange* as [AuthenticRange](#)<sup>383</sup>, *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, ...) // nEventId = 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.

#### 14.3.2.6.1.5 *OnBeforePaste*

**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)
{
    // return false; /* to disable operation */
}
```

**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.

#### 14.3.2.6.1.6 *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.

#### 14.3.2.6.1.7 *OnDragOver*

**Event:** *OnDragOver* (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as [SPYMouseEvent](#)<sup>594</sup>, *objRange* as [AuthenticRange](#)<sup>383</sup>, *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 *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 drag operation. Return *True* (or nothing) to continue normal operation.

**14.3.2.6.1.8 OnKeyboardEvent**

**Event:** OnKeyboardEvent (*eKeyEvent* as [SPYKeyEvent](#)<sup>593</sup>, *nKeyCode* as Long, *nVirtualKeyStatus* as Long) as Boolean

**Scripting environment - VBScript:**

```
Function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
    'On_AuthenticKeyboardEvent = True ' to cancel bubbling of event
End Function
```

**Scripting environment - JScript:**

```
function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
{
    // return true; /* to cancel bubbling of event */
}
```

**IDE Plugin:**

```
IXMLSpyPlugIn.OnEvent (30, ...) // nEventId = 30
```

**Description**

This event gets triggered for *WM\_KEYDOWN*, *WM\_KEYUP* and *WM\_CHAR* Windows messages.

The actual message type is available in the *eKeyEvent* parameter. The status of virtual keys is combined in the parameter *nVirtualKeyStatus*. Use the bit-masks defined in the enumeration datatype [SPYVirtualKeyMask](#)<sup>601</sup>, to test for the different keys or their combinations.



### 14.3.2.6.1.9 OnLoad

**Event:** OnLoad ()

**Description:** OnLoad can be used e.g. to restrict some AuthenticView functionality, as shown in the example below:

```
function On_AuthenticLoad( )
{
    // We are disabling all entry helpers in order to prevent user from manipulating XML tree
    AuthenticView.DisableElementEntryHelper();
    AuthenticView.DisableAttributeEntryHelper();

    // We are also disabling the markup buttons for the same purpose
    AuthenticView.SetToolBarButtonState( 'AuthenticMarkupSmall', authenticToolBarButtonDisabled );
    AuthenticView.SetToolBarButtonState( 'AuthenticMarkupLarge', authenticToolBarButtonDisabled );
    AuthenticView.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](#) <sup>418</sup>.

### 14.3.2.6.1.10 OnMouseEvent

**Event:** OnMouseEvent (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as [SPYMouseEvent](#) <sup>594</sup>, *objRange* as [AuthenticRange](#) <sup>383</sup>) 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](#) <sup>594</sup> 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 object 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*.

#### 14.3.2.6.1.11 *OnSelectionChanged*

**Event:** OnSelectionChanged (*objNewSelection* as [AuthenticRange](#)<sup>383</sup>)

##### **Scripting environment - VBScript:**

```
Function On_AuthenticSelectionChanged (objNewSelection)
End Function
```

##### **Scripting environment - JScript:**

```
function On_AuthenticSelectionChanged (objNewSelection)
{
}
}
```

##### **IDE Plugin:**

```
IXMLSpyPlugIn.OnEvent (23, ...) // nEventId = 23
```

##### **Description**

This event gets triggered whenever the selection in the user interface changes.

#### 14.3.2.6.1.12 *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
- AuthenticTableCenterCellVertically
- AuthenticTableAlignCellContentBottom
- AuthenticTableAlignCellContentLeft
- AuthenticTableCenterCellContent
- AuthenticTableAlignCellContentRight
- AuthenticTableJustifyCellContent
- AuthenticTableInsertTable
- AuthenticTableDeleteTable
- AuthenticTableProperties
- AuthenticAppendRow
- AuthenticInsertRow
- AuthenticDuplicateRow
- AuthenticMoveRowUp
- AuthenticMoveRowDown
- AuthenticDeleteRow
- AuthenticDefineEntities
- AuthenticXMLSignature

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.

#### 14.3.2.6.1.13 *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](#)<sup>418</sup>.

`OnToolBarButtonExecuted` is fired after the toolbar action was executed. It is useful e.g. to add update code, as shown in the example below:

```
//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.

### 14.3.2.6.1.14 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.

### 14.3.2.6.2 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.6.3 AsXMLString

**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.

### 14.3.2.6.4 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**

2000	Invalid object.
2005	Invalid parameter.

### 14.3.2.6.5 CreateXMLNode

**Method:** CreateXMLNode (*nKind* as [SPYXMLDataKind](#)<sup>601</sup>) as [XMLData](#)<sup>576</sup>

**Return Value**

The method returns the new [XMLData](#)<sup>576</sup> object.

**Description**

To create a new XMLData object use the CreateXMLNode() method.

**Errors**

2000	Invalid object.
2012	Cannot create XML node.

### 14.3.2.6.6 DisableAttributeEntryHelper

**Method:** DisableAttributeEntryHelper ()

**Description**

DisableAttributeEntryHelper () disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

**Errors**

2000	Invalid object.
------	-----------------

### 14.3.2.6.7 DisableElementEntryHelper

**Method:** DisableElementEntryHelper ()

**Description**

DisableElementEntryHelper () disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

**Errors**

2000	Invalid object.
------	-----------------

### 14.3.2.6.8 DisableEntityEntryHelper

**Method:** DisableEntityEntryHelper ()

**Description**

DisableEntityEntryHelper () disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

**Errors**

2000	Invalid object.
------	-----------------

### 14.3.2.6.9 DocumentBegin

**Property:** DocumentBegin as [AuthenticRange](#)<sup>383</sup> (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.

### 14.3.2.6.10 DocumentEnd

**Property:** DocumentEnd as [AuthenticRange](#)<sup>383</sup> (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.

### 14.3.2.6.11 DoNotPerformStandardAction

**Method:** DoNotPerformStandardAction ()

**Description**

DoNotPerformStandardAction() serves as cancel bubble for macros, and stops further execution after macro has finished.

**Errors**

2000	Invalid object.
------	-----------------

### 14.3.2.6.12 EvaluateXPath

**Method:** EvaluateXPath (XMLData as [XMLData](#)<sup>576</sup>, strExpression as string) strValue as string

**Return Value**

The method returns a string

**Description**

EvaluateXPath() executes an XPath expression with the given XML context node. The result is returned as a string, in the case of a sequence it is a space-separated string.

**Errors**

2000	Invalid object.
2005	Invalid parameter.
2008	Internal error.
2013	XPath error.

### 14.3.2.6.13 Event

**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.

### 14.3.2.6.14 EventContext

**Property:** EventContext() as EventContext

#### Description

EventContext property gives access to the running macros context. See the [EventContext](#)<sup>380</sup> interface description for more details.

#### Errors

2000	Invalid object.
------	-----------------

### 14.3.2.6.15 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](#)<sup>418</sup>). 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](#)<sup>590</sup>.

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.
2005	Invalid parameter.
2008	Internal error.
2014	Invalid button identifier.

### 14.3.2.6.16 Goto

**Method:** Goto (eKind as SPYAuthenticElementKind<sup>590</sup>, nCount as Long, eFrom as SPYAuthenticDocumentPosition<sup>589</sup>) as AuthenticRange<sup>383</sup>

#### 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 <i>spyAuthenticDocumentBegin</i> or <i>spyAuthenticDocumentEnd</i> . Invalid address for the return parameter 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

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
```

14.3.2.6.17 **IsRedoEnabled**

**Property:** IsRedoEnabled as Boolean (read-only)

**Description**

True if redo steps are available and [Redo](#)<sup>427</sup> is possible.

**Errors**

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

14.3.2.6.18 **IsUndoEnabled**

**Property:** IsUndoEnabled as Boolean (read-only)

**Description**

True if undo steps are available and [Undo](#)<sup>428</sup> is possible.

**Errors**

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

### 14.3.2.6.19 MarkupVisibility

**Property:** MarkupVisibility as [SPYAuthenticMarkupVisibility](#)<sup>590</sup>

**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.

### 14.3.2.6.20 Parent

**Property:** Parent as [Document](#)<sup>445</sup> (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.

### 14.3.2.6.21 Print

**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.
------	---

### 14.3.2.6.22 Redo

**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.

### 14.3.2.6.23 Selection

**Property:** Selection as [AuthenticRange](#)<sup>383</sup>

**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**

```
' -----
' 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.IsEqual(objAuthenticView.DocumentEnd)) Then
    objAuthenticView.Selection = objAuthenticView.DocumentBegin
Else
    ' objAuthenticView.Selection = objAuthenticView.Selection.GotoNextCursorPosition()
    ' or shorter:
    objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```

### 14.3.2.6.24 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](#) <sup>418</sup>). 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](#) <sup>590</sup>.

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.

#### 14.3.2.6.25 Undo

**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.

#### 14.3.2.6.26 UpdateXMLInstanceEntities

**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

```
// -----
// Scripting environment - JavaScript
// -----
if(Application.ActiveDocument && (Application.ActiveDocument.CurrentViewMode == 4))
{
    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();
}
}

```

### 14.3.2.6.27 WholeDocument

**Property:** WholeDocument as [AuthenticRange](#)<sup>383</sup> (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.

### 14.3.2.6.28 XMLDataRoot

**Property:** XMLDataRoot as [XMLData](#)<sup>576</sup> (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.

### 14.3.2.7 CodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Properties and Methods**

Standard automation properties

[Application](#)<sup>430</sup>

[Parent](#)<sup>434</sup>

Programming language selection properties

[ProgrammingLanguage](#) <sup>434</sup>

[TemplateFileName](#) <sup>435</sup>

Settings for C++ code

[CPPSettings\\_DOMType](#) <sup>430</sup>

[CPPSettings\\_LibraryType](#) <sup>432</sup>

[CPPSettings\\_UseMFC](#) <sup>432</sup>

[CPPSettings\\_GenerateVC6ProjectFile](#) <sup>431</sup>

[CPPSettings\\_GenerateVSProjectFile](#) <sup>431</sup>

Settings for C# code

[CSharpSettings\\_ProjectType](#) <sup>432</sup>

Dialog handling for above code generation properties

[PropertySheetDialogAction](#) <sup>434</sup>

Output path selection properties

[OutputPath](#) <sup>433</sup>

[OutputPathDialogAction](#) <sup>433</sup>

Presentation of result

[OutputResultDialogAction](#) <sup>433</sup>

### Description

Use this object to configure the generation of program code for schema files. The method [GenerateProgramCode](#) <sup>461</sup> expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

#### 14.3.2.7.1 Application

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Property:** Application as [Application](#) <sup>356</sup> (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.

#### 14.3.2.7.2 CPPSettings\_DOMType

**Property:** CPPSettings\_DOMType as [SPYDOMType](#) <sup>591</sup>

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.

### 14.3.2.7.3 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.

### 14.3.2.7.4 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.

### 14.3.2.7.5 CPPSettings\_GenerateVSProjectFile

**Property:** CSharpSettings\_GenerateVSProjectFile as [SPYProjectType](#)<sup>596</sup>

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.

**14.3.2.7.6 CPPSettings\_LibraryType**

**Property:** CPPSettings\_LibraryType as [SPYLibType](#) <sup>594</sup>

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.

**14.3.2.7.7 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.

**14.3.2.7.8 CSharpSettings\_ProjectType**

**Property:** CSharpSettings\_ProjectType as [SPYProjectType](#) <sup>596</sup>

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.
------	--

### 14.3.2.7.9 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.

### 14.3.2.7.10 OutputPathDialogAction

**Property:** OutputPathDialogAction as [SPYDialogAction](#)<sup>591</sup>

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](#)<sup>433</sup> 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.

### 14.3.2.7.11 OutputResultDialogAction

**Property:** OutputResultDialogAction as [SPYDialogAction](#)<sup>591</sup>

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.

### 14.3.2.7.12 Parent

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.7.13 ProgrammingLanguage

**Property:** ProgrammingLanguage as [ProgrammingLanguage](#)<sup>595</sup>

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](#)<sup>435</sup> 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.

### 14.3.2.7.14 PropertySheetDialogAction

**Property:** PropertySheetDialogAction as [SPYDialogAction](#)<sup>591</sup>

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.

### 14.3.2.7.15 TemplateFileName

**Property:** TemplateFileName 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](#)<sup>434</sup> property to its appropriate value.

#### Errors

2200	The object is no longer valid.
2201	Invalid address for the return parameter was specified.

## 14.3.2.8 DatabaseConnection

### Properties for import and export

[File](#)<sup>438</sup> or  
[ADOConnection](#)<sup>436</sup> or  
[ODBCConnection](#)<sup>439</sup>

### Properties for import only

[DatabaseKind](#)<sup>437</sup>  
[SQLSelect](#)<sup>440</sup>  
[AsAttributes](#)<sup>436</sup>  
[ExcludeKeys](#)<sup>438</sup>  
[IncludeEmptyElements](#)<sup>439</sup>  
[NumberDateTimeFormat](#)<sup>439</sup>  
[NullReplacement](#)<sup>439</sup>  
[CommentIncluded](#)<sup>436</sup>

### Properties for export only

[CreateMissingTables](#)<sup>437</sup>  
[CreateNew](#)<sup>437</sup>  
[TextFieldLen](#)<sup>441</sup>  
[DatabaseSchema](#)<sup>437</sup>

### Properties for XML Schema from DB Structure generation

[PrimaryKeys](#)<sup>440</sup>  
[ForeignKeys](#)<sup>438</sup>  
[UniqueKeys](#)<sup>441</sup>  
[SchemaExtensionType](#)<sup>440</sup>  
[SchemaFormat](#)<sup>440</sup>  
[ImportColumnsType](#)<sup>438</sup>

#### 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.

#### 14.3.2.8.1 ADOConnection

**Property:** ADOConnection as String

##### Description

The property ADOConnection contains a connection string. Either use this property or [ODBCConnection](#)<sup>439</sup> or [File](#)<sup>438</sup> 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
```

#### 14.3.2.8.2 AsAttributes

**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](#)<sup>363</sup>.

##### Errors

No error codes are returned.

#### 14.3.2.8.3 CommentIncluded

**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.

#### 14.3.2.8.4 CreateMissingTables

**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.

#### 14.3.2.8.5 CreateNew

**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](#)<sup>438</sup>. Default is false. This property is used only when exporting to databases.

**Errors**

No error codes are returned.

#### 14.3.2.8.6 DatabaseKind

**Property:** DatabaseKind as [SPYDatabaseKind](#)<sup>591</sup>

**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.

#### 14.3.2.8.7 DatabaseSchema

**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.

### 14.3.2.8.8 ExcludeKeys

**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.

### 14.3.2.8.9 File

**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](#)<sup>439</sup> or [ADOConnection](#)<sup>436</sup> to refer to the database.

**Errors**

No error codes are returned.

### 14.3.2.8.10 ForeignKeys

**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.

### 14.3.2.8.11 ImportColumnsType

**Property:** ImportColumnsType as [SPYImportColumnsType](#)<sup>593</sup>

**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.

#### 14.3.2.8.12 IncludeEmptyElements

**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.

#### 14.3.2.8.13 NullReplacement

**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.

#### 14.3.2.8.14 NumberDateTimeFormat

**Property:** NumberDateTimeFormat as [SPYNumberDateTimeFormat](#)<sup>595</sup>

**Description**

The property NumberDateTimeFormat sets the format of numbers and date- and time-values. Default is [spySystemLocale](#)<sup>595</sup>. This property is used only when importing from databases.

**Errors**

No error codes are returned.

#### 14.3.2.8.15 ODBCConnection

**Property:** ODBCConnection as String

**Description**

The property ODBCConnection contains a ODBC connection string. Either use this property or [ADOConnection](#)<sup>438</sup> or [File](#)<sup>438</sup> to refer to a database.

**Errors**

No error codes are returned.

### 14.3.2.8.16 PrimaryKeys

**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.

### 14.3.2.8.17 SchemaExtensionType

**Property:** SchemaExtensionType as [SPYSchemaExtensionType](#) <sup>598</sup>

#### 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.

### 14.3.2.8.18 SchemaFormat

**Property:** SchemaFormat as [SPYSchemaFormat](#) <sup>598</sup>

#### 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.

### 14.3.2.8.19 SQLSelect

**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.



### 14.3.2.8.20 TextFieldLen

**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.

### 14.3.2.8.21 UniqueKeys

**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.

## 14.3.2.9 Dialogs

### Properties and Methods

Standard automation properties

[Application](#)<sup>442</sup>

[Parent](#)<sup>443</sup>

Various dialog objects

[CodeGeneratorDlg](#)<sup>442</sup>

[FileSelectionDlg](#)<sup>442</sup>

[SchemaDocumentationDlg](#)<sup>443</sup>

[GenerateSampleXMLDlg](#)<sup>443</sup>

[DTDSchemaGeneratorDlg](#)<sup>443</sup>

[FindInFilesDlg](#)<sup>444</sup>

#### 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.

### 14.3.2.9.1 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.9.2 CodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

**Property:** CodeGeneratorDlg as [CodeGeneratorDlg](#)<sup>429</sup> (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.

### 14.3.2.9.3 FileSelectionDlg

**Property:** FileSelectionDlg as [FileSelectionDlg](#)<sup>490</sup> (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.

### 14.3.2.9.4 Parent

**Property:** Parent as [Application](#)<sup>356</sup> (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.

### 14.3.2.9.5 SchemaDocumentationDlg

**Property:** SchemaDocumentationDlg as [SchemaDocumentationDlg](#)<sup>513</sup> (read-only)

**Description**

Get a new instance of a dialog object that parameterizes generation of schema documentation. See [Document.GenerateSchemaDocumentation](#)<sup>462</sup> 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.

### 14.3.2.9.6 GenerateSampleXMLDlg

**Property:** GenerateSampleXMLDlg as [GenerateSampleXMLDlg](#)<sup>504</sup> (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](#)<sup>462</sup> 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.

### 14.3.2.9.7 DTDSchemaGeneratorDlg

**Property:** DTDSchemaGeneratorDlg as [DTDSchemaGeneratorDlg](#)<sup>481</sup> (read-only)

**Description**

Get a new instance of a dialog object that parameterizes generation of a schema or DTD. See [Document.GenerateDTDOrSchemaEx](#)<sup>461</sup> 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.

#### 14.3.2.9.8 FindInFilesDlg

**Property:** FindInFilesDlg as [FindInFilesDlg](#)<sup>492</sup> (read-only)

##### Description

Get a new instance of a dialog object that parameterizes the search (or replacement) of strings in files. See [Application.FindInFiles](#)<sup>362</sup> 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.

#### 14.3.2.9.9 WSDLDocumentationDlg

**Property:** WSDLDocumentationDlg as [WSDLDocumentationDlg](#)<sup>541</sup> (read-only)

##### Description

Get a new instance of a dialog object that parameterizes generation of WSDL documentation. See [Document.GenerateWSDLDocumentation](#)<sup>463</sup> 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.

#### 14.3.2.9.10 WSDL20DocumentationDlg

**Property:** WSDL20DocumentationDlg as [WSDL20DocumentationDlg](#)<sup>553</sup> (read-only)

##### Description

Get a new instance of a dialog object that parameterizes generation of WSDL 2.0 documentation. See [Document.GenerateWSDL20Documentation](#)<sup>463</sup> 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.

### 14.3.2.9.11 XBRLDocumentationDlg

**Property:** XBRLDocumentationDlg as [XBRLDocumentationDlg](#)<sup>565</sup> (read-only)

#### Description

Get a new instance of a dialog object that parameterizes generation of XBRL documentation. See [Document.GenerateXBRLDocumentation](#)<sup>463</sup> 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.

## 14.3.2.10 Document

The `Document` interface has the following properties and methods.

### Properties and Methods

#### Standard automation properties

[Application](#)<sup>450</sup>  
[Parent](#)<sup>469</sup>

#### Various document properties and methods

[AsXMLString](#)<sup>451</sup>  
[SetActiveDocument](#)<sup>472</sup>  
[Encoding](#)<sup>456</sup>  
[SetEncoding \(obsolete\)](#)<sup>472</sup>  
[Suggestions](#)<sup>473</sup>

#### XML validation

[IsValid](#)<sup>466</sup>  
[IsValidEx](#)<sup>467</sup>  
[SetExternalsIsValid](#)<sup>472</sup>  
[IsWellFormed](#)<sup>468</sup>

#### Document conversion and transformation

[AssignDTD](#)<sup>450</sup>  
[AssignSchema](#)<sup>450</sup>  
[AssignXSL](#)<sup>450</sup>  
[AssignXSLFO](#)<sup>451</sup>  
[ConvertDTDOrSchema](#)<sup>452</sup>  
[ConvertDTDOrSchemaEx](#)<sup>453</sup>  
[GenerateDTDOrSchema](#)<sup>460</sup>  
[GenerateDTDOrSchemaEx](#)<sup>461</sup>  
[FlattenDTDOrSchema](#)<sup>460</sup>  
[CreateSchemaDiagram](#)<sup>455</sup>  
[ExecuteXQuery](#)<sup>457</sup>

[TransformXSL](#) <sup>475</sup>  
[TransformXSLEx](#) <sup>475</sup>  
[TransformXSLFO](#) <sup>475</sup>  
[TransformXSLFOEx](#) <sup>475</sup>  
[GenerateProgramCode](#) <sup>461</sup> (Enterprise Edition only)  
[GenerateSchemaDocumentation](#) <sup>462</sup>  
[GenerateSampleXML](#) <sup>462</sup>  
[ConvertToWSDL20](#) <sup>454</sup>

#### *Document export*

[GetExportElementList](#) <sup>464</sup>  
[ExportToText](#) <sup>459</sup>  
[ExportToDatabase](#) <sup>458</sup>  
[CreateDBStructureFromXMLSchema](#) <sup>455</sup>  
[GetDBStructureList](#) <sup>464</sup>

#### *File saving and naming*

[FullName](#) <sup>460</sup>  
[Name](#) <sup>469</sup>  
[Path](#) <sup>469</sup>  
[GetPathName \(obsolete\)](#) <sup>465</sup>  
[SetPathName \(obsolete\)](#) <sup>473</sup>  
[Title](#) <sup>474</sup>  
[IsModified](#) <sup>466</sup>  
[Saved](#) <sup>470</sup>  
[SaveAs](#) <sup>470</sup>  
[Save](#) <sup>470</sup>  
[SaveInString](#) <sup>471</sup>  
[SaveToURL](#) <sup>471</sup>  
[Close](#) <sup>452</sup>

#### *View access*

[CurrentViewMode](#) <sup>455</sup>  
[SwitchViewMode](#) <sup>474</sup>  
[TextView](#) <sup>474</sup>  
[AuthenticView](#) <sup>451</sup>  
[GridView](#) <sup>465</sup>  
[DocEditView \(obsolete\)](#) <sup>456</sup>

#### *Access to XMLData*

[RootElement](#) <sup>469</sup>  
[DataRoot](#) <sup>456</sup>  
[CreateChild](#) <sup>454</sup>  
[UpdateViews](#) <sup>476</sup>  
[StartChanges](#) <sup>473</sup>  
[EndChanges](#) <sup>457</sup>  
[UpdateXMLData](#) <sup>476</sup>

## Document objects

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](#) <sup>359</sup>

[Application.Documents](#) <sup>362</sup>

Use one of the following methods to open a new document in Authentic Desktop:

[Documents.OpenFile](#) <sup>480</sup>

[Documents.OpenURL](#) <sup>480</sup>

[Documents.OpenURLDialog](#) <sup>481</sup>

[Documents.NewFile](#) <sup>479</sup>

[Documents.NewFileFromText](#) <sup>479</sup>

[SpyProjectItem.Open](#) <sup>530</sup>

[Application.ImportFromDatabase](#) <sup>367</sup>

[Application.ImportFromSchema](#) <sup>368</sup>

[Application.ImportFromText](#) <sup>368</sup>

[Application.ImportFromWord](#) <sup>369</sup>

[Document.ConvertDTDOrSchema](#) <sup>452</sup>

[Document.GenerateDTDOrSchema](#) <sup>460</sup>

## 14.3.2.10.1 Events

### 14.3.2.10.1.1 OnBeforeSaveDocument

**Event:** OnBeforeSaveDocument(*objDocument* as [Document](#) <sup>445</sup>, *objDialog* as [FileSelectionDlg](#) <sup>490</sup>)

#### **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.DialogAction](#)<sup>491</sup> property of *io\_objDialog* at its default value [spyDialogOK](#)<sup>591</sup>. To abort saving of the document set this property to [spyDialogCancel](#)<sup>591</sup>.

**14.3.2.10.1.2 OnBeforeCloseDocument**

**Event:** OnBeforeCloseDocument(*objDocument* as [Document](#)<sup>445</sup>) as Boolean

**XMLSpy scripting environment - VBScript:**

```
Function On_BeforeCloseDocument(objDocument)
    ' 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.

**14.3.2.10.1.3 OnBeforeValidate**

**Event:** OnBeforeValidate(*objDocument* as [Document](#)<sup>445</sup>, *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 [SetExternallsValid](#)<sup>472</sup> 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

#### 14.3.2.10.1.4 OnCloseDocument

**Event:** OnCloseDocument(*objDocument* as [Document](#)<sup>445</sup>)

**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.

#### 14.3.2.10.1.5 OnViewActivation

**Event:** OnViewActivation(*objDocument* as [Document](#)<sup>445</sup>, *eViewMode* as [SPYViewModes](#)<sup>600</sup>, *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](#)<sup>358</sup> event instead.

### 14.3.2.10.2 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.10.3 AssignDTD

**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.

### 14.3.2.10.4 AssignSchema

**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.

### 14.3.2.10.5 AssignXSL

**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.

14.3.2.10.6 AssignXSLFO

**Method:** AssignXSLFO (*strXSLFOFile* as String, *bDialog* as Boolean)

**Description**

The method places a reference to the XSLFO 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.

14.3.2.10.7 AsXMLString

**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.

14.3.2.10.8 AuthenticView

**Method:** AuthenticView as [AuthenticView](#)<sup>412</sup> (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
        End If
    Else
        MsgBox "No document is open"
    End If

```

#### 14.3.2.10.9 Close

**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.

#### 14.3.2.10.10 ConvertDTDOrSchema

**Method:** ConvertDTDOrSchema (*nFormat* as [SPYDTDSchemaFormat](#)<sup>592</sup>, *nFrequentElements* as [SPYFrequentElements](#)<sup>592</sup>)

#### 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](#)<sup>453</sup>.

**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: <i>DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead</i> and <i>Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead</i> .

14.3.2.10.11 [ConvertDTDOrSchemaEx](#)

**Method:** ConvertDTDOrSchemaEx (*nFormat* as [SPYDTDSchemaFormat](#)<sup>592</sup>, *nFrequentElements* as [SPYFrequentElements](#)<sup>592</sup>, *sOutputPath* as String, *nOutputPathDialogAction* as [SPYDialogAction](#)<sup>591</sup>)

**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: <i>DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead</i> and <i>Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead</i> .

