User and Reference Manual

Altova Authentic 2012 Desktop Edition User & Reference Manual

Published: 2012

Table of Contents

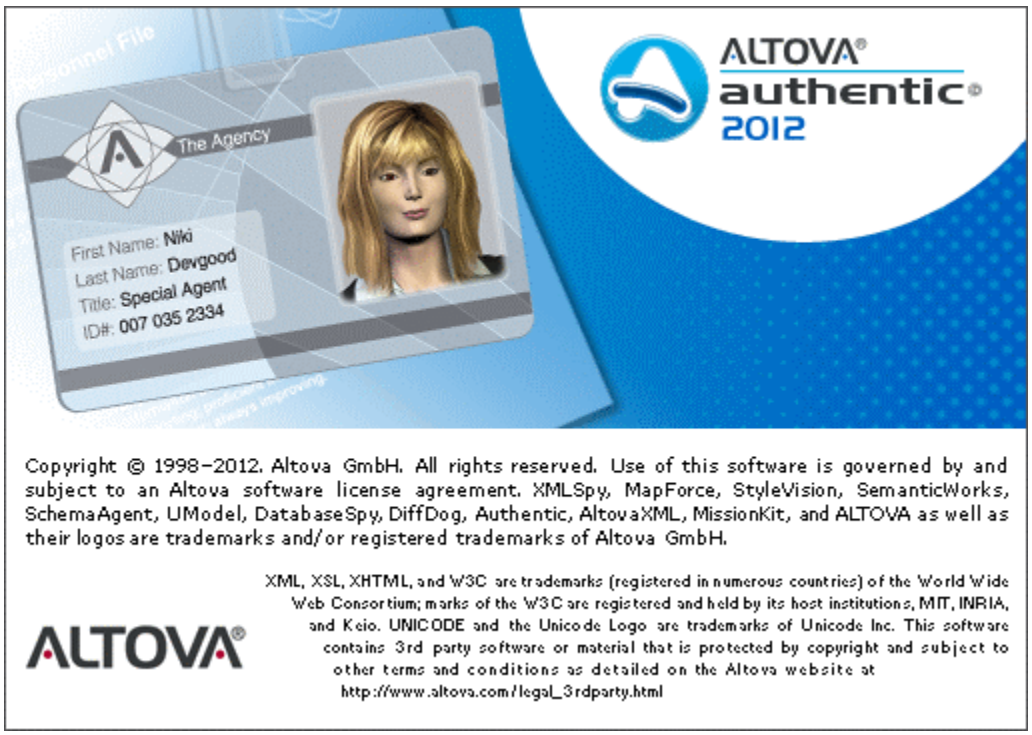
Index

Chapter 1

Welcome to Authentic Desktop

Welcome to Authentic Desktop

Altova® Authentic® 2012 Desktop Enterprise Edition is an innovative visual approach to authoring XML documents that shields the end-user from having to deal with the technical aspects of XML. The Enterprise Edition is available for 64-bit and 32-bit machines.



Chapter 2

User Manual

User Manual

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

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

File paths in Windows XP, Windows Vista, and Windows 7
File paths given in this documentation will not be the same for all operating systems. You should note the following correspondences:

- *(My) Documents folder:* The My Documents folder of Windows XP is the Documents folder of Windows Vista and Windows 7. It is located by default at the following respective locations. Example files are usually located in a sub-folder of the (My) Documents folder.

Windows XP	`C:/Documents and Settings/<username>/My Documents`
Windows Vista, Windows 7	`C:/Users/<username>/Documents`

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

Windows XP	`C:/Program Files/Altova`
Windows Vista, Windows 7	`C:/Program Files/Altova`
32-bit package on 64-bit Windows OS (XP, Vista, 7)	`C:/Program Files (x86)/Altova`

1 Introduction

This introduction describes:

- The application GUI, and
- The application environment.