### 14.3.2.10.12 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](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

#### 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 WSDL view, maybe it is not an '.wsdl' file.
1421	Feature is not available in this edition.
1433	WSDL 1.1 to WSDL 2.0 conversion failed.

### 14.3.2.10.13 CreateChild

**Method:** CreateChild (*nKind* as [SPYXMLDataKind](#)<sup>601</sup>) as [XMLData](#)<sup>576</sup>

#### Return Value

The method returns the new XMLData object.

#### Description

To create a new XMLData object use the CreateChild() method.

#### Errors

1400	The object is no longer valid.
1404	Cannot create XMLData object.
1407	Invalid address for the return parameter was specified.

### 14.3.2.10.14 CreateDBStructureFromXMLSchema

**Method:** CreateDBStructureFromXMLSchema (*pDatabase* as [DatabaseConnection](#)<sup>435</sup>, *pTables* as [ElementList](#)<sup>486</sup>, *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](#)<sup>464</sup>.

**Errors**

1429	Database selection missing.
1430	Document export failed.

### 14.3.2.10.15 CreateSchemaDiagram

**Method:** CreateSchemaDiagram (*nKind* as [SPYSchemaDefKind](#)<sup>597</sup>, *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](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

**Errors**

1400	The object is no longer valid.
1414	Failed to save diagram.
1415	Invalid schema definition type specified.

### 14.3.2.10.16 CurrentViewMode

**Method:** CurrentViewMode as [SPYViewModes](#)<sup>600</sup>

**Description**

The property holds the current view mode of the document. See also [Document.SwitchViewMode](#)<sup>474</sup>.

**Errors**

1400	The object is no longer valid.
------	--------------------------------

1407	Invalid address for the return parameter was specified.
------	---

### 14.3.2.10.17 DataRoot

**Property:** DataRoot as [XMLData](#)<sup>576</sup> (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](#)<sup>469</sup> to get the root element of the whole document including XML prolog data. If the [CurrentViewMode](#)<sup>455</sup> is not *spyViewGrid* or *spyViewAuthentic* an [UpdateXMLData](#)<sup>476</sup> may be necessary to get access to the latest [XMLData](#)<sup>576</sup>.

#### Errors

1400	The document object is no longer valid.
1407	Invalid address for the return parameter was specified.

### 14.3.2.10.18 DocEditView

**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.

### 14.3.2.10.19 Encoding

**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](#)<sup>455</sup> on how to detect a document's actual view mode.

This property makes the method [SetEncoding](#)<sup>472</sup> 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.

### 14.3.2.10.20 EndChanges

**Method:** EndChanges()

**Description**

Use the method EndChanges to display all changes since the call to [Document.StartChanges](#)<sup>473</sup>.

**Errors**

1400	The object is no longer valid.
------	--------------------------------

### 14.3.2.10.21 ExecuteXQuery

**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 <code>strXMLFileName</code> does not exist or is not valid.
------	--

### 14.3.2.10.22 ExportToDatabase

**Method:** `ExportToDatabase` (*pFromChild* as [XMLData](#)<sup>576</sup>, *pExportSettings* as [ExportSettings](#)<sup>488</sup>, *pDatabase* as [DatabaseConnection](#)<sup>435</sup>)

#### Description

`ExportToDatabase` exports the XML document starting with the element `pFromChild`. The parameter `pExportSettings` defines the behaviour of the export (see [Application.GetExportSettings](#)<sup>364</sup>). The parameter `pDatabase` specifies the destination of the export (see [Application.GetDatabaseSettings](#)<sup>363</sup>). [UpdateXMLData\(\)](#)<sup>476</sup> might be indirectly needed as you have to pass the [XMLData](#)<sup>576</sup> 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

```
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
```

### 14.3.2.10.23 ExportToText

**Method:** ExportToText (*pFromChild* as [XMLData](#)<sup>576</sup>, *pExportSettings* as [ExportSettings](#)<sup>488</sup>, *pTextSettings* as [TextImportExportSettings](#)<sup>533</sup>)

#### 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](#)<sup>464</sup> to learn about the data that will be exported. The parameter pExportSettings defines the specifics for the export. Set the property [ExportSettings.ElementList](#)<sup>488</sup> to the - possibly modified - list returned by [GetExportElementList](#)<sup>464</sup> 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](#)<sup>534</sup> before you call ExportToText. [UpdateXMLData\(\)](#)<sup>476</sup> might be indirectly needed as you have to pass the [XMLData](#)<sup>576</sup> 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.GetTextImportExportSettings
    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
```

### 14.3.2.10.24 FlattenDTDOrSchema

**Method:** FlattenDTDOrSchema (sOutputPath as String, nOutputPathDialogAction as [SPYDialogAction](#)<sup>591</sup>)

#### 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, (i) the components of all included schemas are added as global components of the active schema, and (ii) included schemas are deleted.

#### Errors

1400	The object is no longer valid.
1412	Error during conversion.

### 14.3.2.10.25 FullName

**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](#)<sup>465</sup> and [SetPathName](#)<sup>473</sup> obsolete.

#### Errors

1400	The document object is no longer valid.
1402	Empty string has been specified as full file name.

### 14.3.2.10.26 GenerateDTDOrSchema

**Method:** GenerateDTDOrSchema (nFormat as [SPYDTDSchemaFormat](#)<sup>592</sup>, nValuesList as integer, nDetection as [SPYTypeDetection](#)<sup>599</sup>, nFrequentElements as [SPYFrequentElements](#)<sup>592</sup>)

#### 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 the 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](#)<sup>461</sup>.

Note that this functionality is not available in ZIP View only. See [Document.CurrentViewMode](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

**Errors**

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.

### 14.3.2.10.27 [GenerateDTDOrSchemaEx](#)

**Method:** [GenerateDTDOrSchemaEx](#) ( *objDlg* as [DTDSchemaGeneratorDlg](#)<sup>481</sup> ) as [Document](#)<sup>445</sup>

**Description**

Use this method to automatically generate a DTD or schema for the current XML document. A

[DTDSchemaGeneratorDlg](#)<sup>481</sup> 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](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

**Errors**

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.

### 14.3.2.10.28 [GenerateProgramCode](#)

**Method:** [GenerateProgramCode](#) (*objDlg* as [CodeGeneratorDlg](#)<sup>429</sup>)

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](#)<sup>429</sup> 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

14.3.2.10.29 [GenerateSampleXML](#)

**Method:** [GenerateSampleXML](#) (*objDlg* as [GenerateSampleXMLDlg](#)<sup>504</sup>) as [Document](#)<sup>445</sup>

**Description**

Generates a sample XML if the document is a schema or DTD. Use [Dialogs.GenerateSampleXMLDlg](#)<sup>443</sup> to get an initialized set of options.

Available with TypeLibrary version 1.5

**Errors**

1400	The document object is no longer valid.
------	---

14.3.2.10.30 [GenerateSchemaDocumentation](#)

**Method:** [GenerateSchemaDocumentation](#) (*objDlg* as [SchemaDocumentationDlg](#)<sup>513</sup>)

**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](#)<sup>443</sup> to get an initialized set of options. As a minimum, you will need to set the property [SchemaDocumentationDlg.OutputFile](#)<sup>521</sup> before starting the generation process. Note that this functionality is limited to Schema View only. See [Document.CurrentViewMode](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

**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

### 14.3.2.10.31 GenerateWSDL20Documentation

**Method:** GenerateWSDL20Documentation (*objDlg* as [WSDL20DocumentationDlg](#)<sup>553</sup>)

#### 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](#)<sup>444</sup> to get an initialized set of options. As a minimum, you will need to set the property [WSDL20DocumentationDlg.OutputFile](#)<sup>559</sup> before starting the generation process. Note that this functionality is limited to WSDL View only. See [Document.CurrentViewMode](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

#### 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

### 14.3.2.10.32 GenerateWSDLDocumentation

**Method:** GenerateWSDLDocumentation (*objDlg* as [WSDLDocumentationDlg](#)<sup>541</sup>)

#### 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](#)<sup>444</sup> to get an initialized set of options. As a minimum, you will need to set the property [WSDLDocumentationDlg.OutputFile](#)<sup>547</sup> before starting the generation process. Note that this functionality is limited to WSDL View only. See [Document.CurrentViewMode](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

#### 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

### 14.3.2.10.33 GenerateXBRLDocumentation

**Method:** GenerateXBRLDocumentation (*objDlg* as [XBRLDocumentationDlg](#)<sup>565</sup>)

**Description**

Generate documentation for an XBRL file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use [Dialogs.XBRLDocumentationDlg](#)<sup>445</sup> to get an initialized set of options. As a minimum, you will need to set the property [XBRLDocumentationDlg.OutputFile](#)<sup>570</sup> before starting the generation process. Note that this functionality is limited to XBRL View only. See [Document.CurrentViewMode](#)<sup>455</sup> and [SPYViewModes](#)<sup>600</sup>.

**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

14.3.2.10.34 [GetDBStructureList](#)

**Method:** [GetDBStructureList](#) (*pDatabase* as [DatabaseConnection](#)<sup>435</sup>) as [ElementList](#)<sup>486</sup>

**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](#)s where the properties [ElementListItem.Name](#)<sup>487</sup> contain the names of the tables.

See also [CreateDBStructureFromXMLSchema](#)<sup>455</sup>.

**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.

14.3.2.10.35 [GetExportElementList](#)

**Method:** [GetExportElementList](#) (*pFromChild* as [XMLData](#)<sup>576</sup>, *pExportSettings* as [ExportSettings](#)<sup>488</sup>) as [ElementList](#)<sup>486</sup>

**Description**

[GetExportElementList](#) creates a collection of elements to export from the document, depending on the settings in *pExportSettings* and starting from the element *pFromChild*. The function returns a collection of [ElementListItem](#)s where the properties [ElementListItem.Name](#)<sup>487</sup> contain the names of the tables that can be exported from the document. The property [ElementListItem.FieldCount](#)<sup>487</sup> contains the number of columns in



the table. The property [ElementListItem.RecordCount](#)<sup>487</sup> contains the number of records in the table. The property [ElementListItem.ElementKind](#)<sup>487</sup> is unused. [UpdateXMLData\(\)](#)<sup>476</sup> might be indirectly needed as you have to pass the [XMLData](#)<sup>576</sup> 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.

14.3.2.10.36 [GetPathName](#) (obsolete)

**Superseded by** [Document.FullName](#)<sup>460</sup>

```
// ---- javascript sample ----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

**Method:** [GetPathName\(\)](#) as String

**Description**

The method [GetPathName](#) gets the path of the active document.

See also [Document.SetPathName](#)<sup>473</sup> (obsolete).

14.3.2.10.37 [GridView](#)

**Property:** [GridView](#) as [GridView](#)<sup>510</sup>

**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.

### 14.3.2.10.38 IsModified

**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.

### 14.3.2.10.39 IsValid

**Method:** HRESULT IsValid([in, out] VARIANT \*strError, [in, out] VARIANT \*nErrorPos, [in, out] VARIANT \*pBadData, [out,retval] VARIANT\_BOOL \*bValid);

#### 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.

#### Example

The following C++ code snippet provides an example of how to use the `IsValid` method.

```
#import "XMLSpy.tlb"
```

```
CComPtr< XMLSpyLib::IDocument12> ipDoc = ipXMLSpy->GetActiveDocument();
if ( ipDoc )
{
    // prepare in/out parameters for IsValid call
    CComVariant variantError;
    CComVariant variantErrorPos;
    CComVariant variantBadData;
    // IsValid always shows a dialog with the validation result. This cannot be turned
    off.
```

```

    bool bIsValid = ipDoc->IsValid( &variantError, &variantErrorPos, &variantBadData )
    == VARIANT_TRUE;

    if ( !bIsValid )
    {
        // retrieve values from out parameters
        CString strError = (V_VT( &variantError ) == VT_BSTR ?
V_BSTR( &variantError ) : _T( "" ));
        long npos = (V_VT( &variantErrorPos ) == VT_I4 ? V_I4( &variantErrorPos ) : -
1);
        CComQIPtr< XMLSpyLib::IXMLData > ipXMLBadData = (V_VT( &variantBadData ) ==
VT_DISPATCH ? V_DISPATCH( &variantBadData ) : nullptr);

        if ( ipXMLBadData )
            strError += CString( _T("\n\n Node: ") ) + (LPCWSTR)ipXMLBadData-
>GetName();

        if ( !strError.IsEmpty() )
            AfxMessageBox( "Validation failed - " + strError );
    }
}

```

### 14.3.2.10.40 IsValidEx

**Method:** IsValidEx (nXSDVersion as [SPYValidateXSDVersion](#)<sup>599</sup>, nErrorLimit as int, nErrorFormat as [SPYValidateErrorFormat](#)<sup>600</sup>, 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](#)<sup>599</sup> 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](#)<sup>600</sup> 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.
------	--------------------------

### Example

The following C++ code snippet provides an example of how to use the `IsValidEx` method.

```
#import "XMLSpy.tlb"

CComPtr< XMLSpyLib::IDocument12> ipDoc = ipXMLSpy->GetActiveDocument();
if ( ipDoc )
{
    CComVariant variantErrorEx;
    bool bIsValidEx = ipDoc->IsValidEx( XMLSpyLib::spyValidateXSDVersion_AutoDetect,
100, XMLSpyLib::SPYValidateErrorFormat_LongXML, &variantErrorEx ) == VARIANT_TRUE;

    // // retrieve values from out parameters
    CString strErrorEx = (V_VT( &variantErrorEx ) == VT_BSTR ?
V_BSTR( &variantErrorEx ) : _T( "" ));
    if ( !strErrorEx.IsEmpty() )
        AfxMessageBox( "Validation failed - " + strErrorEx );
}
```

#### 14.3.2.10.41 IsWellFormed

**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`.

### 14.3.2.10.42 Name

**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](#)<sup>460</sup>.

**Errors**

1400	The document object is no longer valid.
1407	Invalid address for the return parameter was specified.

### 14.3.2.10.43 Parent

**Property:** Parent as [Documents](#)<sup>477</sup> (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](#)<sup>356</sup> (read-only)

### 14.3.2.10.44 Path

**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](#)<sup>460</sup>.

**Errors**

1400	The document object is no longer valid.
1407	Invalid address for the return parameter was specified.

### 14.3.2.10.45 RootElement

**Property:** RootElement as [XMLData](#)<sup>576</sup> (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`<sup>456</sup> property. If the `CurrentViewMode`<sup>455</sup> is not `spyViewGrid` or `spyViewAuthentic` an `UpdateXMLData`<sup>476</sup> may be necessary to get access to the latest `XMLData`<sup>576</sup>.

**Errors**

1400	The document object is no longer valid.
1407	Invalid address for the return parameter was specified.

14.3.2.10.46 `Save`

**Method:** `Save()`

**Description**

The method writes any modifications of the document to the associated file. See also `Document.FullName`<sup>460</sup>.

**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.

14.3.2.10.47 `SaveAs`

**Method:** `SaveAs (strFileName as String)`

**Description**

Save the document to the file specified. If saving was successful, the `FullName`<sup>460</sup> 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.

14.3.2.10.48 `Saved`

**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`<sup>466</sup>.

**Errors**

1400	The document object is no longer valid.
1407	Invalid address for the return parameter was specified.

### 14.3.2.10.49 SaveInString

**Method:** SaveInString (*pData* as [XMLData](#)<sup>576</sup>, *bMarked* as Boolean) as String

**Parameters**

*pData*

XMLData element to start. Set *pData* to [Document.RootElement](#)<sup>469</sup> 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**

SaveInString starts at the element *pData* and converts the XMLData objects to a string representation.

[UpdateXMLData\(\)](#)<sup>476</sup> might be indirectly needed as you have to pass the [XMLData](#)<sup>576</sup> 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.

### 14.3.2.10.50 SaveToURL

**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.

### 14.3.2.10.51 SetActiveDocument

**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.
------	--------------------------------

### 14.3.2.10.52 SetEncoding (obsolete)

**Superseded by** [Document.Encoding](#) <sup>456</sup>

```
// ---- javascript sample ----
// instead of:
// Application.ActiveDocument.SetEncoding("UTF-16");
// use now:
Application.ActiveDocument.Encoding = "UTF-16";
```

**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

### 14.3.2.10.53 SetExternallsValid

**Method:** SetExternallsValid (*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](#)<sup>448</sup> handler.

Available with TypeLibrary version 1.5

**Errors**

1400	The object is no longer valid.
------	--------------------------------

14.3.2.10.54 [SetPathName](#) (obsolete)

**Superseded by** [Document.FullName](#)<sup>460</sup>

```
// ---- javascript sample ----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

**Method:** 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.

14.3.2.10.55 [StartChanges](#)

**Method:** StartChanges()

**Description**

After StartChanges is executed Authentic Desktop will not update its editor windows until [Document.EndChanges](#)<sup>457</sup> is called. This increases performance of complex tasks to the XML structure.

**Errors**

1400	The object is no longer valid.
------	--------------------------------

14.3.2.10.56 [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](#)<sup>536</sup> event.

**Errors**

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.

**14.3.2.10.57 SwitchViewMode**

**Method:** SwitchViewMode (*nMode* as [SPYViewModes](#)<sup>600</sup>) 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](#)<sup>455</sup>.

**Errors**

1400	The object is no longer valid.
1407	Invalid address for the return parameter was specified.
1417	Invalid view mode specified.

**14.3.2.10.58 TextView**

**Property:** TextView as [TextView](#)<sup>535</sup>

**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.

**14.3.2.10.59 Title**

**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](#)<sup>460</sup>.

**Errors**

1400	The document object is no longer valid.
1407	Invalid address for the return parameter was specified.

### 14.3.2.10.60 TransformXSL

**Method:** TransformXSL()

**Description**

TransformXSL processes the XML document via the associated XSL file. See [Document.AssignXSL](#)<sup>450</sup> 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.

### 14.3.2.10.61 TransformXSLEx

**Method:** TransformXSLEx(*nAction* as [SPYDialogAction](#)<sup>591</sup>)

**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](#)<sup>450</sup> 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.

### 14.3.2.10.62 TransformXSLFO

**Method:** TransformXSLFO()

**Description**

TransformXSLFO processes the XML document via the associated XSLFO file. See [AssignXSLFO](#)<sup>451</sup> 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.

### 14.3.2.10.63 TransformXSLFOEx

**Method:** TransformXSLFOEx(*nAction* as [SPYDialogAction](#)<sup>591</sup>, *string* as sOutputFilepath)<sup>591</sup>

**Description**

TransformXSLFOEx performs an XSL-FO transformation. It processes the XML document via the associated XSL-FO file. The parameter specifies whether a dialog asking for the result document name should pop up or not. See [Document.AssignXSLFO](#)<sup>451</sup> on how to place a reference to an XSL-FO file into the document.

**Errors**

1400	The document object is no longer valid.
1411	Error during transformation process.

### 14.3.2.10.64 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.

### 14.3.2.10.65 UpdateViews

**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.
------	---

### 14.3.2.10.66 UpdateXMLData

**Method:** UpdateXMLData() as Boolean

**Description**

The [XMLData](#)<sup>576</sup> 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](#)<sup>455</sup> is `spyViewGrid` or `spyViewAuthentic` because these views keep the [XMLData](#)<sup>576</sup> updated.

Available with TypeLibrary version 1.5

**Errors**

1400	The document object is no longer valid.
------	---

### 14.3.2.11 Documents

**Properties**

[Count](#) <sup>478</sup>

[Item](#) <sup>478</sup>

**Methods**

[NewAuthenticFile](#) <sup>478</sup>

[NewFile](#) <sup>479</sup>

[NewFileFromText](#) <sup>479</sup>

[OpenAuthenticFile](#) <sup>479</sup>

[OpenFile](#) <sup>480</sup>

[OpenURL](#) <sup>480</sup>

[OpenURLDialog](#) <sup>481</sup>

**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);
}
```

### 14.3.2.11.1 Count

**Property:** Count as long

#### Description

Count of open documents.

#### Errors

1600	Invalid Documents object
1601	Invalid input parameter

### 14.3.2.11.2 Item

**Method:** Item (*n* as long) as [Document](#)<sup>445</sup>

#### Description

Gets the document with the index *n* in this collection. Index is 1-based.

#### Errors

1600	Invalid Documents object
1601	Invalid input parameter

### 14.3.2.11.3 NewAuthenticFile

**Method:** NewAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as [Document](#)<sup>445</sup>

#### 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*.

#### 14.3.2.11.4 NewFile

**Method:** NewFile (*strFile* as String, *strType* as String) as [Document](#)<sup>445</sup>

##### 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.

#### 14.3.2.11.5 NewFileFromText

**Method:** NewFileFromText (*strText* as String, *strType* as String) as [Document](#)<sup>445</sup>

##### 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.

#### 14.3.2.11.6 OpenAuthenticFile

**Method:** OpenAuthenticFile (*strSPSPPath* as String, *strXMLPath* as String) as [Document](#)<sup>445</sup>

##### Parameters

*strSPSPPath*

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.

### 14.3.2.11.7 OpenFile

**Method:** OpenFile (*strPath* as String, *bDialog* as Boolean) as [Document](#)<sup>445</sup>

**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)
```

### 14.3.2.11.8 OpenURL

**Method:** OpenURL (*strURL* as String, *nURLType* as [SPYURLTypes](#)<sup>599</sup>, *nLoading* as [SPYLoading](#)<sup>594</sup>, *strUser* as String, *strPassword* as String) as [Document](#)<sup>445</sup>

**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.

### 14.3.2.11.9 OpenURLDialog

**Method:** OpenURLDialog (*strURL* as String, *nURLType* as [SPYURLTypes](#)<sup>599</sup>, *nLoading* as [SPYLoading](#)<sup>594</sup>, *strUser* as String, *strPassword* as String) as [Document](#)<sup>445</sup>

#### 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.

### 14.3.2.12 DTDSchemaGeneratorDlg

#### Properties and Methods

Standard automation properties

[Application](#)<sup>482</sup>

[Parent](#)<sup>485</sup>

[DTDSchemaFormat](#)<sup>482</sup>

[ValueList](#)<sup>485</sup>

[TypeDetection](#)<sup>485</sup>

[FrequentElements](#)<sup>483</sup>

[MergeAllEqualNamed](#)<sup>483</sup>

[ResolveEntities](#)<sup>485</sup>

[AttributeTypeDefinition](#)<sup>482</sup>

[GlobalAttributes](#)<sup>483</sup>

[OnlyStringEnums](#)<sup>484</sup>

[MaxEnumLength](#)<sup>483</sup>

[OutputPath](#)<sup>484</sup>

[OutputPathDialogAction](#)<sup>484</sup>

**Description**

Use this object to configure the generation of a schema or DTD. The method [GenerateDTDOrSchemaEx](#)<sup>461</sup> expects a DTDSchemaGeneratorDlg as parameter to configure the generation as well as the associated user interactions.

**14.3.2.12.1 Application**

**Property:** Application as [Application](#)<sup>356</sup> (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.

**14.3.2.12.2 AttributeTypeDefinition**

**Property:** AttributeTypeDefinition as [SPYAttributeTypeDefinition](#)<sup>589</sup>

**Description**

Specifies how attribute definitions get merged.

**Errors**

3000	The object is no longer valid.
3001	Invalid address for the return parameter was specified.

**14.3.2.12.3 DTDSchemaFormat**

**Property:** DTDSchemaFormat as [SPYDTDSchemaFormat](#)<sup>592</sup>

**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.

### 14.3.2.12.4 FrequentElements

**Property:** FrequentElements as [SPYFrequentElements](#)<sup>592</sup>

**Description**

Shall the types for all elements be defined as global? Use the 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.

### 14.3.2.12.5 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.

### 14.3.2.12.6 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.

### 14.3.2.12.7 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.

### 14.3.2.12.8 OnlyStringEnums

**Property:** OnlyStringEnums 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.

### 14.3.2.12.9 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.

### 14.3.2.12.10 OutputPathDialogAction

**Property:** OutputPathDialogAction as [SPYDialogAction](#)<sup>591</sup>

#### 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](#)<sup>484</sup> 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.

### 14.3.2.12.11 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.12.12 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.

### 14.3.2.12.13 TypeDetection

**Property:** TypeDetection as [SPYTypeDetection](#)<sup>599</sup>

**Description**

Specifies granularity of simple type detection.

**Errors**

3000	The object is no longer valid.
3001	Invalid address for the return parameter was specified.

### 14.3.2.12.14 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.
------	---

### 14.3.2.13 ElementList

#### Properties

[Count](#)<sup>486</sup>

[Item](#)<sup>486</sup>

#### Methods

[RemoveElement](#)<sup>486</sup>

#### Description

Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of [ElementListItem](#)<sup>487</sup> are used.

It can hold

- a list of table names returned by a call to [Application.GetDatabaseTables](#)<sup>364</sup>,
- a list of field names returned by a call to [Application.GetDatabaseImportElementList](#)<sup>363</sup> or [Application.GetTextImportElementList](#)<sup>365</sup>,
- a field name filter list used in [Application.ImportFromDatabase](#)<sup>367</sup> and [Application.ImportFromText](#)<sup>368</sup>,
- a list of table names and counts for their rows and columns as returned by calls to [GetExportElementList](#)<sup>464</sup> or
- a field name filter list used in [Document.ExportToDatabase](#)<sup>458</sup> and [Document.ExportToText](#)<sup>459</sup>.

#### 14.3.2.13.1 Count

**Property:** Count as long (read-only)

#### Description

Count of elements in this collection.

#### 14.3.2.13.2 Item

**Method:** Item(n as long) as [ElementListItem](#)<sup>487</sup>

#### Description

Gets the element with the index n from this collection. The first item has index 1.

#### 14.3.2.13.3 RemoveElement

**Method:** RemoveElement(Index as long)

#### Description

RemoveElement removes the element Index from the collection. The first Item has index 1.

## 14.3.2.14 ElementListItem

### Properties

[Name](#) <sup>487</sup>

[ElementKind](#) <sup>487</sup>

[FieldCount](#) <sup>487</sup>

[RecordCount](#) <sup>487</sup>

### Description

An element in an [ElementList](#) <sup>486</sup>. Usage of its properties depends on the purpose of the element list. For details see [ElementList](#) <sup>486</sup>.

### 14.3.2.14.1 ElementKind

**Property:** ElementKind as [SPYXMLDataKind](#) <sup>601</sup>

### Description

Specifies if a field should be imported as XML element (data value of spyXMLDataElement) or attribute (data value of spyXMLDataAttr).

### 14.3.2.14.2 FieldCount

**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](#) <sup>464</sup>.

### 14.3.2.14.3 Name

**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.

### 14.3.2.14.4 RecordCount

**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](#) <sup>464</sup>.

## 14.3.2.15 ExportSettings

### Properties

[ElementList](#) <sup>488</sup>

[EntitiesToText](#) <sup>488</sup>

[ExportAllElements](#) <sup>489</sup>

[SubLevelLimit](#) <sup>490</sup>

[FromAttributes](#) <sup>489</sup>

[FromSingleSubElements](#) <sup>489</sup>

[FromTextValues](#) <sup>489</sup>

[CreateKeys](#) <sup>488</sup>

[IndependentPrimaryKey](#) <sup>489</sup>

[Namespace](#) <sup>490</sup>

[ExportCompleteXML](#) <sup>489</sup>

[StartFromElement](#) <sup>490</sup>

### Description

ExportSettings contains options used during export of XML data to a database or text file.

#### 14.3.2.15.1 CreateKeys

**Property:** CreateKeys as Boolean

### Description

This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

#### 14.3.2.15.2 ElementList

**Property:** ElementList as [ElementList](#) <sup>486</sup>

### 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](#) <sup>464</sup>. It is possible to prevent exporting columns by removing elements from this list with [ElementList.RemoveElement](#) <sup>486</sup> before passing it to [Document.ExportToDatabase](#) <sup>458</sup> or [Document.ExportToText](#) <sup>459</sup>.

#### 14.3.2.15.3 EntitiesToText

**Property:** EntitiesToText as Boolean



**Description**

Defines if XML entities should be converted to text or left as they are during export. Default is True.

#### 14.3.2.15.4 ExportAllElements

**Property:** ExportAllElements as Boolean

**Description**

If set to true, all elements in the document will be exported. If set to false, then [ExportSettings.SubLevelLimit](#)<sup>490</sup> is used to restrict the number of sub levels to export. Default is true.

#### 14.3.2.15.5 ExportCompleteXML

**Property:** ExportCompleteXML as Boolean

**Description**

Defines whether the complete XML is exported or only the element specified by [StartFromElement](#)<sup>490</sup> and its children. Default is True.

#### 14.3.2.15.6 FromAttributes

**Property:** FromAttributes as Boolean

**Description**

Set FromAttributes to false if no export data should be created from attributes. Default is True.

#### 14.3.2.15.7 FromSingleSubElements

**Property:** FromSingleSubElements as Boolean

**Description**

Set FromSingleSubElements to false if no export data should be created from elements. Default is True.

#### 14.3.2.15.8 FromTextValues

**Property:** FromTextValues as Boolean

**Description**

Set FromTextValues to false if no export data should be created from text values. Default is True.

#### 14.3.2.15.9 IndependentPrimaryKey

**Property:** IndependentPrimaryKey as Boolean

**Description**

Turns creation of independent primary key counter for every element on or off. If [ExportSettings.CreateKeys](#)<sup>488</sup> is False, this property will be ignored. Default is True.

#### 14.3.2.15.10 Namespace

**Property:** Namespace as [SPYExportNamespace](#)<sup>592</sup>

**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.

#### 14.3.2.15.11 StartFromElement

**Property:** StartFromElement as String

**Description**

Specifies the start element for the export. This property is only considered when [ExportCompleteXML](#)<sup>489</sup> is false.

#### 14.3.2.15.12 SubLevelLimit

**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](#)<sup>489</sup> is true.

### 14.3.2.16 FileSelectionDlg

**Properties and Methods**

Standard automation properties

[Application](#)<sup>491</sup>

[Parent](#)<sup>492</sup>

Dialog properties

[FullName](#)<sup>491</sup>

Acceptance or cancellation of action that caused event

[DialogAction](#)<sup>491</sup>

**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](#)<sup>491</sup> to select or modify the file path and set the [FileSelectionDlg.DialogAction](#)<sup>491</sup> property to cancel or agree with the action that caused the event.

### 14.3.2.16.1 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.16.2 DialogAction

**Property:** DialogAction as [SPYDialogAction](#)<sup>591</sup>

**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.

### 14.3.2.16.3 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.

### 14.3.2.16.4 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

## 14.3.2.17 FindInFilesDlg

### Properties and Methods

Standard automation properties

[Application](#)<sup>493</sup>

[Parent](#)<sup>495</sup>

[Find](#)<sup>494</sup>

[RegularExpression](#)<sup>495</sup>

[Replace](#)<sup>495</sup>

[DoReplace](#)<sup>493</sup>

[ReplaceOnDisk](#)<sup>495</sup>

[MatchWholeWord](#)<sup>494</sup>

[MatchCase](#)<sup>494</sup>

[SearchLocation](#)<sup>496</sup>

[StartFolder](#)<sup>497</sup>

[IncludeSubfolders](#)<sup>494</sup>

[SearchInProjectFilesDoExternal](#)<sup>496</sup>

[FileExtension](#)<sup>493</sup>

[AdvancedXMLSearch](#)<sup>493</sup>

[XMLElementNames](#)<sup>498</sup>

[XMLElementContents](#)<sup>498</sup>

[XMLAttributeNames](#)<sup>497</sup>

[XMLAttributeContents](#)<sup>497</sup>

[XMLComments](#)<sup>498</sup>

[XMLCDATA](#)<sup>497</sup>

[XMLPI](#)<sup>498</sup>

[XMLRest](#)<sup>499</sup>

[ShowResult](#)<sup>496</sup>

#### Description

Use this object to configure the search (or replacement) for strings in files. The method [FindInFiles](#)<sup>362</sup> expects a FindInFilesDlg as parameter.

### 14.3.2.17.1 AdvancedXMLSearch

**Property:** AdvancedXMLSearch as Boolean

**Description**

Specifies if the XML search properties ([XMLElementNames](#)<sup>498</sup>, [XMLElementContents](#)<sup>498</sup>, [XMLAttributeNames](#)<sup>497</sup>, [XMLAttributeContents](#)<sup>498</sup>, [XMLComments](#)<sup>498</sup>, [XMLCDATA](#)<sup>497</sup>, [XMLPI](#)<sup>498</sup> and [XMLRest](#)<sup>499</sup>) are considered. The default is false.

**Errors**

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.2 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.17.3 DoReplace

**Property:** DoReplace as Boolean

**Description**

Specifies if the matched string is replaced by the string defined in [Replace](#)<sup>495</sup>. The default is false.

**Errors**

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.4 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.

**14.3.2.17.5 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.

**14.3.2.17.6 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.

**14.3.2.17.7 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.

**14.3.2.17.8 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.

### 14.3.2.17.9 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.17.10 RegularExpression

**Property:** RegularExpression as Boolean

**Description**

Specifies if [Find](#)<sup>494</sup> 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.

### 14.3.2.17.11 Replace

**Property:** Replace as String

**Description**

Specifies the replacement string. The matched string is only replaced if [DoReplace](#)<sup>493</sup> is set true.

**Errors**

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.12 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.

### 14.3.2.17.13 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.

### 14.3.2.17.14 SearchLocation

**Property:** SearchLocation as [SPYFindInFilesSearchLocation](#)<sup>592</sup>

**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.

### 14.3.2.17.15 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.



### 14.3.2.17.16 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.

### 14.3.2.17.17 XMLAttributeContents

**Property:** XMLAttributeContents as Boolean

**Description**

Specifies if attribute contents are searched when [AdvancedXMLSearch](#)<sup>493</sup> is true. The default is true.

**Errors**

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.18 XMLAttributeNames

**Property:** XMLAttributeNames as Boolean

**Description**

Specifies if attribute names are searched when [AdvancedXMLSearch](#)<sup>493</sup> is true. The default is true.

**Errors**

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.19 XMLCDATA

**Property:** XMLCDATA as Boolean

**Description**

Specifies if CDATA tags are searched when [AdvancedXMLSearch](#)<sup>493</sup> is true. The default is true.

**Errors**

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.20 XMLComments

**Property:** XMLComments as Boolean

#### Description

Specifies if comments are searched when [AdvancedXMLSearch](#)<sup>493</sup> is true. The default is true.

#### Errors

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.21 XMLElementContents

**Property:** XMLElementContents as Boolean

#### Description

Specifies if element contents are searched when [AdvancedXMLSearch](#)<sup>493</sup> is true. The default is true.

#### Errors

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.22 XMLElementNames

**Property:** XMLElementNames as Boolean

#### Description

Specifies if element names are searched when [AdvancedXMLSearch](#)<sup>493</sup> is true. The default is true.

#### Errors

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.17.23 XMLPI

**Property:** XMLPI as Boolean

#### Description

Specifies if XML processing instructions are searched when [AdvancedXMLSearch](#)<sup>493</sup> is true. The default is true.

#### Errors

3500	The object is no longer valid.
------	--------------------------------

3501	Invalid address for the return parameter was specified.
------	---

### 14.3.2.17.24 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](#)<sup>493</sup> is true. The default is true.

**Errors**

3500	The object is no longer valid.
3501	Invalid address for the return parameter was specified.

### 14.3.2.18 FindInFilesResult

**Properties and Methods**

Standard automation properties

[Application](#)<sup>499</sup>

[Parent](#)<sup>500</sup>

[Count](#)<sup>500</sup>

[Item](#)<sup>500</sup>

[Path](#)<sup>500</sup>

[Document](#)<sup>500</sup>

**Description**

This object represents a file that matched the search criteria. It contains a list of [FindInFilesResultMatch](#)<sup>501</sup> objects that describe the matching position.

#### 14.3.2.18.1 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.18.2 Count

**Property:** Count as long (read-only)

#### Description

Count of elements in this collection.

### 14.3.2.18.3 Document

**Property:** Path as [Document](#)<sup>445</sup> (read-only)

#### Description

This property returns the [Document](#)<sup>445</sup> 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.

### 14.3.2.18.4 Item

**Method:** Item(n as long) as [FindInFilesResultMatch](#)<sup>501</sup>

#### Description

Gets the element with the index n from this collection. The first item has index 1.

### 14.3.2.18.5 Parent

**Property:** Parent as [FindInFilesResults](#)<sup>503</sup> (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.

### 14.3.2.18.6 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.

### 14.3.2.19 FindInFilesResultMatch

**Properties and Methods**

Standard automation properties

[Application](#)<sup>501</sup>

[Parent](#)<sup>502</sup>

[Line](#)<sup>502</sup>

[Position](#)<sup>502</sup>

[Length](#)<sup>501</sup>

[LineText](#)<sup>502</sup>

[Replaced](#)<sup>503</sup>

**Description**

Contains the exact position in the file of the matched string.

#### 14.3.2.19.1 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

#### 14.3.2.19.2 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.

### 14.3.2.19.3 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.

### 14.3.2.19.4 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.

### 14.3.2.19.5 Parent

**Property:** Parent as [FindInFilesResult](#)<sup>499</sup> (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.

### 14.3.2.19.6 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.

### 14.3.2.19.7 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.

## 14.3.2.20 FindInFilesResults

**Properties and Methods**

Standard automation properties

[Application](#)<sup>503</sup>

[Parent](#)<sup>504</sup>

[Count](#)<sup>504</sup>

[Item](#)<sup>504</sup>

**Description**

This is the result of the [FindInFiles](#)<sup>362</sup> method. It is a list of [FindInFilesResult](#)<sup>499</sup> objects.

### 14.3.2.20.1 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.20.2 Count

**Property:** Count as long (read-only)

#### Description

Count of elements in this collection.

### 14.3.2.20.3 Item

**Method:** Item(n as long) as [FindInFilesResult](#)<sup>499</sup>

#### Description

Gets the element with the index n from this collection. The first item has index 1.

### 14.3.2.20.4 Parent

**Property:** Parent as [Application](#)<sup>356</sup> (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.

## 14.3.2.21 GenerateSampleXMLDlg

### Properties and Methods

Standard automation properties

[Application](#)<sup>505</sup>

[Parent](#)<sup>508</sup>

[NonMandatoryAttributes](#)<sup>507</sup>

[NonMandatoryElements](#)<sup>507</sup>

[RepeatCount](#)<sup>509</sup>

[FillAttributesWithSampleData](#)<sup>506</sup>

[FillElementsWithSampleData](#)<sup>506</sup>

[ContentOfNillableElementsIsNonMandatory](#)<sup>506</sup>

[TryToUseNonAbstractTypes](#)<sup>510</sup>

[SchemaOrDTDAssignment](#)<sup>509</sup>

[LocalNameOfRootElement](#)<sup>507</sup>

[NamespaceURIOfRootElement](#)<sup>507</sup>

[OptionsDialogAction](#)<sup>508</sup>

Properties that are no longer supported



[TakeFirstChoice - obsolete](#) <sup>509</sup>  
[FillWithSampleData - obsolete](#) <sup>506</sup>  
[Optimization - obsolete](#) <sup>508</sup>

**Description**

Used to set the parameters for the generation of sample XML instances based on a W3C schema or DTD.

14.3.2.21.1 Application

**Property:** Application as [Application](#) <sup>356</sup> (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.

14.3.2.21.2 ChoiceMode

**Property:** ChoiceMode as [SPYSampleXMLGenerationChoiceMode](#) <sup>596</sup>

**Description**

Specifies which elements will be generated.

**Errors**

2200	The object is no longer valid.
2201	Invalid address for the return parameter was specified.

14.3.2.21.3 ConsiderSampleValueHints

**Property:** ConsiderSampleValueHints as Boolean

**Description**

Selects whether to use [SampleValueHints](#) <sup>509</sup> or not.

**Errors**

2200	The object is no longer valid.
2201	Invalid address for the return parameter was specified.

#### 14.3.2.21.4 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.

#### 14.3.2.21.5 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.

#### 14.3.2.21.6 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.

#### 14.3.2.21.7 FillWithSampleData - obsolete

**Property:** FillWithSampleData as Boolean

**Description**

Do no longer access this property. Use [FillAttributesWithSampleData](#)<sup>506</sup> and [FillElementsWithSampleData](#)<sup>506</sup>, instead.

**Errors**

0001	The property is no longer accessible.
------	---------------------------------------

14.3.2.21.8 [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.

14.3.2.21.9 [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.

14.3.2.21.10 [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.

14.3.2.21.11 [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.

## 14.3.2.21.12 Optimization - obsolete

**Property:** Optimization as [SPYSampleXMLGenerationOptimization](#)<sup>596</sup>

**Description**

Do not use this property any longer. Use ChoiceMode and NonMandatoryElements.

**Errors**

0001	The property is no longer accessible.
------	---------------------------------------

## 14.3.2.21.13 OptionsDialogAction

**Property:** OptionsDialogAction as [SPYDialogAction](#)<sup>591</sup>

**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.

## 14.3.2.21.14 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.21.15 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.

### 14.3.2.21.16 SampleValueHints

**Property:** SampleValueHints as [SPYSampleXMLGenerationSampleValueHints](#)<sup>597</sup>

**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.

### 14.3.2.21.17 SchemaOrDTDAssignment

**Property:** SchemaOrDTDAssignment as [SPYSampleXMLGenerationSchemaOrDTDAssignment](#)<sup>597</sup>

**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.

### 14.3.2.21.18 TakeFirstChoice - obsolete

**Property:** TakeFirstChoice as Boolean

**Description**

Do no longer use this property.

**Errors**

0001	The property is no longer accessible.
------	---------------------------------------

14.3.2.21.19 [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.

14.3.2.22 [GridView](#)**Methods**

[Deselect](#) <sup>513</sup>

[Select](#) <sup>513</sup>

[SetFocus](#) <sup>513</sup>

**Properties**

[CurrentFocus](#) <sup>513</sup>

[IsVisible](#) <sup>513</sup>

**Description**

GridView Class

14.3.2.22.1 [Events](#)14.3.2.22.1.1 [OnBeforeDrag](#)

**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.

**14.3.2.22.1.2 OnBeforeDrop**

**Event:** OnBeforeDrop(objXMLData as [XMLData](#)<sup>576</sup>) 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.

**14.3.2.22.1.3 OnBeforeStartEditing**

**Event:** OnBeforeStartEditing(objXMLData as [XMLData](#)<sup>576</sup>, 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.

#### 14.3.2.22.1.4 OnEditingFinished

**Event:** OnEditingFinished(objXMLData as [XMLData](#)<sup>576</sup>, 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, ...) // nEventId = 2

#### Description

This event gets fired when the editing mode of a grid cell is exited. The parameter *bEditingName* specifies if the name part of the element has been edited.

#### 14.3.2.22.1.5 OnFocusChanged

**Event:** OnFocusChanged(objXMLData as [XMLData](#)<sup>576</sup>, 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, ...) // nEventId = 3

#### Description

This event gets fired whenever a grid cell receives or loses 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.



#### 14.3.2.22.2 CurrentFocus

**Property:** CurrentFocus as [XMLData](#)<sup>576</sup>

##### Description

Holds the XML element with the current focus. This property is read-only.

#### 14.3.2.22.3 Deselect

**Method:** Deselect(*pData* as [XMLData](#)<sup>576</sup>)

##### Description

Deselects the element *pData* in the grid view.

#### 14.3.2.22.4 IsVisible

**Property:** IsVisible as Boolean

##### Description

True if the grid view is the active view of the document. This property is read-only.

#### 14.3.2.22.5 Select

**Method:** Select (*pData* as [XMLData](#)<sup>576</sup>)

##### Description

Selects the XML element *pData* in the grid view.

#### 14.3.2.22.6 SetFocus

**Method:** SetFocus (*pFocusData* as [XMLData](#)<sup>576</sup>)

##### Description

Sets the focus to the element *pFocusData* in the grid view.

### 14.3.2.23 SchemaDocumentationDlg

#### Properties and Methods

Standard automation properties

[Application](#)<sup>515</sup>

[Parent](#)<sup>522</sup>

Interaction and visibility properties

[OutputFile](#) <sup>521</sup>  
[OutputFileDialogAction](#) <sup>521</sup>  
[OptionsDialogAction](#) <sup>520</sup>  
[ShowProgressBar](#) <sup>525</sup>  
[ShowResult](#) <sup>525</sup>

Document generation options and methods

[OutputFormat](#) <sup>521</sup>  
[UseFixedDesign](#) <sup>527</sup>  
[SPSFile](#) <sup>527</sup>  
[EmbedDiagrams](#) <sup>516</sup>  
[DiagramFormat](#) <sup>515</sup>  
[MultipleOutputFiles](#) <sup>520</sup>  
[EmbedCSSInHTML](#) <sup>516</sup>  
[CreateDiagramsFolder](#) <sup>515</sup>  
[GenerateRelativeLinks](#) <sup>516</sup>

[IncludeAll](#) <sup>517</sup>  
[IncludeIndex](#) <sup>518</sup>  
[IncludeGlobalAttributes](#) <sup>517</sup>  
[IncludeGlobalElements](#) <sup>518</sup>  
[IncludeLocalAttributes](#) <sup>519</sup>  
[IncludeLocalElements](#) <sup>519</sup>  
[IncludeGroups](#) <sup>518</sup>  
[IncludeComplexTypes](#) <sup>517</sup>  
[IncludeSimpleTypes](#) <sup>520</sup>  
[IncludeAttributeGroups](#) <sup>517</sup>  
[IncludeRedefines](#) <sup>519</sup>  
[IncludeReferencedSchemas](#) <sup>520</sup>

[AllDetails](#) <sup>515</sup>  
[ShowDiagram](#) <sup>523</sup>  
[ShowNamespace](#) <sup>524</sup>  
[ShowType](#) <sup>526</sup>  
[ShowChildren](#) <sup>523</sup>  
[ShowUsedBy](#) <sup>526</sup>  
[ShowProperties](#) <sup>525</sup>  
[ShowSingleFacets](#) <sup>525</sup>  
[ShowPatterns](#) <sup>524</sup>  
[ShowEnumerations](#) <sup>523</sup>  
[ShowAttributes](#) <sup>522</sup>  
[ShowIdentityConstraints](#) <sup>524</sup>  
[ShowAnnotations](#) <sup>522</sup>  
[ShowSourceCode](#) <sup>526</sup>

### Description

This object combines options for JSON 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 JSON Schema documentation. However, before using the object you have to set the [SetOutputFile](#) <sup>521</sup> property to a valid file path. Use [OptionsDialogAction](#) <sup>520</sup>, [OutputFileDialogAction](#) <sup>521</sup> and [ShowProgressBar](#) <sup>525</sup> to specify the level of user interaction desired. You can use [IncludeAll](#) <sup>517</sup> and [AllDetails](#) <sup>515</sup> to set whole option groups at once or the individual properties to operate on a finer granularity.

### 14.3.2.23.1 AllDetails

**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.
------	--------------------------------

### 14.3.2.23.2 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.23.3 CreateDiagramsFolder

**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.

**Errors**

2900	The object is no longer valid.
2901	Invalid address for the return parameter was specified.

### 14.3.2.23.4 DiagramFormat

**Property:** DiagramFormat as [SPYImageKind](#)<sup>593</sup>

**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](#)<sup>462</sup>. 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.

**14.3.2.23.5 EmbedCSSInHTML**

**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.

**14.3.2.23.6 EmbedDiagrams**

**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](#)<sup>462</sup>. 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.

**14.3.2.23.7 GenerateRelativeLinks**

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.8 IncludeAll

**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.
------	--------------------------------

### 14.3.2.23.9 IncludeAttributeGroups

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.10 IncludeComplexTypes

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.11 IncludeGlobalAttributes

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.12 IncludeGlobalElements

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.13 IncludeGroups

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.14 IncludeIndex

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.15 IncludeLocalAttributes

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.16 IncludeLocalElements

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.17 IncludeRedefines

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.18 IncludeReferencedSchemas

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.19 IncludeSimpleTypes

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.20 MultipleOutputFiles

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.21 OptionsDialogAction

**Property:** OptionsDialogAction as [SPYDialogAction](#)<sup>591</sup>



**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.

14.3.2.23.22 [OutputFile](#)

**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](#)<sup>462</sup>.

**Errors**

2900	The object is no longer valid.
2901	Invalid address for the return parameter was specified.

14.3.2.23.23 [OutputFileDialogAction](#)

**Property:** OutputFileDialogAction as [SPYDialogAction](#)<sup>591</sup>

**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](#)<sup>521</sup> 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.

14.3.2.23.24 [OutputFormat](#)

**Property:** OutputFormat as [SPYSchemaDocumentationFormat](#)<sup>598</sup>

**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](#)<sup>462</sup>. 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.

## 14.3.2.23.25 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

## 14.3.2.23.26 ShowAnnotations

**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](#)<sup>462</sup>. 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.

## 14.3.2.23.27 ShowAttributes

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.28 ShowChildren

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.29 ShowDiagram

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.30 ShowEnumerations

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.31 ShowIdentityConstraints

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.32 ShowNamespace

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.33 ShowPatterns

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.34 ShowProgressBar

**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.

### 14.3.2.23.35 ShowProperties

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.36 ShowResult

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.37 ShowSingleFacets

**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](#)<sup>462</sup>. 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.

**14.3.2.23.38 ShowSourceCode**

**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](#)<sup>462</sup>. 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.

**14.3.2.23.39 ShowType**

**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](#)<sup>462</sup>. 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.

**14.3.2.23.40 ShowUsedBy**

**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](#)<sup>462</sup>. 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.

### 14.3.2.23.41 SPSFile

**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.

### 14.3.2.23.42 UseFixedDesign

**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.

## 14.3.2.24 SpyProject

**Methods**

[CloseProject](#) <sup>528</sup>

[SaveProject](#) <sup>528</sup>

[SaveProjectAs](#) <sup>528</sup>

**Properties**

[RootItems](#) <sup>528</sup>

[ProjectFile](#) <sup>528</sup>

**Description**

SpyProject Class

### 14.3.2.24.1 CloseProject

**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.

### 14.3.2.24.2 ProjectFile

**Declaration:** ProjectFile as String

#### Description

Path and filename of the project.

### 14.3.2.24.3 RootItems

**Declaration:** RootItems as [SpyProjectItems](#) <sup>531</sup>

#### Description

Root level of collection of project items.

### 14.3.2.24.4 SaveProject

**Declaration:** SaveProject

#### Description

SaveProject saves the current project.

### 14.3.2.24.5 SaveProjectAs

**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.

## 14.3.2.25 SpyProjectItem

### Methods

[Open](#) <sup>530</sup>

### Properties

[ChildItems](#) <sup>529</sup>

[ParentItem](#) <sup>530</sup>

[FileExtensions](#) <sup>529</sup>

[ItemType](#) <sup>529</sup>

[Name](#) <sup>530</sup>

[Path](#) <sup>530</sup>

[ValidateWith](#) <sup>530</sup>

[XMLForXSLTransformation](#) <sup>530</sup>

[XSLForXMLTransformation](#) <sup>531</sup>

[XSLTransformationFileExtension](#) <sup>531</sup>

[XSLTransformationFolder](#) <sup>531</sup>

### Description

SpyProjectItem Class

### 14.3.2.25.1 ChildItems

**Declaration:** ChildItems as [SpyProjectItems](#) <sup>531</sup>

### Description

If the item is a folder, ChildItems is the collection of the folder content.

### 14.3.2.25.2 FileExtensions

**Declaration:** FileExtensions as String

### Description

Used to set the file extensions if the project item is a folder.

### 14.3.2.25.3 ItemType

**Declaration:** ItemType as [SPYProjectItemTypes](#) <sup>595</sup>

### Description

This property is read-only.

#### 14.3.2.25.4 Name

**Declaration:** Name as String

**Description**

Name of the project item. This property is read-only.

#### 14.3.2.25.5 Open

**Declaration:** Open as [Document](#)<sup>445</sup>

**Return Value**

The project item opened as document.

**Description**

Opens the project item.

#### 14.3.2.25.6 ParentItem

**Declaration:** ParentItem as [SpyProjectItem](#)<sup>529</sup>

**Description**

Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

#### 14.3.2.25.7 Path

**Declaration:** Path as String

**Description**

Path of project item. This property is read-only.

#### 14.3.2.25.8 ValidateWith

**Declaration:** ValidateWith as String

**Description**

Used to set the schema/DTD for validation.

#### 14.3.2.25.9 XMLForXSLTransformation

**Declaration:** XMLForXSLTransformation as String

**Description**

Used to set the XML for XSL transformation.

### 14.3.2.25.10 XSLForXMLTransformation

**Declaration:** XSLForXMLTransformation as String

**Description**

Used to set the XSL for XML transformation.

### 14.3.2.25.11 XSLTransformationFileExtension

**Declaration:** XSLTransformationFileExtension as String

**Description**

Used to set the file extension for XSL transformation output files.

### 14.3.2.25.12 XSLTransformationFolder

**Declaration:** XSLTransformationFolder as String

**Description**

Used to set the destination folder for XSL transformation output files.

## 14.3.2.26 SpyProjectItems

**Methods**

[AddFile](#)<sup>531</sup>

[AddFolder](#)<sup>532</sup>

[AddURL](#)<sup>532</sup>

[RemoveItem](#)<sup>533</sup>

**Properties**

[Count](#)<sup>532</sup>

[Item](#)<sup>533</sup>

**Description**

SpyProjectItems Class

### 14.3.2.26.1 AddFile

**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.

#### 14.3.2.26.2 AddFolder

**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.

#### 14.3.2.26.3 AddURL

**Declaration:** AddURL (*strURL* as String, *nURLType* as [SPYURLTypes](#)<sup>599</sup>, *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.

#### 14.3.2.26.4 Count

**Declaration:** Count as long

##### Description

This property gets the count of project items in the collection. The property is read-only.

### 14.3.2.26.5 Item

**Declaration:** Item (*n* as long) as [SpyProjectItem](#)<sup>529</sup>

#### Description

Retrieves the *n*-th element of the collection of project items. The first item has index 1.

### 14.3.2.26.6 RemoveItem

**Declaration:** RemoveItem (*pltem* as [SpyProjectItem](#)<sup>529</sup>)

#### Description

RemoveItem deletes the item *pltem* from the collection of project items.

## 14.3.2.27 TextImportExportSettings

### Properties for import only

[ImportFile](#)<sup>535</sup>

### Properties for export only

[DestinationFolder](#)<sup>534</sup>

[FileExtension](#)<sup>534</sup>

[CommentIncluded](#)<sup>533</sup>

[RemoveDelimiter](#)<sup>535</sup>

[RemoveNewline](#)<sup>535</sup>

### Properties for import and export

[HeaderRow](#)<sup>535</sup>

[FieldDelimiter](#)<sup>534</sup>

[EnclosingCharacter](#)<sup>534</sup>

[Encoding](#)<sup>534</sup>

[EncodingByteOrder](#)<sup>534</sup>

#### Description

TextImportExportSettings contains options common to text import and export functions.

### 14.3.2.27.1 CommentIncluded

**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.

### 14.3.2.27.2 DestinationFolder

**Property:** DestinationFolder as String

#### Description

The property DestinationFolder sets the folder where the created files are saved during text export.

### 14.3.2.27.3 EnclosingCharacter

**Property:** EnclosingCharacter as [SPYTextEnclosing](#)<sup>599</sup>

#### Description

This property defines the character that encloses all field values for import and export. Default is [spyNoEnclosing](#)<sup>599</sup>.

### 14.3.2.27.4 Encoding

**Property:** Encoding as String

#### Description

The property Encoding sets the character encoding for the text files for importing and exporting.

### 14.3.2.27.5 EncodingByteOrder

**Property:** EncodingByteOrder as [SPYEncodingByteOrder](#)<sup>592</sup>

#### Description

The property EncodingByteOrder sets the byte order for Unicode characters. Default is [spyNONE](#)<sup>592</sup>.

### 14.3.2.27.6 FieldDelimiter

**Property:** FieldDelimiter as [SPYTextDelimiters](#)<sup>598</sup>

#### Description

The property FieldDelimiter defines the delimiter between the fields during import and export. Default is [spyTabulator](#)<sup>598</sup>.

### 14.3.2.27.7 FileExtension

**Property:** FileExtension as String

#### Description

This property sets the file extension for files created on text export.

### 14.3.2.27.8 HeaderRow

**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.

### 14.3.2.27.9 ImportFile

**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.

### 14.3.2.27.10 RemoveDelimiter

**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.

### 14.3.2.27.11 RemoveNewline

**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.

## 14.3.2.28 TextView

### Properties and Methods

[Application](#) <sup>537</sup>

[Parent](#) <sup>539</sup>

[LineFromPosition](#) <sup>538</sup>

[PositionFromLine](#) <sup>539</sup>

[LineLength](#) <sup>539</sup>

[SetText](#) <sup>541</sup>

[GetRangeText](#) <sup>537</sup>

[ReplaceText](#) <sup>540</sup>

[MoveCaret](#) <sup>539</sup>  
[GoToLineChar](#) <sup>538</sup>  
[SelectText](#) <sup>540</sup>  
[SelectionStart](#) <sup>540</sup>  
[SelectionEnd](#) <sup>540</sup>  
[Text](#) <sup>541</sup>  
[LineCount](#) <sup>538</sup>  
[Length](#) <sup>538</sup>

## Description

### 14.3.2.28.1 Events

#### 14.3.2.28.1.1 *OnBeforeShowSuggestions*

**Event:** OnBeforeShowSuggestions() as Boolean

#### Description

This event gets fired before a suggestion window is shown. The [Document](#) <sup>445</sup> property [Suggestions](#) <sup>473</sup> 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](#) <sup>473</sup> property. The best location for this is the [OnDocumentOpened](#) <sup>358</sup> 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()  
{  
}  
}
```

##### **XMLSpy IDE Plugin:**

```
IXMLSpyPlugIn.OnEvent (33, ...) // nEventId = 33
```

### 14.3.2.28.1.2 *OnChar*

**Event:** OnChar(nChar as Long, bExistSuggestion 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](#)<sup>445</sup> property [Suggestions](#)<sup>473</sup> 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](#)<sup>473</sup> property. The best location for this is the [OnDocumentOpened](#)<sup>358</sup> 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](#)<sup>445</sup> property [Suggestions](#)<sup>473</sup> to a string array with your recommendations and return true. This event is fired before the [OnBeforeShowSuggestions](#)<sup>536</sup> event. If you prevent to show the suggestion window by returning false then [OnBeforeShowSuggestions](#)<sup>536</sup> is not fired.

**Examples**

Given below are examples of how this event can be scripted.

**XMLSpy scripting environment - VBScript:**

```
Function On_Char(nChar, bExistSuggestions)
End Function
```

**XMLSpy scripting environment - JScript:**

```
function On_Char(nChar, bExistSuggestions)
{
}
```

**XMLSpy IDE Plugin:**

```
IXMLSpyPlugIn.OnEvent (35, ...) // nEventId = 35
```

14.3.2.28.2 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

14.3.2.28.3 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.

#### 14.3.2.28.4 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.

#### 14.3.2.28.5 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.

#### 14.3.2.28.6 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.

#### 14.3.2.28.7 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.

### 14.3.2.28.8 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.

### 14.3.2.28.9 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.

### 14.3.2.28.10 Parent

**Property:** Parent as [Document](#)<sup>445</sup> (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.

### 14.3.2.28.11 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.

### 14.3.2.28.12 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.

### 14.3.2.28.13 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.

### 14.3.2.28.14 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.

### 14.3.2.28.15 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.

### 14.3.2.28.16 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.

### 14.3.2.28.17 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.

## 14.3.2.29 WSDLDocumentationDlg

**Properties and Methods**

Standard automation properties

[Application](#)<sup>542</sup>

[Parent](#)<sup>548</sup>

Interaction and visibility properties

[GlobalElementsAndTypesOnly](#)<sup>544</sup>

[OptionsDialogAction](#)<sup>547</sup>

[OutputFile](#)<sup>547</sup>

[OutputFileDialogAction](#)<sup>548</sup>

[SeparateSchemaDocument](#)<sup>548</sup>

[ShowProgressBar](#)<sup>551</sup>

[ShowResult](#)<sup>551</sup>

Document generation options and methods

[OutputFormat](#)<sup>548</sup>

[UseFixedDesign](#)<sup>552</sup>

[SPSFile](#)<sup>553</sup>

[EmbedDiagrams](#)<sup>544</sup>

[DiagramFormat](#)<sup>543</sup>

[MultipleOutputFiles](#)<sup>547</sup>

[EmbedCSSInHTML](#)<sup>543</sup>

[CreateDiagramsFolder](#)<sup>543</sup>

[IncludeAll](#)<sup>544</sup>  
[IncludeBinding](#)<sup>545</sup>  
[IncludeImportedWSDLFiles](#)<sup>545</sup>  
[IncludeMessages](#)<sup>545</sup>  
[IncludeOverview](#)<sup>545</sup>  
[IncludePortType](#)<sup>546</sup>  
[IncludeService](#)<sup>546</sup>  
[IncludeTypes](#)<sup>546</sup>  
  
[AllDetails](#)<sup>542</sup>  
[ShowBindingDiagram](#)<sup>549</sup>  
[ShowExtensibility](#)<sup>549</sup>  
[ShowMessageParts](#)<sup>549</sup>  
[ShowPort](#)<sup>550</sup>  
[ShowPortTypeDiagram](#)<sup>550</sup>  
[ShowPortTypeOperations](#)<sup>550</sup>  
[ShowServiceDiagram](#)<sup>551</sup>  
[ShowSourceCode](#)<sup>551</sup>  
[ShowTypesDiagram](#)<sup>552</sup>  
[ShowUsedBy](#)<sup>552</sup>

### 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](#)<sup>547</sup> property to a valid file path. Use [OptionsDialogAction](#)<sup>547</sup>, [OutputFileDialogAction](#)<sup>547</sup> and [ShowProgressBar](#)<sup>551</sup> to specify the level of user interaction desired. You can use [IncludeAll](#)<sup>544</sup> and [AllDetails](#)<sup>542</sup> to set whole option groups at once or the individual properties to operate on a finer granularity.

#### 14.3.2.29.1 AllDetails

**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.
------	--------------------------------

#### 14.3.2.29.2 Application

**Property:** Application as [Application](#)<sup>356</sup> (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.

### 14.3.2.29.3 CreateDiagramsFolder

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.4 DiagramFormat

**Property:** DiagramFormat as [SPYImageKind](#)<sup>593</sup>

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.5 EmbedCSSInHTML

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.6 EmbedDiagrams

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.7 GlobalElementsAndTypesOnly

**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.

### 14.3.2.29.8 IncludeAll

**Method:** IncludeAll (i\_blnclude as Boolean)

#### Description

Use this method to mark or unmark all include options.

#### Errors

4300	The object is no longer valid.
------	--------------------------------



### 14.3.2.29.9 IncludeBinding

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.10 IncludeImportedWSDLFiles

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.11 IncludeMessages

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.12 IncludeOverview

**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](#)<sup>463</sup>. 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.

**14.3.2.29.13 IncludePortType**

**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](#)<sup>463</sup>. 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.

**14.3.2.29.14 IncludeService**

**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](#)<sup>463</sup>. 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.

**14.3.2.29.15 IncludeTypes**

**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](#)<sup>463</sup>. 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.
------	---

### 14.3.2.29.16 MultipleOutputFiles

**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](#)<sup>463</sup>. 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. Invalid address for the return parameter was specified.

### 14.3.2.29.17 OptionsDialogAction

**Property:** OptionsDialogAction as [SPYDialogAction](#)<sup>591</sup>

**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.

### 14.3.2.29.18 OutputFile

**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](#)<sup>463</sup>.

**Errors**

3900	The object is no longer valid.
3901	Invalid address for the return parameter was specified.

### 14.3.2.29.19 OutputFileDialogAction

**Property:** OutputFileDialogAction as [SPYDialogAction](#)<sup>591</sup>

#### 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](#)<sup>547</sup> 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.

### 14.3.2.29.20 OutputFormat

**Property:** OutputFormat as [SPYSchemaDocumentationFormat](#)<sup>598</sup>

#### 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](#)<sup>463</sup>. 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.

### 14.3.2.29.21 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.29.22 SeparateSchemaDocument

**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.

14.3.2.29.23 ShowBindingDiagram

**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](#)<sup>463</sup>. 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.

14.3.2.29.24 ShowExtensibility

**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](#)<sup>463</sup>. 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.

14.3.2.29.25 ShowMessageParts

**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](#)<sup>463</sup>. 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.

#### 14.3.2.29.26 ShowPort

**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](#)<sup>463</sup>. 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.

#### 14.3.2.29.27 ShowPortTypeDiagram

**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](#)<sup>463</sup>. 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.

#### 14.3.2.29.28 ShowPortTypeOperations

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.29 ShowProgressBar

**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.

### 14.3.2.29.30 ShowResult

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.31 ShowServiceDiagram

**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](#)<sup>463</sup>. 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.

### 14.3.2.29.32 ShowSourceCode

**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](#)<sup>463</sup>. 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.

**14.3.2.29.33 ShowTypesDiagram**

**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](#)<sup>463</sup>. 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.

**14.3.2.29.34 ShowUsedBy**

**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](#)<sup>463</sup>. 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.

**14.3.2.29.35 UseFixedDesign**

**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.

### 14.3.2.29.36 SPSFile

**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.

## 14.3.2.30 WSDL20DocumentationDlg

**Properties and Methods**

Standard automation properties

- [Application](#) <sup>554</sup>
- [Parent](#) <sup>560</sup>

Interaction and visibility properties

- [GlobalElementsAndTypesOnly](#) <sup>556</sup>
- [OptionsDialogAction](#) <sup>559</sup>
- [OutputFile](#) <sup>559</sup>
- [OutputFileDialogAction](#) <sup>559</sup>
- [SeparateSchemaDocument](#) <sup>560</sup>
- [ShowProgressBar](#) <sup>562</sup>
- [ShowResult](#) <sup>563</sup>

Document generation options and methods

- [OutputFormat](#) <sup>560</sup>
- [UseFixedDesign](#) <sup>564</sup>
- [SPSFile](#) <sup>564</sup>
- [EmbedDiagrams](#) <sup>556</sup>
- [DiagramFormat](#) <sup>555</sup>
- [MultipleOutputFiles](#) <sup>558</sup>
- [EmbedCSSInHTML](#) <sup>555</sup>
- [CreateDiagramsFolder](#) <sup>555</sup>
- [IncludeAll](#) <sup>556</sup>
- [IncludeBinding](#) <sup>556</sup>
- [IncludeImportedWSDLFiles](#) <sup>557</sup>
- [IncludeInterface](#) <sup>557</sup>
- [IncludeOverview](#) <sup>557</sup>

[IncludeService](#) <sup>558</sup>  
[IncludeTypes](#) <sup>558</sup>

[AllDetails](#) <sup>554</sup>  
[ShowBindingDiagram](#) <sup>560</sup>  
[ShowExtensibility](#) <sup>561</sup>  
[ShowEndpoint](#) <sup>561</sup>  
[ShowFault](#) <sup>561</sup>  
[ShowInterfaceDiagram](#) <sup>562</sup>  
[ShowOperation](#) <sup>562</sup>  
[ShowServiceDiagram](#) <sup>563</sup>  
[ShowSourceCode](#) <sup>563</sup>  
[ShowTypesDiagram](#) <sup>564</sup>  
[ShowUsedBy](#) <sup>564</sup>

### 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](#) <sup>559</sup> property to a valid file path. Use [OptionsDialogAction](#) <sup>559</sup>, [OutputFileDialogAction](#) <sup>559</sup> and [ShowProgressBar](#) <sup>562</sup> to specify the level of user interaction desired. You can use [IncludeAll](#) <sup>556</sup> and [AllDetails](#) <sup>554</sup> to set whole option groups at once or the individual properties to operate on a finer granularity.

#### 14.3.2.30.1 AllDetails

**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.
------	--------------------------------

#### 14.3.2.30.2 Application

**Property:** Application as [Application](#) <sup>356</sup> (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.

### 14.3.2.30.3 CreateDiagramsFolder

**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.GenerateWSDL20Documentation](#)<sup>463</sup>. 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.

### 14.3.2.30.4 DiagramFormat

**Property:** DiagramFormat as [SPYImageKind](#)<sup>593</sup>

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.5 EmbedCSSInHTML

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.6 EmbedDiagrams

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.7 GlobalElementsAndTypesOnly

**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.

### 14.3.2.30.8 IncludeAll

**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.
------	--------------------------------

### 14.3.2.30.9 IncludeBinding

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.10 IncludeImportedWSDLFiles

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.11 IncludeInterface

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.12 IncludeOverview

**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](#)<sup>463</sup>. 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.

#### 14.3.2.30.13 IncludeService

**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](#)<sup>463</sup>. 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.

#### 14.3.2.30.14 IncludeTypes

**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](#)<sup>463</sup>. 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.

#### 14.3.2.30.15 MultipleOutputFiles

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.16 OptionsDialogAction

**Property:** OptionsDialogAction as [SPYDialogAction](#)<sup>591</sup>

**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.

### 14.3.2.30.17 OutputFile

**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](#)<sup>463</sup>.

**Errors**

4300	The object is no longer valid.
4301	Invalid address for the return parameter was specified.

### 14.3.2.30.18 OutputFileDialogAction

**Property:** OutputFileDialogAction as [SPYDialogAction](#)<sup>591</sup>

**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](#)<sup>559</sup> 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.

### 14.3.2.30.19 OutputFormat

**Property:** OutputFormat as [SPYSchemaDocumentationFormat](#)<sup>598</sup>

#### 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](#)<sup>463</sup>. 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.

### 14.3.2.30.20 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.30.21 SeparateSchemaDocument

**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.

### 14.3.2.30.22 ShowBindingDiagram

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.23 ShowEndpoint

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.24 ShowExtensibility

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.25 ShowFault

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.26 ShowInterfaceDiagram

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.27 ShowOperation

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.28 ShowProgressBar

**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.

### 14.3.2.30.29 ShowResult

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.30 ShowServiceDiagram

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.31 ShowSourceCode

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.32 ShowTypesDiagram

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.33 ShowUsedBy

**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](#)<sup>463</sup>. 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.

### 14.3.2.30.34 SPSFile

**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.

### 14.3.2.30.35 UseFixedDesign

**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.

### 14.3.2.31 XBRLDocumentationDlg

**Properties and Methods**

Standard automation properties

- [Application](#) <sup>566</sup>
- [Parent](#) <sup>571</sup>

Interaction and visibility properties

- [OptionsDialogAction](#) <sup>570</sup>
- [OutputFile](#) <sup>570</sup>
- [OutputFileDialogAction](#) <sup>570</sup>
- [ShowProgressBar](#) <sup>574</sup>
- [ShowResult](#) <sup>575</sup>

Document generation options and methods

- [OutputFormat](#) <sup>571</sup>
- [UseFixedDesign](#) <sup>576</sup>
- [SPSFile](#) <sup>576</sup>
- [EmbedDiagrams](#) <sup>567</sup>
- [DiagramFormat](#) <sup>567</sup>
- [EmbedCSSInHTML](#) <sup>567</sup>
- [CreateDiagramsFolder](#) <sup>566</sup>

- [IncludeAll](#) <sup>568</sup>
- [IncludeOverview](#) <sup>569</sup>
- [IncludeNamespacePrefixes](#) <sup>569</sup>
- [IncludeGlobalElements](#) <sup>569</sup>
- [IncludeDefinitionLinkroles](#) <sup>568</sup>
- [IncludePresentationLinkroles](#) <sup>569</sup>
- [IncludeCalculationLinkroles](#) <sup>568</sup>

- [AllDetails](#) <sup>566</sup>
- [ShowDiagram](#) <sup>572</sup>
- [ShowSubstitutiongroup](#) <sup>575</sup>
- [ShowItemtype](#) <sup>573</sup>
- [ShowBalance](#) <sup>572</sup>
- [ShowPeriod](#) <sup>574</sup>
- [ShowAbstract](#) <sup>572</sup>
- [ShowNillable](#) <sup>574</sup>
- [ShowLabels](#) <sup>573</sup>

[ShowReferences](#) <sup>575</sup>[ShowLinkbaseReferences](#) <sup>573</sup>[ShortQualifiedNames](#) <sup>571</sup>[ShowImportedElements](#) <sup>572</sup>**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](#) <sup>570</sup> property to a valid file path. Use [OptionsDialogAction](#) <sup>570</sup>, [OutputFileDialogAction](#) <sup>570</sup> and [ShowProgressBar](#) <sup>574</sup> to specify the level of user interaction desired. You can use [IncludeAll](#) <sup>568</sup> and [AllDetails](#) <sup>566</sup> to set whole option groups at once or the individual properties to operate on a finer granularity.

### 14.3.2.31.1 AllDetails

**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.
------	--------------------------------

### 14.3.2.31.2 Application

**Property:** Application as [Application](#) <sup>356</sup> (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.

### 14.3.2.31.3 CreateDiagramsFolder

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.4 DiagramFormat

**Property:** DiagramFormat as [SPYImageKind](#)<sup>593</sup>

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.5 EmbedCSSInHTML

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.6 EmbedDiagrams

**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](#)<sup>463</sup>. 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.

**14.3.2.31.7 IncludeAll**

**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.
------	--------------------------------

**14.3.2.31.8 IncludeCalculationLinkroles**

**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](#)<sup>463</sup>. 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.

**14.3.2.31.9 IncludeDefinitionLinkroles**

**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](#)<sup>463</sup>. 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.



### 14.3.2.31.10 IncludeGlobalElements

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.11 IncludeNamespacePrefixes

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.12 IncludeOverview

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.13 IncludePresentationLinkroles

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.14 OptionsDialogAction

**Property:** OptionsDialogAction as [SPYDialogAction](#)<sup>591</sup>

**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.

### 14.3.2.31.15 OutputFile

**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](#)<sup>463</sup>.

**Errors**

4400	The object is no longer valid.
4401	Invalid address for the return parameter was specified.

### 14.3.2.31.16 OutputFileDialogAction

**Property:** OutputFileDialogAction as [SPYDialogAction](#)<sup>591</sup>

**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](#)<sup>570</sup> 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.

### 14.3.2.31.17 OutputFormat

**Property:** OutputFormat as [SPYSchemaDocumentationFormat](#)<sup>598</sup>

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.18 Parent

**Property:** Parent as [Dialogs](#)<sup>441</sup> (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.

### 14.3.2.31.19 ShortQualifiedName

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.20 ShowAbstract

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.21 ShowBalance

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.22 ShowDiagram

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.23 ShowImportedElements

**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](#)<sup>463</sup>. 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.

14.3.2.31.24 ShowItemtype

**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](#)<sup>463</sup>. 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.

14.3.2.31.25 ShowLabels

**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](#)<sup>463</sup>. 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.

14.3.2.31.26 ShowLinkbaseReferences

**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](#)<sup>463</sup>. 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.

**14.3.2.31.27 ShowNillable**

**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](#)<sup>463</sup>. 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.

**14.3.2.31.28 ShowPeriod**

**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](#)<sup>463</sup>. 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.

**14.3.2.31.29 ShowProgressBar**

**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.

### 14.3.2.31.30 ShowReferences

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.31 ShowResult

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.32 ShowSubstitutiongroup

**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](#)<sup>463</sup>. 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.

### 14.3.2.31.33 SPSFile

**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.

### 14.3.2.31.34 UseFixedDesign

**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.

## 14.3.2.32 XMLData

### Properties

[Kind](#) <sup>586</sup>

[Name](#) <sup>587</sup>

[TextValue](#) <sup>588</sup>

[HasChildren](#) <sup>584</sup>

[MayHaveChildren](#) <sup>587</sup>

[Parent](#) <sup>587</sup>

### Methods

[GetFirstChild](#) <sup>582</sup>

[GetNextChild](#) <sup>583</sup>

[GetCurrentChild](#) <sup>582</sup>

[InsertChild](#) <sup>585</sup>

[InsertChildAfter](#) <sup>585</sup>

[InsertChildBefore](#) <sup>586</sup>

[AppendChild](#) <sup>577</sup>

[EraseAllChildren](#) <sup>579</sup>



- [EraseChild](#) <sup>579</sup>
- [EraseCurrentChild](#) <sup>579</sup>
- [IsSameNode](#) <sup>586</sup>
- [CountChildren](#) <sup>578</sup>
- [CountChildrenKind](#) <sup>578</sup>
- [GetChild](#) <sup>580</sup>
- [GetChildAttribute](#) <sup>581</sup>
- [GetChildElement](#) <sup>581</sup>
- [GetChildKind](#) <sup>581</sup>
- [GetNamespacePrefixForURI](#) <sup>582</sup>
- [HasChildrenKind](#) <sup>585</sup>
- [SetTextValueXMLEncoded](#) <sup>588</sup>

**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](#) <sup>587</sup> 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.

14.3.2.32.1 AppendChild

**Declaration:** AppendChild (pNewData as [XMLData](#) <sup>576</sup>)

**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**

```
Dim objCurrentParent As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)
Set objCurrentParent = objSpy.ActiveDocument.RootElement

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

**14.3.2.32.2 CountChildren**

**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.
------	--

**14.3.2.32.3 CountChildrenKind**

**Declaration:** CountChildrenKind (*nKind* as [SPYXMLDataKind](#)<sup>601</sup>) 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.
------	--

### 14.3.2.32.4 EraseAllChildren

**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.

```
Dim objCurrentParent As XMLData

Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

### 14.3.2.32.5 EraseChild

**Method:** EraseChild (Child as [XMLData](#)<sup>576</sup>)

**Description**

Deletes the given child node.

**Errors**

1500	Invalid object.
1506	Invalid input xml
1510	Invalid parameter.

### 14.3.2.32.6 EraseCurrentChild

**Declaration:** EraseCurrentChild

**Description**

EraseCurrentChild deletes the current XMLData child object. Before you call EraseCurrentChild you must initialize an internal iterator with [XMLData.GetFirstChild](#)<sup>582</sup>. 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

```
// -----
// 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
var objList = Application.ActiveDocument.GridView.CurrentFocus;

// 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); }
}
```

### 14.3.2.32.7 GetChild

**Declaration:** GetChild (*position* as long) as [XMLData](#)<sup>576</sup>

#### 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.

### 14.3.2.32.8 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.

### 14.3.2.32.9 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.

### 14.3.2.32.10 GetChildKind

**Declaration:** GetChildKind (position as long, nKind as [SPYXMLDataKind](#)<sup>601</sup>) as [XMLData](#)<sup>576</sup>

**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.

### 14.3.2.32.11 GetCurrentChild

**Declaration:** GetCurrentChild as [XMLData](#)<sup>576</sup>

#### 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](#)<sup>582</sup>.

#### 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.

### 14.3.2.32.12 GetFirstChild

**Declaration:** GetFirstChild (nKind as [SPYXMLDataKind](#)<sup>601</sup>) as [XMLData](#)<sup>576</sup>

#### 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](#)<sup>582</sup>, [GetNextChild](#)<sup>583</sup> or [EraseCurrentChild](#)<sup>579</sup>.

#### 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](#)<sup>583</sup>.

### 14.3.2.32.13 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.

14.3.2.32.14 `GetNextChild`

**Declaration:** `GetNextChild` as [XMLData](#)<sup>576</sup>

**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](#)<sup>582</sup>.

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 usefull 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);
}

```

#### 14.3.2.32.15 GetTextValueXMLDecoded

**Method:** GetTextValueXMLDecoded ()as string

##### Description

Gets the decoded text value of the XML.

##### Errors

1500	Invalid object.
1510	Invalid parameter.

#### 14.3.2.32.16 HasChildren

**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.



### 14.3.2.32.17 HasChildrenKind

**Declaration:** HasChildrenKind (*nKind* as [SPYXMLDataKind](#)<sup>601</sup>) 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.

### 14.3.2.32.18 InsertChild

**Declaration:** InsertChild (*pNewData* as [XMLData](#)<sup>576</sup>)

**Description**

InsertChild inserts the new child before the current child (see also [XMLData.GetFirstChild](#)<sup>582</sup>, [XMLData.GetNextChild](#)<sup>583</sup> 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

### 14.3.2.32.19 InsertChildAfter

**Method:** InsertChildAfter (Node 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

#### 14.3.2.32.20 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

#### 14.3.2.32.21 IsSameNode

**Declaration:** IsSameNode (pNodeToCompare as [XMLData](#)<sup>576</sup>) 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.

#### 14.3.2.32.22 Kind

**Declaration:** Kind as [SPYXMLDataKind](#)<sup>601</sup>

##### 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.

### 14.3.2.32.23 MayHaveChildren

**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.

### 14.3.2.32.24 Name

**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.

### 14.3.2.32.25 Parent

**Declaration:** Parent as [XMLData](#) <sup>576</sup>

**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.

### 14.3.2.32.26 SetTextValueXMLEncoded

**Method:** SetTextValueXMLEncoded ( *strVal* as [String](#)<sup>601</sup> )

#### Description

Sets the encoded text value of the XML.

#### Errors

1500	Invalid object.
1513	Modification not allowed.

### 14.3.2.32.27 TextValue

**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.

## 14.3.3 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.

### 14.3.3.1 ENUMApplicationStatus

Enumeration to specify the current Application status.

eApplicationRunning	= 0
eApplicationAfterLicenseCheck	= 1
eApplicationBeforeLicenseCheck	= 2
eApplicationConcurrentLicenseCheckFailed	= 3
eApplicationProcessingCommandLine	= 4

### 14.3.3.2 SPYAttributeTypeDefinition

Attribute type definition that can be selected for generation of Sample XML. This type is used with the method [GenerateDTDOrSchema](#)<sup>460</sup> and [GenerateDTDOrSchemaEx](#)<sup>461</sup>.

spyMergedGlobal	= 0
spyDistinctGlobal	= 1
spyLocal	= 2

### 14.3.3.3 SPYAuthenticActions

Actions that can be performed on [AuthenticRange](#)<sup>383</sup> objects.

spyAuthenticInsertAt	= 0
spyAuthenticApply	= 1
spyAuthenticClearSurr	= 2
spyAuthenticAppend	= 3
spyAuthenticInsertBefore	= 4
spyAuthenticRemove	= 5

### 14.3.3.4 SPYAuthenticDocumentPosition

Relative and absolute positions used for navigating with [AuthenticRange](#)<sup>383</sup> objects.

spyAuthenticDocumentBegin	= 0
spyAuthenticDocumentEnd	= 1
spyAuthenticRangeBegin	= 2
spyAuthenticRangeEnd	= 3

### 14.3.3.5 SPYAuthenticElementActions

Actions that can be used with the obsolete object `GetAllowedElements` (superseded by [AuthenticRange.CanPerformActionWith](#)<sup>386</sup>).

k_ActionInsertAt	= 0
k_ActionApply	= 1
k_ActionClearSurr	= 2

k_ActionAppend	= 3
k_ActionInsertBefore	= 4
k_ActionRemove	= 5

### 14.3.3.6 SPYAuthenticElementKind

Enumeration of the different kinds of elements used for navigation and selection within the [AuthenticRange](#)<sup>383</sup> and [AuthenticView](#)<sup>412</sup> objects.

spyAuthenticChar	= 0
spyAuthenticWord	= 1
spyAuthenticLine	= 3
spyAuthenticParagraph	= 4
spyAuthenticTag	= 6
spyAuthenticDocument	= 8
spyAuthenticTable	= 9
spyAuthenticTableRow	= 10
spyAuthenticTableColumn	= 11

### 14.3.3.7 SPYAuthenticMarkupVisibility

Enumeration values to customize the visibility of markup with [MarkupVisibility](#)<sup>426</sup>.

spyAuthenticMarkupHidden	= 0
spyAuthenticMarkupSmall	= 1
spyAuthenticMarkupLarge	= 2
spyAuthenticMarkupMixed	= 3

### 14.3.3.8 SPYAuthenticToolbarButtonState

Authentic toolbar button states are given by the following enumerations.

authenticToolbarButtonDefault	= 0
authenticToolbarButtonEnabled	= 1
authenticToolbarButtonDisabled	= 2

### 14.3.3.9 SPYDatabaseKind

Values to select different kinds of databases for import. See [DatabaseConnection.DatabaseKind](#)<sup>437</sup> for its use.

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

### 14.3.3.10 SPYDialogAction

Values to simulate different interactions on dialogs. See [Dialogs](#)<sup>441</sup> for all dialogs available.

spyDialogOK	= 0	// simulate click on OK button
spyDialogCancel	= 1	// simulate click on Cancel button
spyDialogUserInput	= 2	// show dialog and allow user interaction

### 14.3.3.11 SPYDOMType

Enumeration values to parameterize generation of C++ code from schema definitions.

spyDOMType_msxml4	= 0	Obsolete
spyDOMType_xerces	= 1	
spyDOMType_xerces3	= 2	
spyDOMType_msxml6	= 3	

spyDOMType\_xerces indicates Xerces 2.x usage.  
 spyDOMType\_xerces3 indicates Xerces 3.x usage.

### 14.3.3.12 SPYDTDSchemaFormat

Enumeration to identify the different schema formats.

spyDTD	= 0
spyW3C	= 1

### 14.3.3.13 SPYEncodingByteOrder

Enumeration values to specify encoding byte ordering for text import and export.

spyNONE	= 0
spyLITTLE_ENDIAN	= 1
spyBIG_ENDIAN	= 2

### 14.3.3.14 SPYExportNamespace

Enumeration type to configure handling of namespace identifiers during export.

spyNoNamespace	= 0
spyReplaceColonWithUnderscore	= 1

### 14.3.3.15 SPYFindInFilesSearchLocation

The different locations where a search can be performed. This type is used with the [FindInFilesDlg](#)<sup>492</sup> dialog.

spyFindInFiles_Documents	= 0
spyFindInFiles_Project	= 1
spyFindInFiles_Folder	= 2

### 14.3.3.16 SPYFrequentElements

Enumeration values to parameterize schema generation.

spyGlobalElements	= 0
spyGlobalComplexType	= 1



### 14.3.3.17 SPYImageKind

Enumeration values to parameterize image type of the generated documentation. These values are used in [SchemaDocumentationDialog.DiagramFormat](#)<sup>515</sup>.

spyImageType_PNG	= 0
spyImageType_EMF	= 1

### 14.3.3.18 SPYImportColumnsType

Enumeration to specify different Import columns types.

spyImportColumns_Element	= 0
spyImportColumns_Attribute	= 1

### 14.3.3.19 SPYKeyEvent

Enumeration type to identify the different key events. These events correspond with the equally named windows messages.

spyKeyDown	= 0
spyKeyUp	= 1
spyKeyPressed	= 2

### 14.3.3.20 SPYKeyStatus

Enumeration type to identify the key status.

spyLeftShiftKeyMask	= 1
spyRightShiftKeyMask	= 2
spyLeftCtrlKeyMask	= 4
spyRightCtrlKeyMask	= 8
spyLeftAltKeyMask	= 16
spyRightAltKeyMask	= 32

### 14.3.3.21 SPYLibType

Enumeration values to parameterize generation of C++ code from schema definitions.

spyLibType_static	= 0
spyLibType_dll	= 1

### 14.3.3.22 SPYLoading

Enumeration values to define loading behaviour of URL files.

spyUseCacheProxy	= 0
spyReload	= 1

### 14.3.3.23 SPYMouseEvent

Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather than directly comparing with them.

spyNoButtonMask	= 0
spyMouseMoveMask	= 1
spyLeftButtonMask	= 2
spyMiddleButtonMask	= 4
spyRightButtonMask	= 8
spyButtonUpMask	= 16
spyButtonDownMask	= 32
spyDoubleClickMask	= 64
spyShiftKeyDownMask	= 128
spyCtrlKeyDownMask	= 256
spyLeftButtonDownMask	= 34 // spyLeftButtonMask   spyButtonDownMask
spyMiddleButtonDownMask	= 36 // spyMiddleButtonMask   spyButtonDownMask
spyRightButtonDownMask	= 40 // spyRightButtonMask   spyButtonDownMask
spyLeftButtonUpMask	= 18 // spyLeftButtonMask   spyButtonUpMask
spyMiddleButtonUpMask	= 20 // spyMiddleButtonMask   spyButtonUpMask
spyRightButtonUpMask	= 24 // spyRightButtonMask   spyButtonUpMask
spyLeftDoubleClickMask	= 66 // spyRightButtonMask   spyButtonUpMask
spyMiddleDoubleClickMask	= 68 // spyMiddleButtonMask   spyDoubleClickMask

spyRightDoubleClickMask	= 72 // spyRightButtonMask   spyDoubleClickMask
-------------------------	---

**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
```

### 14.3.3.24 SPYNumberDateTimeFormat

Enumeration value to configure database connections.

spySystemLocale	= 0
spySchemaCompatible	= 1

### 14.3.3.25 SPYProgrammingLanguage

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.

spyUndefinedLanguage	= -1
spyJava	= 0
spyCpp	= 1
spyCSharp	= 2

### 14.3.3.26 SPYProjectItemTypes

Enumeration values to identify the different elements in project item lists. See [SpyProjectItem.ItemType](#)<sup>529</sup>.

spyUnknownItem	= 0
spyFileItem	= 1
spyFolderItem	= 2
spyURLItem	= 3

### 14.3.3.27 SPYProjectType

Enumeration values to generation C# and C++ code from schema definitions.

spyVisualStudio2010Project	= 6	
spyVisualStudio2013Project	= 7	
spyVisualStudio2015Project	= 8	
spyVisualStudio2017Project	= 9	
spyVisualStudio2019Project	=10	
spyDotNetCore3_1_Project	=11	C# only
spyDotNet5_0_Project	=12	C# only
spyDotNet6_0_Project	=13	C# only
spyVisualStudio2022Project	=14	
spyDotNet8_0_Project	=15	C# only

### 14.3.3.28 SpySampleXMLGenerationChoiceMode

This enumeration is used in [GenerateSampleXMLDlg.ChoiceMode](#)<sup>505</sup>:

spySampleXMLGen_FirstBranch	= 0
spySampleXMLGen_AllBranches	= 1
spySampleXMLGen_ShortestBranch	= 2

### 14.3.3.29 SPYSampleXMLGenerationOptimization (Obsolete)

**This enumeration is OBSOLETE since v2014.**

Specify the elements that will be generated in the Sample XML. This enumeration is used in [GenerateSampleXMLDlg](#)<sup>504</sup>.

spySampleXMLGen_Optimized	= 0
spySampleXMLGen_NonMandatoryElements	= 1
spySampleXMLGen_Everything	= 2

### 14.3.3.30 SpySampleXMLGenerationSampleValueHints

This enumeration is used in [GenerateSampleXMLDlg.SampleValueHints](#)<sup>509</sup>

spySampleXMLGen_FirstFit	= 0
spySampleXMLGen_RandomFit	= 1
spySampleXMLGen_CycleThrough	= 2

### 14.3.3.31 SPYSampleXMLGenerationSchemaOrDTDAssignment

Specifies what kind of reference to the schema/DTD should be added to the generated Sample XML. This enumeration is used in [GenerateSampleXMLDlg](#)<sup>504</sup>.

spySampleXMLGen_AssignRelatively	= 0
spySampleXMLGen_AssignAbsolutely	= 1
spySampleXMLGen_DoNotAssign	= 2

### 14.3.3.32 SPYSchemaDefKind

Enumeration type to select schema diagram types.

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

### 14.3.3.33 SPYSchemaDocumentationFormat

Enumeration values to parameterize generation of schema documentation. These values are used in [SchemaDocumentationDialog.OutputFormat](#)<sup>521</sup>.

spySchemaDoc_HTML	= 0
spySchemaDoc_MSWord	= 1
spySchemaDoc_RTF	= 2
spySchemaDoc_PDF	= 3

### 14.3.3.34 SPYSchemaExtensionType

Enumeration to specify different Schema Extension types.

spySchemaExtension_None	= 0
spySchemaExtension_SQL_XML	= 1
spySchemaExtension_MS_SQL_Server	= 2
spySchemaExtension_Oracle	= 3

### 14.3.3.35 SPYSchemaFormat

Enumeration to specify different Schema Format types.

spySchemaFormat_Hierarchical	= 0
spySchemaFormat_Flat	= 1

### 14.3.3.36 SPYTextDelimiters

Enumeration values to specify text delimiters for text export.

spyTabulator	= 0
spySemicolon	= 1
spyComma	= 2
spySpace	= 3

### 14.3.3.37 SPYTextEnclosing

Enumeration value to specify text enclosing characters for text import and export.

spyNoEnclosing	= 0
spySingleQuote	= 1
spyDoubleQuote	= 2

### 14.3.3.38 SPYTypeDetection

Enumeration to select how type detection works during [GenerateDTDOrSchema](#)<sup>460</sup> and [GenerateDTDOrSchemaEx](#)<sup>461</sup>.

spyBestPossible	= 0
spyNumbersOnly	= 1
spyNoDetection	= 2

### 14.3.3.39 SPYURLTypes

Enumeration to specify different URL types.

spyURLTypeAuto	= -1
spyURLTypeXML	= 0
spyURLTypeDTD	= 1

### 14.3.3.40 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_AutoDetect	= 0
spyValidateXSDVersion_1_1	= 1
spyValidateXSDVersion_1_0	= 2

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.

### 14.3.3.41 SPYValidateErrorFormat

Enumeration values that select the format of the error message.

spyValidateErrorFormat_Text	= 0
spyValidateErrorFormat_ShortXML	= 1
spyValidateErrorFormat_LongXML	= 2

### 14.3.3.42 SPYViewModes

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.

spyViewGrid	= 0
spyViewText	= 1
spyViewBrowser	= 2
spyViewSchema	= 3
spyViewContent	= 4 // obsolete
spyViewAuthentic	= 4
spyViewWSDL	= 5
spyViewZIP	= 6
spyViewEditionInfo	= 7
spyViewXBRL	= 8
spyViewJsonSchema	= 9



### 14.3.3.43 SPYVirtualKeyMask

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.

spyNoVirtualKeyMask	= 0
spyLeftShiftKeyMask	= 1
spyRightShiftKeyMask	= 2
spyLeftCtrlKeyMask	= 4
spyRightCtrlKeyMask	= 8
spyLeftAltKeyMask	= 16
spyRightAltKeyMask	= 32
spyShiftKeyMask	= 3 // spyLeftShiftKeyMask   spyRightShiftKeyMask
spyCtrlKeyMask	= 12 // spyLeftCtrlKeyMask   spyRightCtrlKeyMask
spyAltKeyMask	= 48 // spyLeftAltKeyMask   spyRightAltKeyMask

#### Examples

```
' VBScript sample: check if ctrl-key is pressed
If ((i_nVirtualKeyStatus And spyCtrlKeyMask) <> 0) Then
  ' ctrl-key is pressed
End If

' VBScript sample: check if ONLY ctrl-key is pressed
If (i_nVirtualKeyStatus == spyCtrlKeyMask) 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
}
```

### 14.3.3.44 SPYXMLDataKind

The different types of XMLData elements available for XML documents.

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

## 14.4 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#, and VB.NET. 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](#)<sup>603</sup>).

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](#)<sup>606</sup> and [Integration at Document Level](#)<sup>608</sup> describe the key steps at these respective levels. The [ActiveX Integration Examples](#)<sup>611</sup> section provides examples in C# and Java. Looking through these examples will help you to make the right decisions quickly. The [Object Reference](#)<sup>632</sup> 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](#)<sup>149</sup>.

### 14.4.1 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 2012/2013/2015/2017/2019/2022
- 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](#)<sup>604</sup>).

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](#)<sup>611</sup>.

### 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.

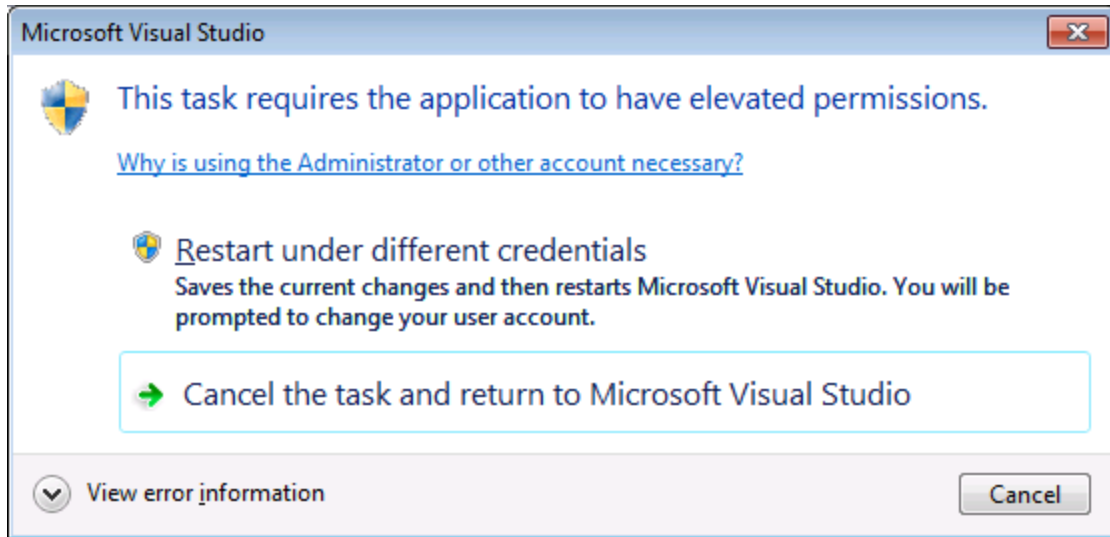
## 14.4.2 Adding the ActiveX Controls to the Toolbox

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:

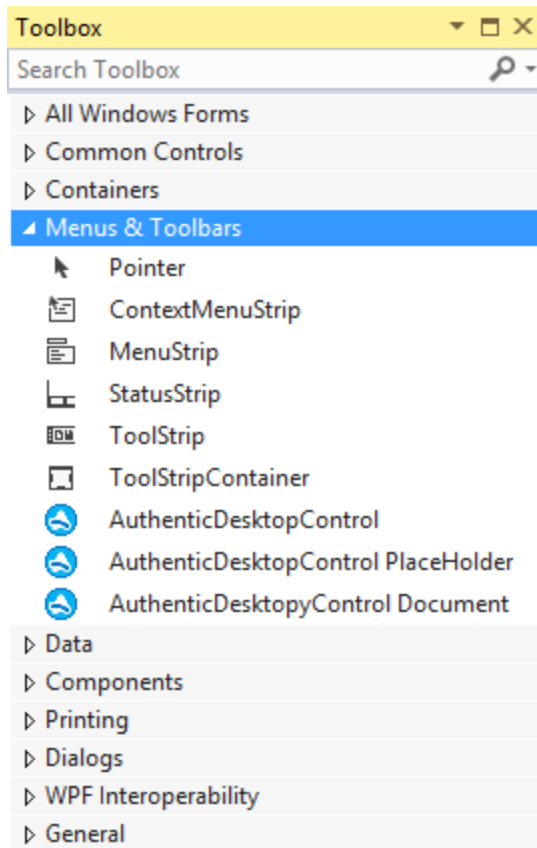
1. On the **Tools** menu of Visual Studio, click **Choose Toolbox Items**.
2. On the **COM Components** tab, select the check boxes next to the Authentic DesktopControl, Authentic DesktopControl Document, and Authentic DesktopControl Placeholder.

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](#) <sup>603</sup>.
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](#)<sup>606</sup>). The **AuthenticDesktopControl Document** and **AuthenticDesktopControl Placeholder** controls are used for document-level integration (see [Integration at Document Level](#)<sup>606</sup>).

### 14.4.3 Integration at Application Level

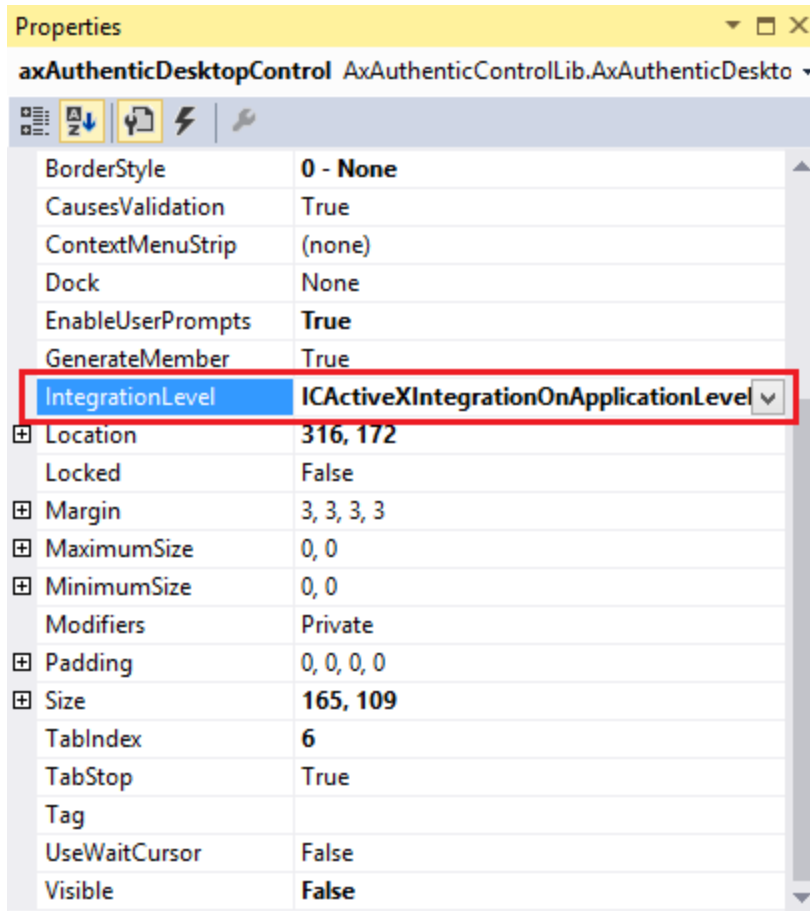
Integration at application level allows you to embed the complete interface of Authentic Desktop into a window of your application. With this type of integration, you get the whole user interface of Authentic Desktop, including all menus, toolbars, the status bar, document windows, and helper windows. 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](#)<sup>635</sup>. Do not instantiate or access [AuthenticDesktopControlDocument](#)<sup>643</sup> or [AuthenticDesktopControlPlaceholder](#)<sup>649</sup> ActiveX controls when integrating at application-level.

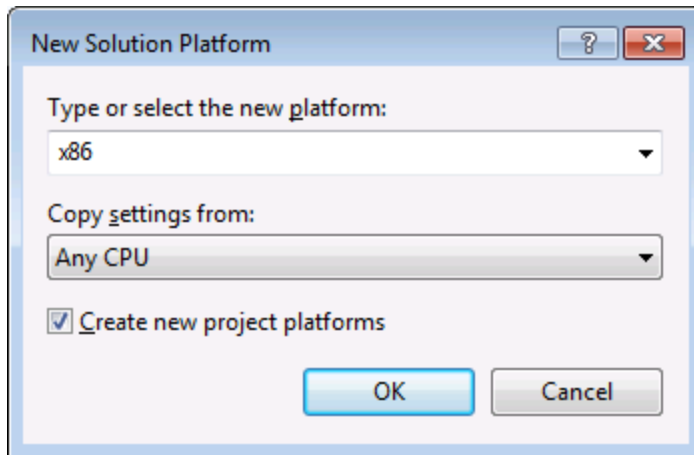
If you have any initialization to do or if you want to automate some behaviour of Authentic Desktop, use the properties, methods, and events described for [AuthenticDesktopControl](#)<sup>635</sup>. Consider using [AuthenticDesktopControl.Application](#)<sup>636</sup> for more complex access to Authentic Desktop functionality.

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](#) <sup>603</sup>).
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](#) <sup>604</sup>).
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).

#### 14.4.4 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](#)<sup>606</sup>, for an ActiveX integration at application level, only one control is required, namely the **AuthenticDesktopControl**. However, for an ActiveX integration at document level, Authentic Desktop functionality is provided by the following ActiveX controls:

- [AuthenticDesktopControl](#)<sup>635</sup>
- [AuthenticDesktopControl Document](#)<sup>643</sup>
- [AuthenticDesktopControl Placeholder](#)<sup>649</sup>

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](#)<sup>604</sup>).

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](#)<sup>637</sup> property to ICActiveXIntegrationOnDocumentLevel (or "1"). To hide the control from the user, set its Visible property to False. Note that, when integrating at document level, do not use the Open method of the



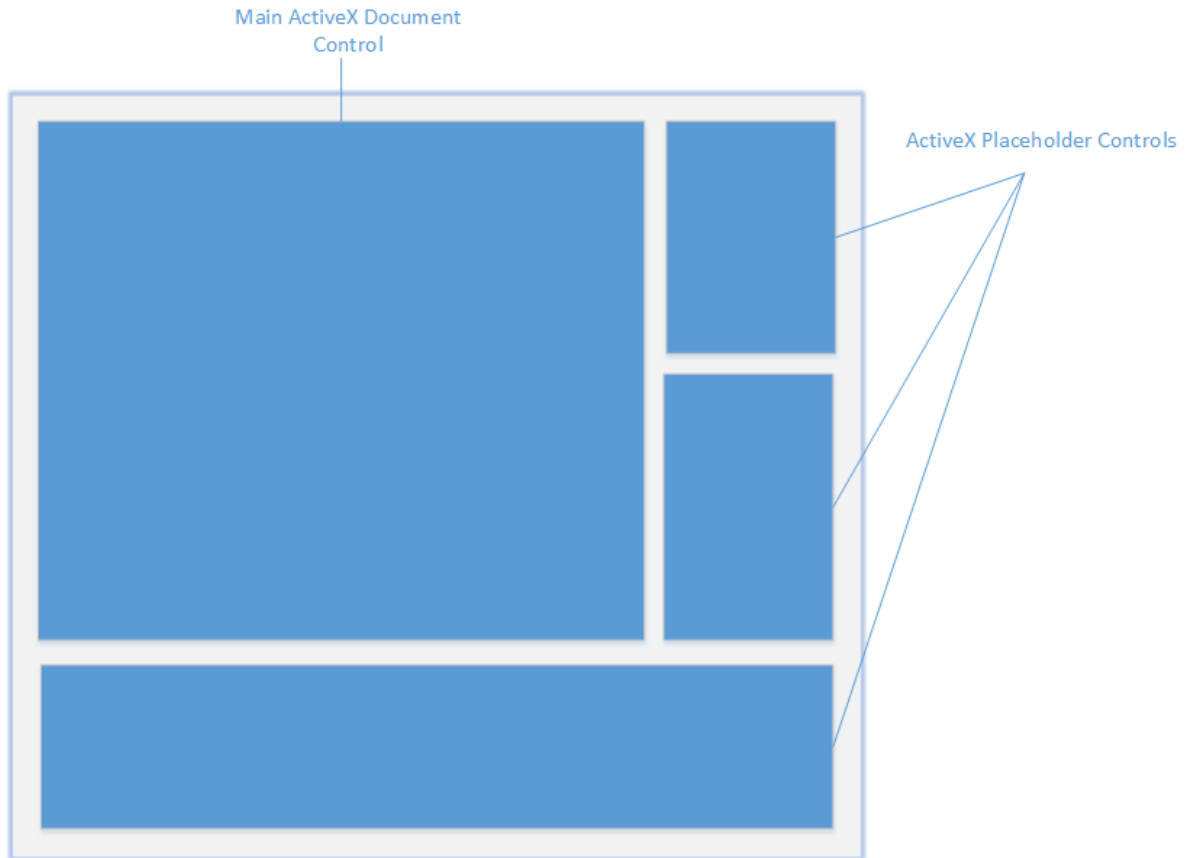
AuthenticDesktopControl; this might lead to unexpected results. Use the corresponding open methods of AuthenticDesktopControl Document and AuthenticDesktopControl Placeholder instead.

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 that 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](#)<sup>652</sup>. 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](#)<sup>604</sup>).
3. Drag the [AuthenticDesktopControl](#)<sup>635</sup> 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](#)<sup>643</sup> 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](#)<sup>649</sup> 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](#)<sup>652</sup>.
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](#)<sup>637</sup> property of the AuthenticDesktopControl.
- To retrieve commands organized according to their menu structure, use the [MainMenu](#)<sup>638</sup> property.
- To retrieve commands organized by the toolbar in which they appear, use the [Toolbars](#)<sup>638</sup> property.
- To send commands to Authentic Desktop, use the [Exec](#)<sup>639</sup> method.
- To query if a command is currently enabled or disabled, use the [QueryStatus](#)<sup>640</sup> 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](#)<sup>624</sup> section.

## General considerations

To automate the behaviour of Authentic Desktop, use the properties, methods, and events described for the [AuthenticDesktopControl](#)<sup>635</sup>, [AuthenticDesktopControl Document](#)<sup>643</sup>, and [AuthenticDesktopControl Placeholder](#)<sup>649</sup>.

For more complex access to Authentic Desktop functionality, consider using the following properties:

- [AuthenticDesktopControl.Application](#)<sup>636</sup>
- [AuthenticDesktopControlDocument.Document](#)<sup>644</sup>
- [AuthenticDesktopControlPlaceHolder.Project](#)<sup>650</sup>

These properties give you access to the Authentic Desktop automation interface (AuthenticDesktopAPI)

**Note:** To open a document, always use [AuthenticDesktopControlDocument.Open](#)<sup>646</sup> or [AuthenticDesktopControlDocument.New](#)<sup>646</sup> on the appropriate document control. To open a project, always use [AuthenticDesktopControlPlaceHolder.OpenProject](#)<sup>651</sup> 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](#)<sup>611</sup>.

## 14.4.5 ActiveX Integration Examples

This section contains examples of Authentic Desktop document-level integration using different container environments and programming languages. Source code for all examples is available in the folder `<ApplicationFolder>\Examples\ActiveX` of your Authentic Desktop installation.

### 14.4.5.1 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](#)<sup>611</sup>).

#### 14.4.5.1.1 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](#)<sup>603</sup>.

## 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

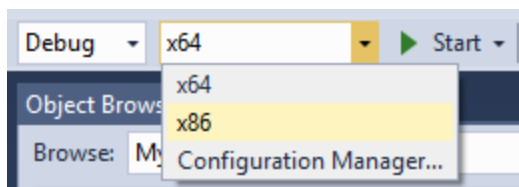
The sample application contains one instance of [AuthenticDesktopControlDocument](#)<sup>643</sup> and several instances of [AuthenticDesktopControlPlaceHolder](#)<sup>649</sup> controls. Double-check that the following properties of these controls are set as shown in the table below:

Control name	Property	Property value
axAuthenticDesktopControl	IntegrationLevel	ICActiveXIntegrationOnDocumentLevel
axAuthenticDesktopControlHelperWndEntities	PlaceholderWindowID	2
axAuthenticDesktopControlHelperWndAttributes	PlaceholderWindowID	1
axAuthenticDesktopControlHelperWndElements	PlaceholderWindowID	0
axAuthenticDesktopControlHelperWndInfo	PlaceholderWindowID	18
axAuthenticDesktopControlHelperWndProject	PlaceholderWindowID	4

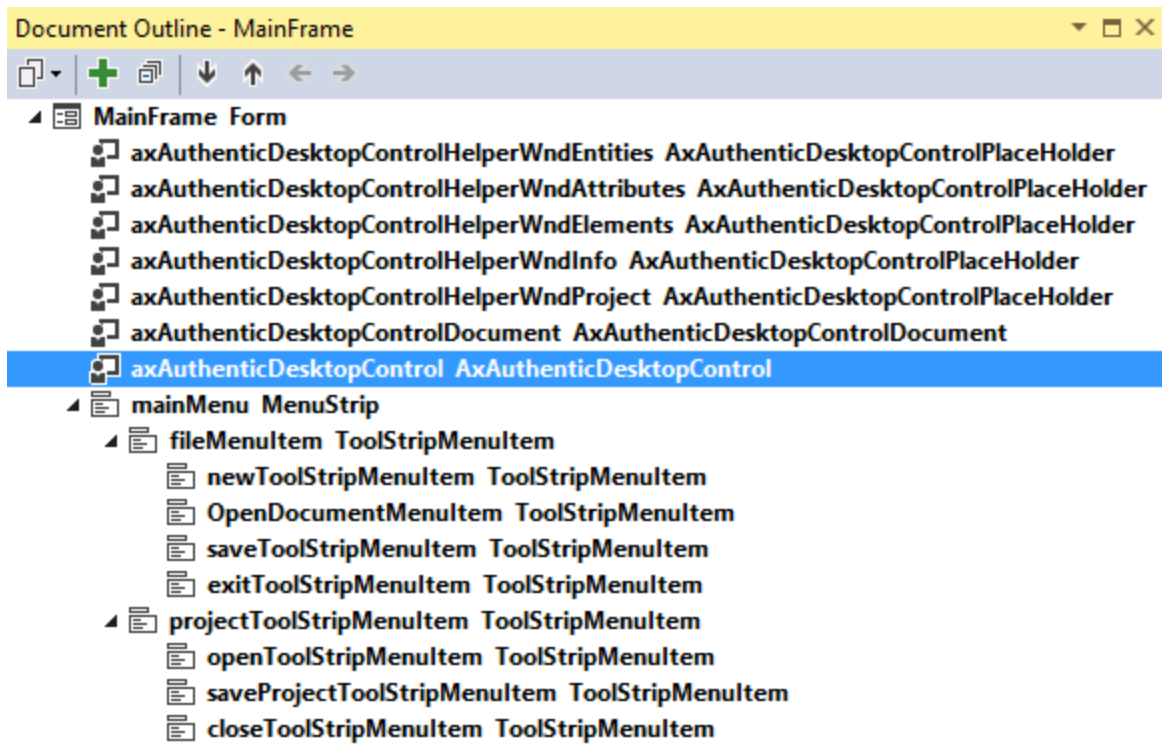
Here is how you can view or set the properties of an ActiveX control:

1. Open the **MDIMain.cs** form in the designer window.

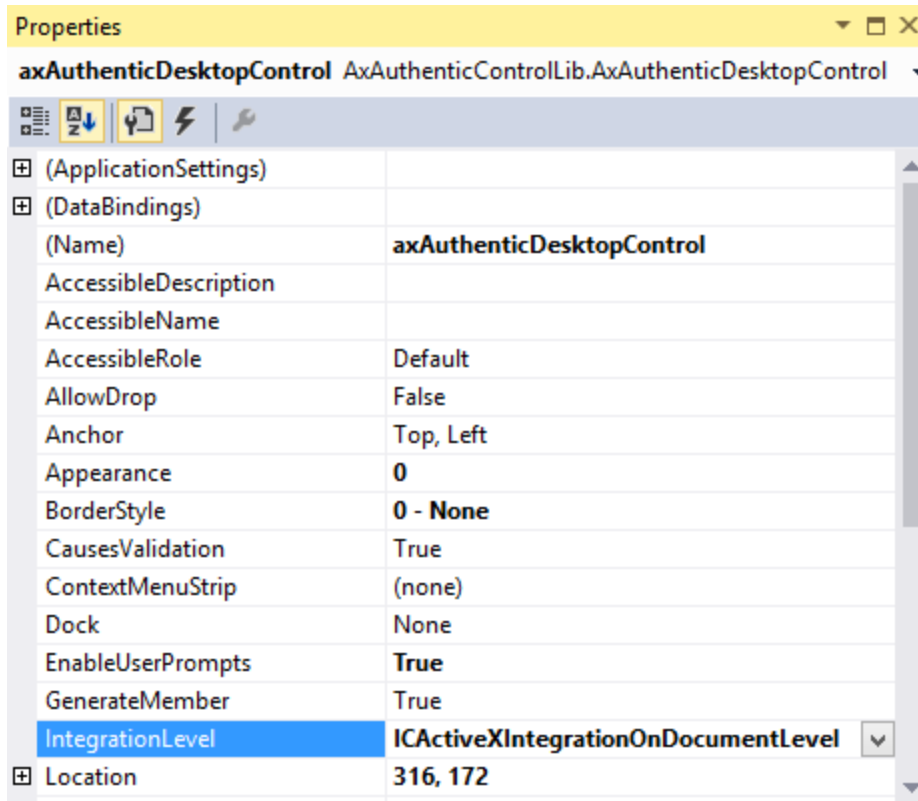
**Note:** On 64-bit Windows, it may be necessary to change the build configuration of the Visual Studio solution to "x86" **before** opening the designer window. If you need to build the sample as a 64-bit application, see [Prerequisites](#)<sup>603</sup>.



2. Open the **Document Outline** window of Visual Studio (On the **View** menu, click **Other Windows | Document Outline**).



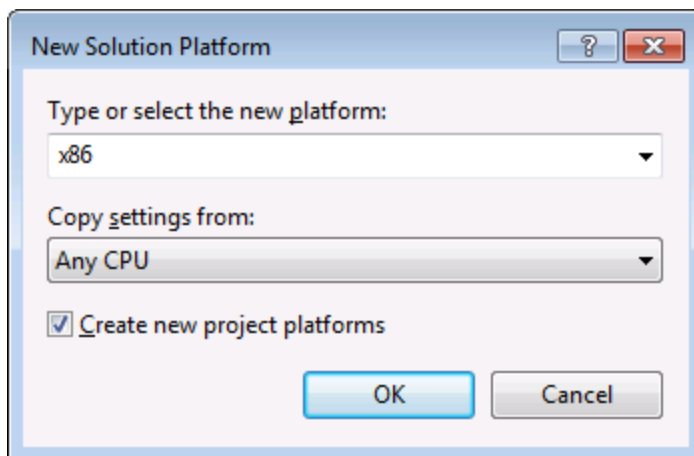
3. Click an ActiveX control in the **Document Outline** window, and edit its required property in the **Properties** window, for example:



### IntegrationLevel

#### 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.

## 14.4.5.2 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](#)<sup>603</sup>).

- `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](#)<sup>616</sup>.

### Rules for mapping the ActiveX Control names to Java

For the documentation of ActiveX controls, see [Object Reference](#)<sup>632</sup>. 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
AuthenticDesktopControlEvents: Events interface for the AuthenticDesktopControl
AuthenticDesktopControlEventsDefaultHandler: Default handler for
AuthenticDesktopControlEvents
```

### Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

Interface	Java name
AuthenticDesktopControlDocument, method New	newDocument
Document, method SetEncoding	setFileEncoding
AuthenticView, method Goto	gotoElement
AuthenticRange, method Goto	gotoElement
AuthenticRange, method Clone	cloneRange

### 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](#) <sup>616</sup>
- [Creating the ActiveX Controls](#) <sup>618</sup>
- [Loading Data in the Controls](#) <sup>619</sup>
- [Basic Event Handling](#) <sup>619</sup>
- [Menus](#) <sup>620</sup>
- [UI Update Event Handling](#) <sup>622</sup>
- [Creating an XML Tree](#) <sup>623</sup>

#### 14.4.5.2.1 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:

.classpath	Eclipse project helper file
.project	Eclipse project file
AltovaAutomation.dll	Java-COM bridge: DLL part (for the 32-bit installation)
AltovaAutomation_x64.dll	Java-COM bridge: DLL part (for the 64-bit installation)
AltovaAutomation.jar	Java-COM bridge: Java library part
AuthenticActiveX.jar	Java classes of the Authentic Desktop ActiveX control
AuthenticActiveX_JavaDoc.zip	Javadoc file containing help documentation for the Java API
AuthenticDesktopContainer.java	Java example source code
AuthenticDesktopContainerEventHandler.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.
XMLTreeDialog.java	Java example source code

## 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](#)<sup>618</sup>: Starts Authentic Desktop, which is registered as an automation server, or activates Authentic Desktop if it is already running.
- [Loading Data in the Controls](#)<sup>619</sup>: Locates one of the example documents installed with Authentic Desktop and opens it.
- [Basic Event Handling](#)<sup>619</sup>: Changes the view of all open documents to Browser View. The code also shows how to iterate through open documents.
- [Menus](#)<sup>620</sup>: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- [UI Update Event Handling](#)<sup>622</sup>: Shows how to handle Authentic Desktop events.
- [Creating an XML Tree](#)<sup>623</sup>: 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:

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\Documents\\Altova\\
\\Authentic2024\\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](#)<sup>603</sup>).
2. Open a command prompt window, change the current directory to the sample Java project folder, and type:

```
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](#)<sup>603</sup>).
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**.

### 14.4.5.2.2 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.

```
01 /**
```

```

02  * Authentic Desktop manager control - always needed
03  */
04  public static AuthenticDesktopControl      authenticDesktopControl = null;
05
06  /**
07  * 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
18  // Create the Authentic Desktop ActiveX control, The parameter determines that we want
// to place document controls and place-holder controls individually.
19  // It gives us full control over the menu, as well.
20  authenticDesktopControl = new AuthenticDesktopControl(
    ICActiveXIntegrationLevel.ICActiveXIntegrationOnDocumentLevel.getValue() );
21
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 ) );

```

### 14.4.5.2.3 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\\Authentic2024\\AuthenticExamples\\";
3  authenticDesktopProjectToolWindow = new
AuthenticDesktopControlPlaceHolder( XMLSpyControlPlaceholderWindow.XMLSpyControlProjectWind
owToolWnd.getValue() );

```

### 14.4.5.2.4 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
02     btnOpenPxf.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04             try {
05                 authenticDesktopControl.open( strExamplesFolder + "OrgChart.pxf" );
06             } catch (AutomationException e1) {
07                 e1.printStackTrace();
08             }
09         }
10     } );
11     public void onOpenedOrFocused( String i_strFileName, boolean
i_bOpenWithThisControl, boolean i_bFileAlreadyOpened ) throws AutomationException
12     {
13         // Handle the New/Open events coming from the Project tree or from the menus
14         if ( !i_bFileAlreadyOpened )
15         {
16             // This is basically an SDI interface, so open the file in the already existing
document control
17             try {
18                 AuthenticDesktopContainer.authenticDesktopDocument.open( i_strFileName );
19                 AuthenticDesktopContainer.authenticDesktopDocument.requestFocusInWindow();
20             } catch (Exception e) {
21                 e.printStackTrace();
22             }
23         }
24     }

```

#### 14.4.5.2.5 Menus

The code listing below shows how menu items can be created. Each `Authentic DesktopCommand` 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](#)<sup>622</sup>).

```

01 // Load the file menu when the button is pressed
02     btnMenu.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04             try {
05                 // Create the menubar that will be attached to the frame
06                 MenuBar mb = new MenuBar();
07                 // Load the main menu's first item - the File menu
08                 XMLSpyCommand xmlSpyMenu =
xmlSpyControl.getMainMenu().getSubCommands().getItem( 0 );
09                 // Create Java menu items from the Commands objects
10                 Menu fileMenu = new Menu();
11                 handlerObject.fillMenu( fileMenu, xmlSpyMenu.getSubCommands() );

```

```

12         fileMenu.setLabel( xmlSpyMenu.getLabel().replace( "&", "" ) );
13         mb.add( fileMenu );
14         frame.setMenuBar( mb );
15         frame.validate();
16     } catch (AutomationException e1) {
17         e1.printStackTrace();
18     }
19     // Disable the button when the action has been performed
20     ((AbstractButton) e.getSource()).setEnabled( false );
21 }
22 } ) ;
23 /** * Populates a menu with the commands and submenus contained in an XMLSpyCommands
object */
24     public void fillMenu(Menu newMenu, XMLSpyCommands xmlSpyMenu) throws
AutomationException
25     {
26         // For each command/submenu in the xmlSpyMenu
27         for ( int i = 0 ; i < xmlSpyMenu.getCount() ; ++i )
28         {
29             XMLSpyCommand xmlSpyCommand = xmlSpyMenu.getItem( i );
30             if ( xmlSpyCommand.getIsSeparator() )
31                 newMenu.addSeparator();
32             else
33             {
34                 XMLSpyCommands subCommands = xmlSpyCommand.getSubCommands();
35                 // Is it a command (leaf), or a submenu?
36                 if ( subCommands.isNull() || subCommands.getCount() == 0 )
37                 {
38                     // Command -> add it to the menu, set its ActionCommand to its ID and store it
in the menuMap
39                     MenuItem mi = new MenuItem( xmlSpyCommand.getLabel().replace( "&", "" ) );
40                     mi.setActionCommand( "" + xmlSpyCommand.getID() );
41                     mi.addActionListener( this );
42                     newMenu.add( mi );
43                     menuMap.put( xmlSpyCommand.getID(), mi );
44                 }
45                 else
46                 {
47                     // Submenu -> create submenu and repeat recursively
48                     Menu newSubMenu = new Menu();
49                     fillMenu( newSubMenu, subCommands );
50                     newSubMenu.setLabel( xmlSpyCommand.getLabel().replace( "&", "" ) );
51                     newMenu.add( newSubMenu );
52                 }
53             }
54         }
55     }
56
57     /**
58     * Action handler for the menu items
59     * Called when the user selects a menu item; the item's action command corresponds to
the command table for XMLSpy
60     */
61     public void actionPerformed( ActionEvent e )
62     {
63         try

```

```

64  {
65      int iCmd = Integer.parseInt( e.getActionCommand() );
66      // Handle explicitly the Close commands
67      switch ( iCmd )
68      {
69          case 57602:      // Close
70          case 34050:      // Close All
71              AuthenticDesktopContainer.initXmlSpyDocument();
72              break;
73          default:
74              AuthenticDesktopContainer.xmlSpyControl.exec( iCmd );
75              break;
76      }
77  }
78  catch ( Exception ex )
79  {
80      ex.printStackTrace();
81  }
82
83  }

```

#### 14.4.5.2.6 UI Update Event Handling

The code listing below shows how a UI-Update event handler can be created.

```

01  /**
02   * Call-back from the XMLSpyControl.
03   * Called to enable/disable commands
04   */
05  @Override
06  public void onUpdateCmdUI() throws AutomationException
07  {
08      // A command should be enabled if the result of queryStatus contains the Supported
09      (1) and Enabled (2) flags
10      for ( java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet() )
11      pair.getValue().setEnabled( AuthenticDesktopContainer.authenticDesktopControl.queryStatus(
12      pair.getKey() ) > 2 );
13  }
14  /**
15   * Call-back from the XMLSpyControl.
16   * Usually called while enabling/disabling commands due to UI updates
17   */
18  @Override
19  public boolean onIsActiveEditor( String i_strFilePath ) throws AutomationException
20  {
21      try {
22          return
23          AuthenticDesktopContainer.authenticDesktopDocument.getDocument().getFullName().equalsIgnore
24          Case( i_strFilePath );
25      } catch ( Exception e ) {
26          return false;
27      }
28  }

```

```
23     }
24 }
```

#### 14.4.5.2.7 Creating an XML Tree

The listing below loads an XML data object as nodes in a tree.

```
01 // access required XMLSpy Java-COM classes
02 import com.altova.automation.XMLSpy.XMLData;
03
04 // access AWT and Swing components
05 import java.awt.*;
06 import javax.swing.*;
07 import javax.swing.tree.*;
08
09 /**
10  * A simple example of a tree control loading the structure from an XMLData object.
11  * The class receives an XMLData object, loads its nodes in a JTree, and prepares
12  * for modal activation.
13  *
14  * Feel free to modify and extend this sample.
15  *
16  * @author Altova GmbH
17  */
18 class XMLTreeDialog extends JDialog
19 {
20     /**
21      * The tree control
22      */
23     private JTree myTree;
24
25     /**
26      * Root node of the tree control
27      */
28     private DefaultMutableTreeNode top ;
29
30     /**
31      * Constructor that prepares the modal dialog containing the filled tree control
32      * @param xml    The data to be displayed in the tree
33      * @param parent  Parent frame
34      */
35     public XMLTreeDialog( XMLData xml, Frame parent )
36     {
37         // Construct the modal dialog
38         super( parent, "XML tree", true );
39         // Arrange controls in the dialog
40         top = new DefaultMutableTreeNode("root");
41         myTree = new JTree(top);
42         setContentPane( new JScrollPane( myTree ) );
43         // Build up the tree
44         fillTree( top, xml );
45         myTree.expandRow( 0 );

```

```
46 }
47
48 /**
49  * Loads the nodes of an XML element under a given tree node
50  * @param node Target tree node
51  * @param elem Source XML element
52  */
53 private void fillTree( DefaultMutableTreeNode node, XMLData elem)
54 {
55     try
56     {
57         // There are several ways to iterate through child elements: either using the
58         // getFirstChild/getNextChild,
59         // or by incrementing an index up to countChildren and calling getChild [as shown
60         // below].
61         // If you only want to get children of one kind, you should use
62         // countChildrenKind/getChildKind,
63         // or provide a kind to the getFirstChild before iterating with the getNextChild.
64         int nSize = elem.countChildren() ;
65         for ( int i = 0 ; i < nSize ; ++i)
66         {
67             // Create a new tree node for each child element, and continue recursively
68             XMLData newElem = elem.getChild(i) ;
69             DefaultMutableTreeNode newNode = new DefaultMutableTreeNode( newElem.getName() )
70             ;
71             node.add( newNode ) ;
72             fillTree( newNode, newElem ) ;
73         }
74     }
75     catch (Exception e)
76     {
77         e.printStackTrace();
78     }
79 }
```

## 14.4.6 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](#)<sup>639</sup> or the [AuthenticDesktopControlDocument.Exec](#)<sup>645</sup> methods. To query the status of a command, use the



[AuthenticDesktopControl.QueryStatus](#)<sup>640</sup> or [AuthenticDesktopControlDocument.QueryStatus](#)<sup>646</sup> methods.

Depending on the edition of Authentic Desktop you have installed, some of these commands might not be supported.

### 14.4.6.1 "File" Menu

The "File" menu has the following commands:

Menu item	Command name	ID
New...	ID_FILE_NEW	57600
Open...	ID_FILE_OPEN	57601
Reload	IDC_FILE_RELOAD	34065
Encoding...	IDC_ENCODING	34061
Close	ID_FILE_CLOSE	57602
Close All	IDC_CLOSE_ALL	34050
Close All But Active	IDC_CLOSE_OTHERS	34271
Save	ID_FILE_SAVE	57603
Save As...	ID_FILE_SAVE_AS	57604
Save All	ID_FILE_SAVE_ALL	34208
Send by Mail...	ID_FILE_SEND_MAIL	57612
Print...	ID_FILE_PRINT	57607
Print Preview	IDC_PRINT_PREVIEW	34104
Print Setup...	ID_FILE_PRINT_SETUP	57606
Recent File	ID_FILE_MRU_FILE1	57616
Exit	ID_APP_EXIT	57665

### 14.4.6.2 "Edit" Menu

The "Edit" menu has the following commands:

Menu item	Command name	ID
Undo	ID_EDIT_UNDO	57643
Redo	ID_EDIT_REDO	57644
Cut	ID_EDIT_CUT	57635
Copy	ID_EDIT_COPY	57634
Paste	ID_EDIT_PASTE	57637
Delete	ID_EDIT_CLEAR	57632
Select All	ID_EDIT_SELECT_ALL	57642
Find...	ID_EDIT_FIND	57636
Find Next	ID_EDIT_REPEAT	57640
Replace...	ID_EDIT_REPLACE	57641

### 14.4.6.3 "Project" Menu

The "Project" menu has the following commands:

Menu item	Command name	ID
New Project	IDC_ICPROJECTGUI_NEW	37200
Open Project...	IDC_ICPROJECTGUI_OPEN	37201
Reload Project	IDC_ICPROJECTGUI_RELOAD	37202
Close Project	IDC_ICPROJECTGUI_CLOSE	37203
Save Project	IDC_ICPROJECTGUI_SAVE	37204
Save Project As...	IDC_ICPROJECTGUI_SAVE_AS	37207
Enable Source Control	ID_SCC_ENABLE	38602
Add Files to Project...	IDC_ICPROJECTGUI_ADD_FILES_TO_PROJECT	37205
Add Global Resource to Project...	IDC_ICPROJECTGUI_ADD_GLOBAL_RESOURCE_TO_PROJECT	37239
Add URL to Project...	IDC_ICPROJECTGUI_ADD_URL_TO_PROJECT	37206

Menu item	Command name	ID
Add Active File to Project	IDC_ICPROJECTGUI_ADD_ACTIVE_FILE_TO_PROJECT	37208
Add Active and Related Files to Project	IDC_ICPROJECTGUI_ADD_ACTIVE_AND_RELATED_FILES_TO_PROJECT	37209
Add Project Folder to Project...	IDC_ICPROJECTGUI_ADD_FOLDER_TO_PROJECT	37210
Add External Folder to Project...	IDC_ICPROJECTGUI_ADD_EXT_FOLDER_TO_PROJECT	37211
Add External Web Folder to Project...	IDC_ICPROJECTGUI_ADD_EXT_URL_FOLDER_TO_PROJECT	37212
Script settings...	IDC_PROJECT_SCRIPT_SETTINGS	34136
Properties...	IDC_ICPROJECTGUI_PROJECT_PROPERTIES	37223
Recent Project	IDC_ICPROJECTGUI_RECENT	37224

### 14.4.6.4 "XML" Menu

The "XML" menu has the following commands:

Menu item	Command name	ID
Check Well-Formedness	IDC_CHECK_WELL_FORM	34049
Validate XML	IDC_VALIDATE	32954

### 14.4.6.5 "XSL/XQuery" Menu

The "XSL/XQuery" menu has the following commands:

Menu item	Command name	ID
XSL Transformation	IDC_TRANSFORM_XSL	33006
XSL-FO Transformation	IDC_TRANSFORM_XSLFO	33007
XSL Parameters / XQuery Variables...	IDC_TRANSFORM_XSL_PARAMS	33008

### 14.4.6.6 "Authentic" Menu

The "Authentic" menu has the following commands:

Menu item	Command name	ID
New Document...	IDC_AUTHENTIC_NEW_FILE	34036
Edit Database Data...	IDC_AUTHENTIC_EDIT_DB	34035
Edit StyleVision Stylesheet	IDC_EDIT_SPS	34060
Select New Row with XML Data for Editing...	IDC_CHANGE_WORKING_DB_XML_CELL	32861
XML Signature...	IDC_AUTHENTICGUI_XMLSIGNATURE	32862
Define XML Entities...	IDC_DEFINE_ENTITIES	32805
Hide Markup	IDC_MARKUP_HIDE	32855
Show Small Markup	IDC_MARKUP_SMALL	32858
Show Large Markup	IDC_MARKUP_LARGE	32856
Show Mixed Markup	IDC_MARKUP_MIXED	32857
Toggle Bold	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEBOLD	32813
Toggle Italic	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEITALIC	32814
Toggle Underline	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEUNDERLINE	32815
Toggle Strikethrough	IDC_AUTHENTICGUI_RICHEDIT_TOGGLESTRIKETHROUGH	32816
Foreground Color	IDC_AUTHENTICGUI_RICHEDIT_COLOR_FOREGROUND	32824
Background Color	IDC_AUTHENTICGUI_RICHEDIT_COLOR_BACKGROUND	32830
Align Left	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_LEFT	32818
Center	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_CENTER	32819
Align Right	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_RIGHT	32820
Append Row	IDC_ROW_APPEND	32806
Insert Row	IDC_ROW_INSERT	32809

Menu item	Command name	ID
Duplicate Row	IDC_ROW_DUPLICATE	32808
Move Row Up	IDC_ROW_MOVE_UP	32811
Move Row Down	IDC_ROW_MOVE_DOWN	32810
Delete Row	IDC_ROW_DELETE	32807
Generate an HTML document	IDC_PXF_GENERATE_HTML	34283
Generate an RTF document	IDC_PXF_GENERATE_RTF	34284
Generate a PDF document	IDC_PXF_GENERATE_PDF	34285
Generate a Word 2007+ document	IDC_PXF_GENERATE_DOCX	34286
Trusted Locations...	IDC_TRUSTED_LOCATIONS	34288

### 14.4.6.7 "View" Menu

The "View" menu has the following commands:

Menu item	Command name	ID
Authentic View	IDC_VIEW_CONTENT	34177
Browser View	IDC_VIEW_BROWSER	34176
Text View Settings	IDC_TEXTVIEW_SETTINGS	34119

### 14.4.6.8 "Browser" Menu

The "Browser" menu has the following commands:

Menu item	Command name	ID
Back	IDC_STEP_BACK	32958
Forward	IDC_STEP_FORWARD	32957
Stop	IDC_BROWSER_STOP	34047
Refresh	IDC_BROWSER_REFRESH	34046
Largest	IDC_BROWSER_FONT_LARGEST	34041
Larger	IDC_BROWSER_FONT_LARGE	34040

Menu item	Command name	ID
Medium	IDC_BROWSER_FONT_MEDIUM	34042
Smaller	IDC_BROWSER_FONT_SMALL	34043
Smallest	IDC_BROWSER_FONT_SMALLEST	34044

### 14.4.6.9 "Tools" Menu

The "Tools" menu has the following commands:

Menu item	Command name	ID
Spelling...	IDC_SPELL_CHECK	34154
Spelling Options...	IDC_SPELL_OPTIONS	34155
Scripting Editor...	ID_SCRIPTFORMEDITOR_EDIT_PROJECT	39666
none	ID_SCRIPTFORMEDITOR_EXECUTE_MACRO_MENU_UPPDATE	39600
	IDC_TOOLS_ENTRY	34292
Global Resources	IDC_GLOBALRESOURCES	37401
	IDC_GLOBALRESOURCES_SUBMENUENTR Y1	37408
Customize...	IDC_APP_TOOLS_CUSTOMIZE	32959
Options...	IDC_SETTINGS	34133
	ID_SCRIPTING_MACROITEMS	34249

### 14.4.6.10 "Window" Menu

The "Window" menu has the following commands:

Menu item	Command name	ID
Cascade	ID_WINDOW_CASCADE	57650
Tile horizontally	ID_WINDOW_TILE_HORZ	57651
Tile vertically	ID_WINDOW_TILE_VERT	57652

Menu item	Command name	ID
Project window	IDC_PROJECT_WINDOW	34128
Info window	IDC_INFO_WINDOW	34085
Entry Helpers	IDC_ENTRY_HELPERS	34062
Output windows	IDC_OUTPUT_DIALOGBARS	34004
Project and Entry Helpers	IDC_PROJECT_ENTRYHELPERS	34006
All on/off	IDC_ALL_BARS	34031

### 14.4.6.11 "Help" Menu

The "Help" menu has the following commands:

Menu item	Command name	ID
Table of Contents...	IDC_HELP_CONTENTS	32966
Index...	IDC_HELP_INDEX	32967
Search...	IDC_HELP_SEARCH	32969
Keyboard Map...	IDC_HELP_KEYMAPDLG	32968
Software Activation...	IDC_ACTIVATION	32970
Order Form...	IDC_OPEN_ORDER_PAGE	32971
Registration...	IDC_REGISTRATION	32972
Check for Updates...	IDC_CHECK_FOR_UPDATES	32973
XMLSpy Product Comparison...	IDC_PRODUCT_COMPARISON	32955
Support Center...	IDC_OPEN_SUPPORT_PAGE	32961
FAQ on the Web...	IDC_OPEN_FAQ_PAGE	32962
Download Components and Free Tools...	IDC_OPEN_COMPONENTS_PAGE	32963
Authentic on the Internet..	IDC_OPEN_HOME_PAGE	32964
Authentic Training...	IDC_OPEN_TRAINING_PAGE	32965
About Authentic...	ID_APP_ABOUT	57664

## 14.4.7 Object Reference

### Objects:

[Authentic DesktopCommand](#) <sup>632</sup>

[Authentic DesktopCommands](#) <sup>634</sup>

[AuthenticDesktopControl](#) <sup>635</sup>

[AuthenticDesktopControlDocument](#) <sup>643</sup>

[AuthenticDesktopControlPlaceHolder](#) <sup>649</sup>

To give access to standard Authentic Desktop functionality, objects of the **Authentic Desktop automation interface** can be accessed as well. See [AuthenticDesktopControl.Application](#) <sup>636</sup>, [AuthenticDesktopControlDocument.Document](#) <sup>644</sup> and [AuthenticDesktopControlPlaceHolder.Project](#) <sup>650</sup> for more information.

### 14.4.7.1 Authentic DesktopCommand

#### Properties:

[ID](#) <sup>633</sup>

[Label](#) <sup>633</sup>

[Name](#) <sup>633</sup>

[IsSeparator](#) <sup>633</sup>

[ToolTip](#) <sup>634</sup>

[StatusText](#) <sup>634</sup>

[Accelerator](#) <sup>633</sup>

[SubCommands](#) <sup>634</sup>

#### 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.

The Command object is...	When...
An executable command	<ul style="list-style-type: none"> <li>• <code>ID</code> is greater than zero</li> <li>• <code>IsSeparator</code> is false</li> <li>• <code>SubCommands</code> is empty</li> </ul>
A command container	<ul style="list-style-type: none"> <li>• <code>ID</code> is zero</li> <li>• <code>IsSeparator</code> is false</li> <li>• <code>SubCommands</code> contains a collection of Command objects.</li> </ul>
Separator	<ul style="list-style-type: none"> <li>• <code>ID</code> is zero</li> <li>• <code>IsSeparator</code> is true</li> </ul>



### 14.4.7.1.1 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`.

### 14.4.7.1.2 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](#)<sup>639</sup>) or query its status (using [QueryStatus](#)<sup>640</sup>). If the command is a container for other commands (for example, a top-level menu), or a separator, the ID is 0.

### 14.4.7.1.3 IsSeparator

**Property:** IsSeparator as `boolean`

**Description:**

The property returns `true` if the command object is a menu separator; `false` otherwise. See also [Command](#)<sup>632</sup>.

### 14.4.7.1.4 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.

### 14.4.7.1.5 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.

#### 14.4.7.1.6 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.

#### 14.4.7.1.7 SubCommands

**Property:** SubCommands as [Commands](#)<sup>634</sup>

**Description:**

The `SubCommands` property gets the collection of [Command](#)<sup>632</sup> 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`.

#### 14.4.7.1.8 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.

### 14.4.7.2 Authentic DesktopCommands

**Properties:**

[Count](#)<sup>635</sup>  
[Item](#)<sup>635</sup>

**Description:**

Collection of [Command](#)<sup>632</sup> objects to get access to command labels and IDs of the `AuthenticDesktopControl`. Those commands can be executed with the [Exec](#)<sup>639</sup> method and their status can be queried with [QueryStatus](#)<sup>640</sup>.

### 14.4.7.2.1 Count

**Property:** Count as [long](#)

**Description:**

Number of [Command](#)<sup>632</sup> objects on this level of the collection.

### 14.4.7.2.2 Item

**Property:** Item (n as [long](#)) as [Command](#)<sup>632</sup>

**Description:**

Gets the command with the index n in this collection. Index is 1-based.

## 14.4.7.3 AuthenticDesktopControl

**Properties:**

[IntegrationLevel](#)<sup>637</sup>

[Appearance](#)<sup>636</sup>

[Application](#)<sup>636</sup>

[BorderStyle](#)<sup>636</sup>

[CommandsList](#)<sup>637</sup>

[EnableUserPrompts](#)<sup>637</sup>

[MainMenu](#)<sup>638</sup>

[Toolbars](#)<sup>638</sup>

**Methods:**

[Open](#)<sup>640</sup>

[Exec](#)<sup>639</sup>

[QueryStatus](#)<sup>640</sup>

**Events:**

[OnUpdateCmdUI](#)<sup>642</sup>

[OnOpenedOrFocused](#)<sup>642</sup>

[OnCloseEditingWindow](#)<sup>641</sup>

[OnFileChangedAlert](#)<sup>641</sup>

[OnDocumentOpened](#)<sup>641</sup>

[OnValidationWindowUpdated](#)<sup>643</sup>

This object is a complete ActiveX control and should only be visible if the Authentic Desktop library is used in the Application Level mode.

### 14.4.7.3.1 Properties

The following properties are defined:

[IntegrationLevel](#)<sup>637</sup>  
[EnableUserPrompts](#)<sup>637</sup>  
[Appearance](#)<sup>636</sup>  
[BorderStyle](#)<sup>636</sup>

Command related properties:

[CommandsList](#)<sup>637</sup>  
[MainMenu](#)<sup>638</sup>  
[Toolbars](#)<sup>638</sup>

Access to AuthenticDesktopAPI:

[Application](#)<sup>636</sup>

#### 14.4.7.3.1.1 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.

#### 14.4.7.3.1.2 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.

#### 14.4.7.3.1.3 BorderStyle

**Property:** BorderStyle as `short`

**Dispatch Id:** -504

**Description:**

A value of 1 displays the control with a thin border. Default value is 0.

### 14.4.7.3.1.4 *CommandsList*

**Property:** `CommandList` as [Commands](#)<sup>634</sup> (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](#)<sup>638</sup>. To get toolbar commands, use [Toolbars](#)<sup>638</sup>.

```
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*

### 14.4.7.3.1.5 *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*.

### 14.4.7.3.1.6 *IntegrationLevel*

**Property:** `IntegrationLevel` as [IActiveXIntegrationLevel](#)<sup>652</sup>

**Dispatch Id:** 1000

**Description:**

The `IntegrationLevel` property determines the operation mode of the control. See also [Integration at Application Level](#)<sup>606</sup> and [Integration at Document Level](#)<sup>608</sup> for more information.

**Note:** It is important to set this property immediately after the creation of the `AuthenticDesktopControl` object.

#### 14.4.7.3.1.7 *MainMenu*

**Property:** `MainMenu` as [Command](#)<sup>632</sup> (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 need a recursive function.

```
public void GetAuthenticMenus()
{
    // 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*

#### 14.4.7.3.1.8 *Toolbars*

**Property:** `Toolbars` as [Commands](#)<sup>634</sup> (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, cmd.Label.Replace("&",
                ""));
            }
        }
    }
}
```

*C# example*

### 14.4.7.3.2 Methods

The following methods are defined:

[Open](#) <sup>640</sup>  
[Exec](#) <sup>639</sup>  
[QueryStatus](#) <sup>640</sup>

#### 14.4.7.3.2.1 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](#)<sup>637</sup>. To retrieve the status of any command, use [QueryStatus](#)<sup>640</sup>.

#### 14.4.7.3.2.2 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](#)<sup>646</sup> and [AuthenticDesktopControlPlaceholder.OpenProject](#)<sup>651</sup>.

#### 14.4.7.3.2.3 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.

#### 14.4.7.3.3 Events

The `AuthenticDesktopControl` ActiveX control provides the following connection point events:

[OnUpdateCmdUI](#)<sup>642</sup>  
[OnOpenedOrFocused](#)<sup>642</sup>  
[OnCloseEditingWindow](#)<sup>641</sup>



[OnFileChangedAlert](#) <sup>641</sup>  
[OnDocumentOpened](#) <sup>641</sup>  
[OnValidationWindowUpdated](#) <sup>643</sup>

#### 14.4.7.3.3.1 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.

#### 14.4.7.3.3.2 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](#) <sup>648</sup> instead.

#### 14.4.7.3.3.3 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.

#### 14.4.7.3.3.4 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).

#### 14.4.7.3.3.5 *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.

#### 14.4.7.3.3.6 *OnToolWindowUpdated*

**Event:** OnToolWindowUpdated (pToolWnd as long )

**Dispatch Id:** 1006**Description:**

This event is triggered when the tool window is updated.

#### 14.4.7.3.3.7 *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](#)<sup>640</sup>. Do not perform long operations in this callback.

### 14.4.7.3.3.8 OnValidationWindowUpdated

**Event:** OnValidationWindowUpdated()

**Dispatch Id:** 3

**Description:**

This event is triggered whenever the validation output window is updated with new information.

## 14.4.7.4 AuthenticDesktopControlDocument

**Properties:**

[Appearance](#) <sup>644</sup>  
[BorderStyle](#) <sup>644</sup>  
[Document](#) <sup>644</sup>  
[IsModified](#) <sup>644</sup>  
[Path](#) <sup>645</sup>  
[ReadOnly](#) <sup>645</sup>

**Methods:**

[Exec](#) <sup>645</sup>  
[New](#) <sup>646</sup>  
[Open](#) <sup>646</sup>  
[QueryStatus](#) <sup>646</sup>  
[Reload](#) <sup>647</sup>  
[Save](#) <sup>647</sup>  
[SaveAs](#) <sup>647</sup>

**Events:**

[OnDocumentOpened](#) <sup>648</sup>  
[OnDocumentClosed](#) <sup>648</sup>  
[OnModifiedFlagChanged](#) <sup>649</sup>  
[OnFileChangedAlert](#) <sup>649</sup>  
[OnActivate](#) <sup>648</sup>

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.

### 14.4.7.4.1 Properties

The following properties are defined:

[ReadOnly](#) <sup>645</sup>  
[IsModified](#) <sup>644</sup>  
[Path](#) <sup>645</sup>

[Appearance](#) <sup>644</sup>  
[BorderStyle](#) <sup>644</sup>

Access to AuthenticDesktopAPI:  
[Document](#) <sup>644</sup>

#### 14.4.7.4.1.1 *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.

#### 14.4.7.4.1.2 *BorderStyle*

**Property:** BorderStyle as `short`

**Dispatch Id:** -504

**Description:**

A value of 1 displays the control with a thin border. Default value is 0.

#### 14.4.7.4.1.3 *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.

#### 14.4.7.4.1.4 *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.

#### 14.4.7.4.1.5 Path

**Property:** Path as `string`

**Dispatch Id:** 1005

**Description:**

Sets or gets the full path name of the document loaded into the control.

#### 14.4.7.4.1.6 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.

### 14.4.7.4.2 Methods

The following methods are defined:

Document handling:

[New](#) <sup>646</sup>

[Open](#) <sup>646</sup>

[Reload](#) <sup>647</sup>

[Save](#) <sup>647</sup>

[SaveAs](#) <sup>647</sup>

Command Handling:

[Exec](#) <sup>645</sup>

[QueryStatus](#) <sup>646</sup>

#### 14.4.7.4.2.1 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](#)<sup>638</sup> property of `AuthenticDesktopControl`. To get toolbar commands, use the [Toolbars](#)<sup>638</sup> property of the `AuthenticDesktopControl`.

#### 14.4.7.4.2.2 *New*

**Method:** `New ()` as `boolean`

**Dispatch Id:** 1000

**Description:**

This method initializes a new document inside the control.

#### 14.4.7.4.2.3 *Open*

**Method:** `Open (strFileName as string)` as `boolean`

**Dispatch Id:** 1001

**Description:**

`Open` loads the file `strFileName` as the new document into the control.

#### 14.4.7.4.2.4 *QueryStatus*

**Method:** `QueryStatus (nCmdID as long)` as `long`

**Dispatch Id:** 9

**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. The client should call the `QueryStatus` method of the document control if there is currently an active document available in the application.

#### 14.4.7.4.2.5 Reload

**Method:** Reload() as `boolean`

**Dispatch Id:** 1002

**Description:**

Reload updates the document content from the file system.

#### 14.4.7.4.2.6 Save

**Method:** Save() as `boolean`

**Dispatch Id:** 1003

**Description:**

Save saves the current document at the location [Path](#)<sup>645</sup>.

#### 14.4.7.4.2.7 SaveAs

**Method:** SaveAs (strFileName as `string`) as `boolean`

**Dispatch Id:** 1004

**Description:**

SaveAs sets [Path](#)<sup>645</sup> to *strFileName* and then saves the document to this location.

#### 14.4.7.4.3 Events

The AuthenticDesktopControlDocument ActiveX control provides following connection point events:

[OnDocumentOpened](#)<sup>648</sup>

[OnDocumentClosed](#)<sup>648</sup>

[OnModifiedFlagChanged](#)<sup>649</sup>

[OnFileChangedAlert](#)<sup>649</sup>

[OnActivate](#)<sup>648</sup>

[OnSetEditorTitle](#)<sup>649</sup>

#### 14.4.7.4.3.1 *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.

#### 14.4.7.4.3.2 *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.

#### 14.4.7.4.3.3 *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.

#### 14.4.7.4.3.4 *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.



#### 14.4.7.4.3.5 *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.

#### 14.4.7.4.3.6 *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.

#### 14.4.7.4.3.7 *OnSetEditorTitle*

**Event:** `OnSetEditorTitle ()`

**Dispatch Id:** 1006

**Description:**

This event is being raised when the contained document is being internally renamed.

### 14.4.7.5 AuthenticDesktopControlPlaceHolder

**Properties available for all kinds of placeholder windows:**

[PlaceholderWindowID](#) <sup>650</sup>

**Properties for project placeholder window:**

[Project](#) <sup>650</sup>

**Methods for project placeholder window:**

[OpenProject](#) <sup>651</sup>

[CloseProject](#) <sup>651</sup>

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.

### 14.4.7.5.1 Properties

The following properties are defined:

[PlaceholderWindowID](#) <sup>650</sup>

Access to AuthenticDesktopAPI:

[Project](#) <sup>650</sup>

#### 14.4.7.5.1.1 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.

#### 14.4.7.5.1.2 PlaceholderWindowID

**Property:** PlaceholderWindowID as [AuthenticDesktopControlPlaceholderWindow](#) <sup>652</sup>

**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](#) <sup>652</sup> enumeration. The control changes its state immediately and shows the new Authentic Desktop window.

#### 14.4.7.5.1.3 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](#) <sup>650</sup> with a value of `Authentic DesktopXProjectWindow (=3)`. The property is read-only.

### 14.4.7.5.2 Methods

The following method is defined:

[OpenProject](#)<sup>651</sup>  
[CloseProject](#)<sup>651</sup>

#### 14.4.7.5.2.1 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](#)<sup>650</sup> different to Authentic DesktopXProjectWindow (=3).

#### 14.4.7.5.2.2 CloseProject

**Method:** CloseProject ()

**Dispatch Id:** 4

**Description:**

CloseProject closes the project loaded by the control. The method will fail if the placeholder window has a [PlaceholderWindowID](#)<sup>650</sup> different to Authentic DesktopXProjectWindow (=3).

### 14.4.7.5.3 Events

The AuthenticDesktopControlPlaceholder ActiveX control provides following connection point events:

[OnModifiedFlagChanged](#)<sup>651</sup>

#### 14.4.7.5.3.1 OnModifiedFlagChanged

**Event:** OnModifiedFlagChanged (i\_bIsModified as boolean)

**Dispatch Id:** 1

**Description:**

This event gets triggered only for placeholder controls with a [PlaceholderWindowID](#)<sup>650</sup> of `AuthenticDesktopXProjectWindow (=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.

#### 14.4.7.5.3.2 OnSetLabel

**Event:** `OnSetLabel(i_strNewLabel as string)`

**Dispatch Id:** 1000

**Description:**

Raised when the title of the placeholder window is changed.

### 14.4.7.6 Enumerations

The following enumerations are defined:

#### [IActiveXIntegrationLevel](#)<sup>652</sup>

`AuthenticDesktopControlPlaceholderWindow`

##### 14.4.7.6.1 IActiveXIntegrationLevel

Possible values for the [IntegrationLevel](#)<sup>637</sup> property of the `AuthenticDesktopControl`.

```
IActiveXIntegrationOnApplicationLevel = 0
IActiveXIntegrationOnDocumentLevel = 1
```

##### 14.4.7.6.2 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
```

## 15 Appendices

These appendices contain technical information about Authentic Desktop and important licensing information. Each appendix contains sub-sections as given below:

### [Technical Data](#) <sup>654</sup>

- OS and memory requirements
- Altova XML Parser
- Altova XSLT and XQuery Engines
- Unicode support
- Internet usage

### [License Information](#) <sup>658</sup>

- Electronic software distribution
- Intellectual property rights and copyright
- End User License Agreement

## 15.1 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](#) <sup>654</sup>
- [Altova Engines](#) <sup>654</sup>
- [Unicode Support](#) <sup>655</sup>
- [Internet Usage](#) <sup>655</sup>

### 15.1.1 OS and Memory Requirements

#### Operating System

Altova software applications are available for the following platforms:

- Windows 10, Windows 11
- Windows Server 2016 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.

### 15.1.2 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.

### 15.1.3 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.

### 15.1.4 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.

## 15.2 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](#) at the [Altova website](#).

### 15.2.1 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](http://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 uploading the license file in the Software Activation dialog of your product.

You can purchase product licenses at <https://shop.altova.com/>.

#### 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.



## 15.2.2 Software Activation and License Metering

As part of Altova's Software Activation, the software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration, use, or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the software and to improve customer service. Activation is based on the exchange of license related data such as operating system, IP address, date/time, software version, and computer name, along with other information between your computer and an Altova license server.

Your Altova product has a built-in license metering module that further helps you avoid any unintentional violation of the End User License Agreement. Your product is licensed either as a single-user or multi-user installation, and the license-metering module makes sure that no more than the licensed number of users use the application concurrently.

This license-metering technology uses your local area network (LAN) to communicate between instances of the application running on different computers.

### Single license

When the application starts up, as part of the license metering process, the software sends a short broadcast datagram to find any other instance of the product running on another computer in the same network segment. If it doesn't get any response, it will open a port for listening to other instances of the application.

### Multi-user license

If more than one instance of the application is used within the same LAN, these instances will briefly communicate with each other on startup. These instances exchange key-codes in order to help you to better determine that the number of concurrent licenses purchased is not accidentally violated. This is the same kind of license metering technology that is common in the Unix world and with a number of database development tools. It allows Altova customers to purchase reasonably-priced concurrent-use multi-user licenses.

We have also designed the applications so that they send few and small network packets so as to not put a burden on your network. The TCP/IP ports (2799) used by your Altova product are officially registered with the IANA (see the [IANA Service Name Registry](#) for details) and our license-metering module is tested and proven technology.

If you are using a firewall, you may notice communications on port 2799 between the computers that are running Altova products. You are, of course, free to block such traffic between different groups in your organization, as long as you can ensure by other means, that your license agreement is not violated.

### Note about certificates

Your Altova application contacts the Altova licensing server ([link.altova.com](https://link.altova.com)) via HTTPS. For this communication, Altova uses a registered SSL certificate. If this certificate is replaced (for example, by your IT department or an external agency), then your Altova application will warn you about the connection being insecure. You could use the replacement certificate to start your Altova application, but you would be doing this at your own risk. If you see a *Non-secure connection* warning message, check the origin of the certificate and consult your IT team (who would be able to decide whether the interception and replacement of the Altova certificate should continue or not).

If your organization needs to use its own certificate (for example, to monitor communication to and from client machines), then we recommend that you install Altova's free license management software, [Altova LicenseServer](#), on your network. Under this setup, client machines can continue to use your organization's certificates, while Altova LicenseServer can be allowed to use the Altova certificate for communication with Altova.

### 15.2.3 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

## .NET,

- differences to Authentic Desktop standalone, 151
- integration of Authentic Desktop with, 149

## A

### Activating the software, 275

#### Active configuration,

- for global resources, 236

#### ActiveX,

- integration at application level, 606
- integration at document level, 608
- integration prerequisites, 603

#### ActiveX controls,

- adding to the Visual Studio Toolbox, 604
- support, 311

#### AI-Assistant,

- OpenAI API key for, 271

#### Alias,

- see Global Resources, 90

#### Altova Global Resources,

- see under Global Resources, 90

#### Altova products, 22

#### Altova support, 22

#### Altova XML Parser,

- about, 654

#### API,

- documentation, 325
- overview, 326

#### Append,

- row (in Authentic View), 224

#### Application,

- ActiveDocument, 359
- AddMacroMenuItem, 360
- AddXSLT\_XQParameter, 360
- Application, 360
- ClearMacroMenu, 361
- CurrentProject, 361

Dialogs, 362

Documents, 362

GetDatabaseImportElementList, 363

GetDatabaseSettings, 363

GetDatabaseTables, 364

GetExportSettings, 364

GetTextImportElementList, 365

GetTextImportExportSettings, 366

GetXSLT\_XQParameterCount, 366

GetXSLT\_XQParameterName, 366

GetXSLT\_XQParameterXPath, 366

ImportFromDatabase, 367

ImportFromSchema, 368

ImportFromText, 368

ImportFromWord, 369

NewProject, 370

OnBeforeOpenDocument, 357

OnBeforeOpenProject, 358

OnDocumentOpened, 358

OnProjectOpened, 359

OpenProject, 371

Parent, 371

Quit, 372

ReloadSettings, 372

RemoveXSLT\_XQParameter, 372

RunMacro, 373

ScriptingEnvironment, 373

ShowApplication, 373

ShowForm, 374

URLDelete, 375

URLMakeDirectory, 375

WarningNumber, 376

WarningText, 376

### Apply, 253

#### Assign,

- shortcut to a command, 242

#### ATL,

- plug-in sample files, 314

### Attribute preview, 261

#### Attribute values,

- entering in Authentic View, 36

#### Attributes entry helper,

- in Authentic View, 48

### Authentic Desktop,

- features, 22
- help, 22
- integration, 603
- user manual, 10

**Authentic Desktop perspective in Eclipse, 155****Authentic DesktopCommand,**

in AuthenticDesktopControl, 632

**Authentic DesktopCommands,**

in AuthenticDesktopControl, 634

**Authentic Integration Package, 150, 153****Authentic menu, 217**

dynamic table editing, 42

markup display, 42

**Authentic Plugin for Eclipse,**

installing, 153

**Authentic Plugin for VS .NET,**

installing, 150

**Authentic Scripting,**

security settings, 226

trusted locations, 226

**Authentic View, 64**

adding nodes, 30

applying elements, 30

CDATA sections in, 33

clearing elements, 30

context menu, 27

context menus, 53

data entry devices in, 33

displaying markup tags, 27

document display, 45

editing data in an XML DB, 219

editing DB data in, 218

entering attribute values, 36

entering data in, 33

entities in, 33

entry helpers, 27

entry helpers in, 48

formatting text in, 42

generating output documents from PXF file, 225

inserting entities in, 37

inserting nodes, 30

main window in, 45

markup display in, 42, 45

opening an XML document in, 25

opening new XML file in, 218

overview of GUI, 40

paste as XML/Text, 53

printing an XML document from, 38

removing nodes, 30

special characters in, 33

SPS Tables, 63

switching to, 227

tables (SPS and XML), 63

tables in, 30

toolbar icons, 42

usage of important features, 55

usage of XML tables, 64

XML table icons, 68

XML tables, 64

**Authentic View template, 25****AuthenticDataTransfer,**

dropEffect, 379

getData, 379

ownDrag, 379

type, 379

**AuthenticDesktopControl, 635**

documentation of, 603

examples of integration at document level, 611

integration using C#, 611

object reference, 632

**AuthenticDesktopControlDocument, 643****AuthenticDesktopControlPlaceHolder, 649****AuthenticRange,**

AppendRow, 385

Application, 385

CanPerformAction, 386

CanPerformActionWith, 386

Close, 387

CollapsToBegin, 387

CollapsToEnd, 387

Copy, 387

Cut, 388

Delete, 388

DeleteRow, 388

DuplicateRow, 389

ExpandTo, 390

FirstTextPosition, 390

FirstXMLData, 391

FirstXMLDataOffset, 392

GetElementAttributeNames, 393

GetElementAttributeValue, 393

GetElementHierarchy, 394

GetEntityNames, 394

Goto, 395

GotoNext, 395

GotoNextCursorPosition, 396

GotoPrevious, 397

GotoPreviousCursorPosition, 397

HasElementAttribute, 398

InsertEntity, 398

**AuthenticRange,**

- InsertRow, 399
- IsCopyEnabled, 399
- IsCutEnabled, 400
- IsDeleteEnabled, 400
- IsEmpty, 400
- IsEqual, 400
- IsFirstRow, 401
- IsInDynamicTable, 401
- IsLastRow, 401
- IsPasteEnabled, 401
- IsTextStateApplied, 402
- LastTextPosition, 402
- LastXMLData, 403
- LastXMLDataOffset, 404
- MoveBegin, 405
- MoveEnd, 405
- MoveRowDown, 406
- MoveRowUp, 406
- Parent, 406
- Paste, 406
- PerformAction, 407
- Select, 408
- SelectNext, 408
- SelectPrevious, 409
- SetElementAttributeValue, 410
- SetFromRange, 411
- Text, 411

**AuthenticView, 428**

- Application, 420
- AsXMLString, 420
- DocumentBegin, 422
- DocumentEnd, 422
- Event, 423
- Goto, 424
- IsRedoEnabled, 425
- IsUndoEnabled, 425
- Markup Visibility, 426
- OnBeforeCopy, 413
- OnBeforeCut, 413
- OnBeforeDelete, 414
- OnBeforeDrop, 414
- OnBeforePaste, 415
- OnDragOver, 415
- OnKeyboardEvent, 416
- OnMouseEvent, 417
- OnSelectionChanged, 418
- Parent, 426

- Print, 426
- Redo, 427
- Selection, 427
- Undo, 428
- WholeDocument, 429
- XMLDataRoot, 429

**Auto-hiding windows, 14****Automatic backup settings, 254****Automatic validation, 256**

## B

**Background Information, 654****Big-endian, 258****Browser, 261**

- View, 227

**Browser View,**

- font size, 228
- moving back and forward, 228
- refresh content of, 228
- separate windows, 228
- stop loading page, 228

## C

**C#,**

- integration of Authentic Desktop, 611

**Carriage return key,**

- see Enter key, 85

**CDATA sections,**

- inserting in Authentic View, 58

**Changing view,**

- to Authentic View, 42

**Character-Set,**

- encoding, 258

**Check,**

- spelling checker, 229

**CodeGeneratorDlg,**

- Application, 430
- CPPSettings\_DOMType, 430
- CPPSettings\_LibraryType, 432
- CPPSettings\_UseMFC, 432
- CSharpSettings\_ProjectType, 432
- OutputPath, 433

**CodeGeneratorDlg,**

- OutputPathDialogAction, 433
- OutputResultDialogAction, 433
- Parent, 434
- ProgrammingLanguage, 434
- PropertySheetDialogAction, 434
- TemplateFileName, 435

**COM API,**

- in Scripting Editor, 292

**COM-API,**

- documentation, 325

**Command, 245**

- add to toolbar/menu, 237
- context menu, 245
- delete from menu, 245
- reset menu, 245

**Command line actions, 280****Command reference, 624****Commands,**

- listing in keyboard map, 274

**Configurations,**

- of a global resource, 91

**Configurations in global resources, 105****Configure,**

- XMLSPY UI, 311

**Context menu,**

- commands, 245
- for customization, 250

**Context menus,**

- in Authentic View, 53

**Copy command, 175****Copyright information, 656****CR&LF, 254****Custom dictionary, 229****Customization, 21****Customize, 245**

- context menu, 245
- Customize context menu, 250
- macros, 246
- menu, 245
- toolbar/menu commands, 237

**Cut command, 175****D****DatabaseConnection,**

- ADOConnection, 436
- AsAttributes, 436
- CreateMissingTables, 437
- CreateNew, 437
- DatabaseKind, 437
- ExcludeKeys, 438
- File, 438
- IncludeEmptyElements, 439
- NumberDateTimeFormat, 439
- ODBCConnection, 439
- SQLSelect, 440
- TextFieldLen, 441

**Databases,**

- editing in Authentic View, 218
- see also DB, 71

**Date Picker,**

- using in Authentic View, 78

**Dates,**

- changing manually, 79

**DB, 71, 72**

- creating queries, 72
- editing in Authentic View, 71, 76
- filtering display in Authentic View, 72
- navigating tables in Authentic View, 71
- parameters in DB queries, 72
- queries in Authentic View, 71

**Default,**

- encoding, 258
- menu, 245

**Default editor, 256****Default view,**

- setting in Main Window, 256

**Delete, 237**

- Application.URLDelete, 375
- command from context menu, 245
- command from toolbar, 237
- icon from toolbar, 237
- row (in Authentic View), 224
- shortcut, 242
- toolbar, 239

**Delete command, 175****Dialogs,**

- Application, 442
- CodeGeneratorDlg, 442
- DTDSchemaGeneratorDlg, 443
- FileSelectionDlg, 442
- GenerateSampleXMLDlg, 443
- Parent, 443

**Dialogs,**

SchemaDocumentationDlg, 443

**Dictionary,**

adding custom, 229  
modifying existing, 229  
spelling checker, 229

**directories,**

creating with Application.URLMakeDirectory, 375

**Distribution,**

of Altova's software products, 656

**Docking windows, 14****Document, 457**

Application, 450  
AssignDTD, 450  
AssignSchema, 450  
AssignXSL, 450  
AssignXSLFO, 451  
AuthenticView, 451  
Close, 452  
ConvertDTDOrSchema, 452  
CreateChild, 454  
CreateSchemaDiagram, 455  
CurrentViewMode, 455  
DataRoot, 456  
DocEditView, 456  
Encoding, 456  
EndChanges, 457  
ExecuteXQuery, 457  
ExportToDatabase, 458  
ExportToText, 459  
FullName, 460  
GenerateDTDOrSchema, 460, 461  
GenerateProgramCode, 461  
GenerateSampleXML, 462  
GenerateSchemaDocumentation, 462  
GetExportElementList, 464  
GetPathName, 465  
GridView, 465  
IsModified, 466  
IsValid, 466  
IsWellFormed, 468  
Name, 469  
OnBeforeCloseDocument, 448  
OnBeforeSaveDocument, 447  
OnBeforeValidate, 448  
OnCloseDocument, 449  
OnViewActivation, 449  
Path, 469

RootElement, 469

Save, 470

SaveAs, 470

Saved, 470

SaveInString, 471

SaveToURL, 471

SetActiveDocument, 472

SetEncoding, 472

SetExternalsValid, 472

SetPathName, 473

Spelling checker, 229

StartChanges, 473

SwitchViewMode, 474

Title, 474

TransformXSL, 475

TransformXSLFO, 475

UpdateViews, 476

UpdateXMLData, 476

XQuery, 457

**Document-level,**

examples of integration of <%SPY-GEN%>, 611

**Documents,**

Count, 478

Item, 478

NewAuthenticFile, 478

NewFile, 479

NewFileFromText, 479

OpenAuthenticFile, 479

OpenFile, 480

OpenURL, 480

OpenURLDialog, 481

**Documents in Main Window, 15****DTDs, 254, 256****DTDSchemaGeneratorDlg,**

Application, 482  
AttributeTypeDefinition, 482  
DTDSchemaFormat, 482  
FrequentElements, 483  
GlobalAttributes, 483  
MaxEnumLength, 483  
MergeAllEqualNamed, 483  
OnlyStringEnums, 484  
OutputPath, 484  
OutputPathDialogAction, 484  
Parent, 485  
ResolveEntities, 485  
TypeDetection, 485  
ValueList, 485

**Duplicate,**

row (in Authentic View), 224

**Dynamic (SPS) tables in Authentic View,**

usage of, 63

**Dynamic tables,**

editing, 42

**E****Eclipse platform,**

and Authentic Desktop, 152

and Authentic Integration Package, 153

Authentic Desktop entry points in, 158

Authentic Desktop Perspective in, 155

**Edit,**

macro button, 250

**Edit menu, 175****Edited with XMLSPY, 254****ElementList,**

Count, 486

Item, 486

RemoveElement, 486

**ElementListItem,**

ElementKind, 487

FieldCount, 487

Name, 487

RecordCount, 487

**Elements entry helper,**

in Authentic View, 48

**E-mail,**

sending files with, 172

**Empty elements, 256****Encoding,**

default, 258

of files, 166

**End User License Agreement, 656****Enter key,**

effects of using, 85

**Entities,**

defining in Authentic View, 58, 81

inserting in Authentic View, 37, 58

**Entities entry helper,**

in Authentic View, 48

**Entry Helpers, 19**

display of, 272

**Enumerations,**

in AuthenticDesktopControl, 652

SPYAttributeTypeDefinition, 589

SPYAuthenticActions, 589

SPYAuthenticDocumentPosition, 589

SpyAuthenticElementActions, 589

SPYAuthenticElementKind, 590

SPYAuthenticMarkupVisibility, 590

SPYDatabaseKind, 591

SPYDialogAction, 591

SPYDOMType, 591

SPYDTDSchemaFormat, 592

SPYEncodingByteOrder, 592

SPYExportNamespace, 592

SPYFrequentElements, 592

SPYKeyEvent, 593

SPYLibType, 594

SPYLoading, 594

SPYMouseEvent, 594

SPYNumberDateTimeFormat, 595

SPYProgrammingLanguage, 595

SPYProjectItemTypes, 595

SPYProjectType, 596

SPYSampleXMLGenerationOptimization, 596

SPYSampleXMLGenerationSchemaOrDTDAssignment, 597

SPYSchemaDefKind, 597

SPYSchemaDocumentationFormat, 598

SPYTextDelimiters, 598

SPYTextEnclosing, 599

SPYTypeDetection, 599

SPYURLTapes, 599

SPYViewModes, 600

SPYVirtualKeyMask, 601

SPYXMLDataKind, 601

**Evaluation key for Altova software, 275****Evaluation period,**

of Altova's software products, 656

**Event, 357, 358, 359, 413, 414, 415, 416, 417, 418, 447, 448, 449, 510, 511, 512****Events, 332****Exit mode, 254****Explorer, 256****ExportSettings,**

CreateKeys, 488

ElementList, 488

EntitiesToText, 488

ExportAllElements, 489

FromAttributes, 489



**ExportSettings,**

- FromSingleSubElements, 489
- FromTextValues, 489
- IndependentPrimaryKey, 489
- Namespace, 490
- SubLevelLimit, 490

**External applications,**

- opening files in, 241

**External parsed entites, 256****External XSL processor, 261**

## F

**File, 254**

- closing, 167
- creating new, 160
- default encoding, 258
- encoding, 166
- opening, 161
- opening options, 254
- printing options, 173
- saving, 167
- sending by e-mail, 172
- tab, 254

**File menu, 160****File types, 256****Files,**

- adding to source control, 188
- most recently used, 174

**FileSelectionDlg,**

- Application, 491
- DialogAction, 491
- FullName, 491
- Parent, 492

**Find and replace text in document, 177****Find text in document, 176****Floating windows, 14**

## G

**Generate Sample XML, 589, 596, 597****GenerateSampleXMLDlg,**

- Application, 505
- FillWithSampleData, 506

NonMandatoryAttributes, 507

NonMandatoryElements, 507

Parent, 508

RepeatCount, 509

TakeFirstChoice, 509

**Global,**

- settings, 253

**Global resources, 90**

- active configuration for, 236
- changing configurations, 105
- defining, 91, 235
- defining database-type, 100
- defining file-type, 93
- defining folder-type, 98
- toolbar activation, 239
- using, 102, 105
- using file-type and folder-type, 102

**Global Resources XML File, 91****Grammar, 256****Graphics formats,**

- in Authentic View, 84

**GridView,**

- CurrentFocus, 513
- Deselect, 513
- IsVisible, 513
- OnBeforeDrag, 510
- OnBeforeDrop, 511
- OnBeforeStartEditing, 511
- OnEditingFinished, 512
- OnFocusChanged, 512
- Select, 513
- SetFocus, 513

**GUI description, 14**

## H

**Help menu, 274****Hide markup, 42, 45****Hide markup (in Authentic View), 224****Hotkey, 242****HTML output,**

- generating in Authentic View from PXF file, 225

**I****Icon,**

- add to toolbar/menu, 237
- show large, 249

**Image formats,**

- in Authentic View, 84

**Info Window, 19**

- display of, 272

**Insert,**

- row (in Authentic View), 224

**Integrating,**

- Authentic Desktop in applications, 603

**Internet, 279****Internet usage,**

- in Altova products, 655

**J****Java, 615****Java settings, 265****Java virtual machine,**

- path setting, 265

**JScript,**

- scripting with Authentic Desktop, 283

**K****Keyboard map, 274****Keyboard shortcut, 242****Key-codes for Altova software, 275****L****Large markup (in Authentic View), 224****Legal information, 656****License,**

- information about, 656

**License metering,**

- in Altova products, 657

**Licenses for Altova software, 275****Line-breaks, 254****Links,**

- following in Authentic View, 58

**Little-endian, 258****loading, 480****M****Macro,**

- add to menu/toolbar, 246
- edit button, 250

**Macros,**

- developing, 283, 288
- enabling, 295, 307
- running, 308
- running application macros, 235

**Main Window, 15****Markup,**

- in Authentic View, 42, 45

**Markup (in Authentic View),**

- collapse/expand, 225
- hide, 224
- show small/large/mixed, 224

**Maximum cell width, 261****Memory requirements, 654****Menu, 245**

- add macro to, 246
- add/delete command, 237
- Authentic, 217
- customize, 245
- Default/XMLSPY, 245
- delete commands from, 245
- Edit, 175
- Help, 274
- Project, 178
- Tools, 229
- View, 227
- XML, 208
- XSL/XQuery, 210

**Menu Bar, 20****Menu commands, 159****Messages Window, 19**

- display of, 272

**Microsoft® SharePoint® Server, 200****MIME, 256**

**Mixed markup (in Authentic View), 224**

**Mostly recently used files,**

list of, 174

**Move row in Authentic View, 225**

**MSXML, 261**

**Multi-user, 254**

## N

**Network Proxy, 269**

**Network settings, 268**

**New file,**

creating, 160

**Non-XML files, 256**

## O

**Online Help, 271, 274**

**Open,**

file, 161

**OpenAI API key, 271**

**Opening options,**

file, 254

**Optimal Widths, 261**

**Ordering Altova software, 275**

**OS,**

for Altova products, 654

**Output formatting, 254**

**Output Windows,**

display of, 272

**Overview,**

of XMLSpy API, 326

## P

**Parameters,**

in DB queries, 72

passing to stylesheet via interface, 212

**Parent, 469**

**Parser,**

built into Altova products, 654

XSLT, 261

**Paste,**

as Text, 58

as XML, 58

**Paste As,**

Text, 53

XML, 53

**Paste command, 175**

**PDF Help, 271, 274**

**PDF output,**

generating in Authentic View from PXF file, 225

**Platforms,**

for Altova products, 654

**Plug-in,**

ATL sample files, 314

registration, 310

User interface configuration, 311

XMLSPY, 310

**Presentation, 261**

**Print preview and setup, 173**

**Printing,**

from Authentic View, 38

**Printing options, 173**

**Program settings, 253**

**Programmers' Reference, 281**

**Project,**

properties, 204

**Project menu, 178**

**Project Window, 17**

display of, 272

**Projects, 197**

adding active files to, 197

adding external folders to, 197

adding external Web folders to, 200

adding files to, 196

adding folders to, 197

adding global resources to, 196

adding related files to, 197

adding to source control, 188

adding URL to, 196

closing, 181

creating new, 181

most recently used, 207

opening, 181

overview, 178

reloading, 181

saving, 182

**Proxy settings, 269**

**PXF file,**

generating output documents from Authentic View, 225

## Q

### Queries,

for DB display in Authentic View, 72

## R

### Redo command, 175

### Register,

plug-in, 310

### Registering your Altova software, 275

### Registry,

settings, 253

### Reload, 254

### Reloading,

changed files, 166

### Replace text in document, 177

### Reset,

menu commands, 245

shortcut, 242

toolbar & menu commands, 239

### Return key,

see Enter key, 85

### RichEdit, 224

### Row,

append (in Authentic View), 224

delete, 225

delete (in Authentic View), 224

duplicate (in Authentic View), 224

insert (in Authentic View), 224

move up/down, 225

### RTF output,

generating in Authentic View from PXF file, 225

## S

### save, 471

### Saving files,

encoding of, 166

### schema, 368

settings, 254

### Schema Manager, 237

CLI Help command, 143

CLI Info command, 144

CLI Initialize command, 144

CLI Install command, 145

CLI List command, 145

CLI overview, 143

CLI Reset command, 146

CLI Uninstall command, 147

CLI Update command, 148

CLI Upgrade command, 148

how to run, 135

installing a schema, 140

listing schemas by status in, 138

overview of, 131

patching a schema, 140

resetting, 142

status of schemas in, 138

uninstalling a schema, 142

upgrading a schema, 140

### SchemaDocumentationDlg,

AllDetails, 515

Application, 515

IncludeAll, 517

IncludeAttributeGroups, 517

IncludeComplexTypes, 517

IncludeGlobalElements, 518

IncludeGroups, 518

IncludeIndex, 518

IncludeLocalElements, 519

IncludeRedefines, 519

IncludeSimpleTypes, 520

OptionsDialogAction, 520

OutputFile, 521

OutputFileDialogAction, 521

OutputFormat, 521

Parent, 522

ShowAnnotations, 522

ShowAttributes, 522

ShowChildren, 523

ShowConstraints, 524

ShowDiagram, 523

ShowEnumerations, 523

ShowNamespace, 524

ShowPatterns, 524

ShowProgressBar, 525

ShowProperties, 525

ShowResult, 525

ShowSingleFacets, 525

**SchemaDocumentationDlg,**

- ShowSourceCode, 526
- ShowType, 526
- ShowUsedBy, 526

**Script language, 265****Scripting, 265****Scripting Editor,**

- overview, 283, 285
- starting, 234

**Search,**

- see Find, 177

**Select All command, 176****Sending files by email, 172****Settings, 21, 253**

- scripting, 265

**SharePoint@ Server, 200****Shortcut, 242**

- assigning/deleting, 242
- show in tooltip, 249

**Show large markup, 42, 45****Show mixed markup, 42, 45****Show small markup, 45****Show small markup, 42****Side-by-side, 261****Small markup (in Authentic View), 224****Source control, 266**

- add to source control, 188
- changing provider, 195
- checking out, 186
- enabling, disabling, 184
- get latest version, 184
- getting files, 184
- installing a source-control plug-in, 106
- open project, 183
- properties, 194
- refresh status, 195
- removing from, 189
- sharing from, 189
- show differences, 192
- show history, 191
- supported providers, 182
- undo check out, 188

**Source control manager, 195****Spelling checker,**

- custom dictionary, 229

**Spelling options, 232****Splash screen, 261****SPP file locations, 178****SPS,**

- assigning to new XML file, 160

**SPS tables,**

- editing dynamic tables, 42

**SPS tables in Authentic View,**

- usage of, 63

**SpyProject,**

- CloseProject, 528
- ProjectFile, 528
- RootItems, 528
- SaveProject, 528
- SaveProjectAs, 528

**SpyProjectItem,**

- ChildItems, 529
- FileExtensions, 529
- ItemType, 529
- Name, 530
- Open, 530
- ParentItem, 530
- Path, 530
- ValidateWith, 530
- XMLForXSLTransformation, 530
- XSLForXMLTransformation, 531
- XSLTransformationFileExtension, 531
- XSLTransformationFolder, 531

**SpyProjectItems,**

- AddFile, 531
- AddFolder, 532
- AddURL, 532
- Count, 532
- Item, 533
- RemoveItem, 533

**Start group,**

- add (context menu), 250

**Static (SPS) tables in Authentic View,**

- usage of, 63

**Status Bar, 20****Support Center, 279****Support options, 22****Syntax-coloring, 256, 261****T****Tab characters, 254****Table,**

- build automatically, 256

**Tables,**

- editing dynamic (SPS) tables, 42
- in Authentic View, 30

**Tables in Authentic View,**

- icons for editing XML tables, 68
- usage of, 63
- using SPS (static and dynamic) tables, 63
- using XML tables, 64

**Technical Information, 654****Technical Support, 279****Template files,**

- for new documents, 160

**Template XML File,**

- in Authentic View, 25

**Templates,**

- of XML documents in Authentic View, 218

**terminate, 372****Text,**

- editing in Authentic View, 58
- find and replace, 177
- finding in document, 176
- formatting in Authentic View, 58

**TextImportExportSettings,**

- DestinationFolder, 534
- EnclosingCharacter, 534
- Encoding, 534
- EncodingByteOrder, 534
- FieldDelimiter, 534
- FileExtension, 534
- HeaderRow, 535
- ImportFile, 535

**Toolbar, 20, 239**

- activate/deactivate, 239
- add command to, 237
- add macro to, 246
- create new, 239
- reset toolbar & menu commands, 239
- show large icons, 249

**Tools,**

- see also External applications, 241

**Tools menu, 229****Tooltip, 249**

- show, 249
- show shortcuts in, 249

**Transformation,**

- see XSLT transformation, 211

**Trusted locations for Authentic scripts, 226****Turn off automatic validation, 256****U****UCS-2, 258****Undo command, 175****Unicode support,**

- in Altova products, 655

**Unsaved changes, 254****URL, 375, 471, 480, 481**

- sending by e-mail, 172

**User interface,**

- configure using plug-in, 311

**User interface description, 14****User manual, 10, 271, 274****User Manual of Authentic Desktop, 12****UTF-16, 258****V****Validate on modification, 209****Validating XML documents, 208****Validation, 21****Validation messages, 19****Validation settings, 254****Validator,**

- in Altova products, 654

**VBScript,**

- scripting with Authentic Desktop, 283

**View,**

- Browser vView, 227

**View menu, 227****Visual Studio,**

- adding the Authentic Desktop ActiveX Controls to the toolbox, 604

**Visual Studio .Net,**

- and Authentic Desktop, 149
- and Authentic Desktop differences, 151

**VS .NET,**

- and Authentic Integration Package, 150

**W****Watch for changes, 254**

**Well-formedness check, 208****Window menu, 272****Windows,**

- arranging, 272
- auto-hiding, 14
- cascading, 272
- floating, docking, tabbing, 14
- managing display of, 14
- support for Altova products, 654
- tiling, 272
- turning display on/off, 272

**Word 2007+ output,**

- generating in Authentic View from PXF file, 225

## X

**XML,**

- spelling checker, 229

**XML DB,**

- loading new data row into Authentic View, 219
- loading new XML data row, 71

**XML document,**

- opening in Authentic View, 25

**XML menu, 208****XML Parser,**

- about, 654

**XML Schema Manager, 237****XML Signature, 220****XML signatures, 83****XML tables in Authentic View,**

- icons for editing, 68
- usage of, 64

**XML-Conformance, 256****XMLData,**

- AppendChild, 577
- EraseAllChildren, 579
- EraseCurrentChild, 579
- GetChild, 580
- GetChildKind, 581
- GetCurrentChild, 582
- GetFirstChild, 582
- GetNextChild, 583
- HasChildren, 584
- HasChildrenKind, 585
- InsertChild, 585
- IsSameNode, 586

Kind, 586

MayHaveChildren, 587

Name, 587

Parent, 587

TextValue, 588

**XMLSPY,**

- plug-in registration, 310

**XMLSpy API,**

- documentation, 325
- overview, 326

**XMLSPY plug-in, 310****XMLSpyLib, 325, 326**

- Application, 356
- AuthenticDataTransfer, 378
- AuthenticRange, 383
- AuthenticView, 412
- CodeGeneratorDlg, 429
- DatabaseConnection, 435
- Dialogs, 441
- Document, 445
- Documents, 477
- DTDSchemaGeneratorDlg, 481
- ElementList, 486
- ElementListItem, 487
- ExportSettings, 488
- FileSelectionDlg, 490
- GenerateSampleXMLDlg, 504
- GridView, 510
- ProjectItem, 529
- SchemaDocumentationDlg, 513
- SpyProject, 527
- SpyProjectItems, 531
- TextImportExportSettings, 533
- XMLData, 576

**XPath to selected node, 40****XQuery,**

- passing variables to the XQuery document, 212

**XSL/XQuery menu, 210****XSLT,**

- processor, 261

**XSLT parameters,**

- passing to stylesheet via interface, 212

**XSLT transformation, 210, 211**

- to FO, 211
- to PDF, 211