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

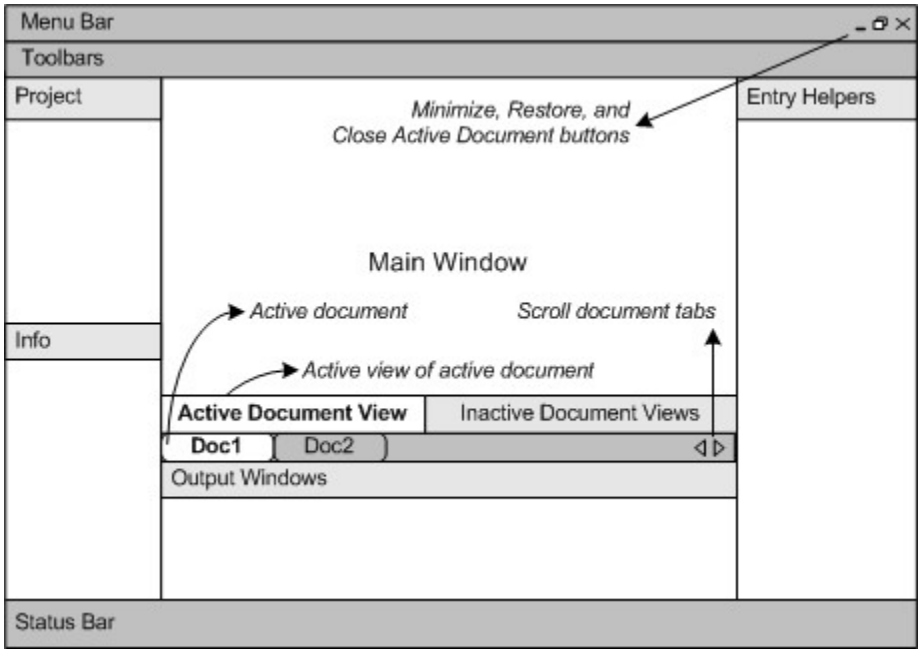
The Application Environment section points out the various settings that control how files are displayed and can be edited. It also explains how and where you can customize your application. In this section, you will learn where important example and tutorial files have been installed on your machine,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

1.1 The Graphical User Interface (GUI)

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

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

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



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.

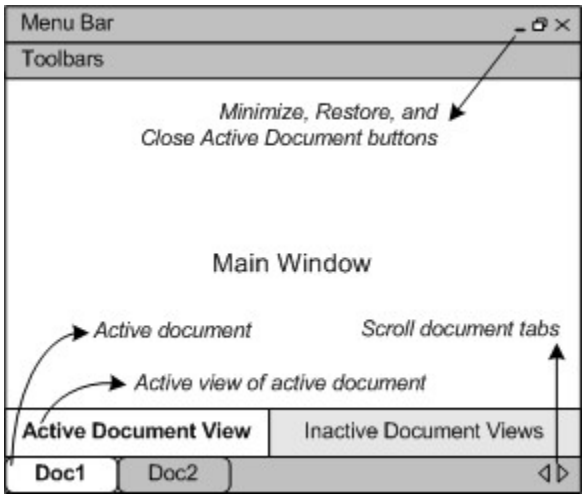
To float a tabbed window, double-click its tab. To drag a tabbed window out of a group of tabbed windows, drag its tab.

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**).

1.1.1 Main Window

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



Files in the Main Window

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

Views in the Main Window

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

The available views are either editing or browser views:

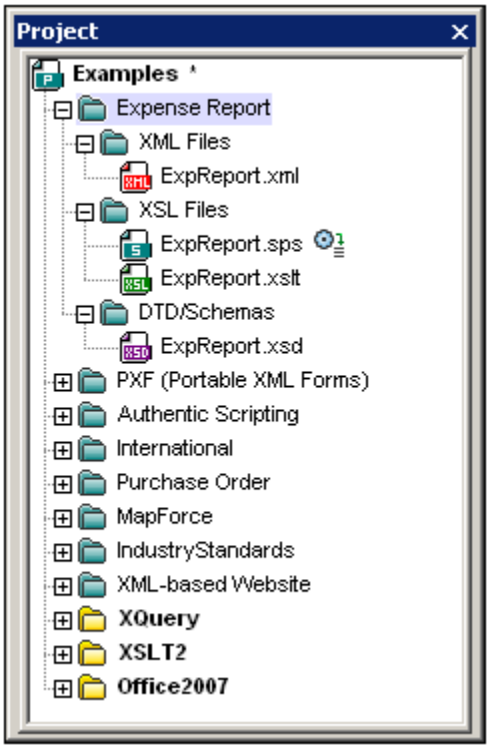
- **Authentic View:** For editing XML documents that are based on StyleVision Power Stylesheets
- **Browser View:** An integrated browser view that supports both CSS and XSL stylesheets.

Note: The default view for individual file extensions can be customized in the **Tools | Options**

dialog: in the Default View pane of the File Types tab.

1.1.2 Project Window

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



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

Project operations

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

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

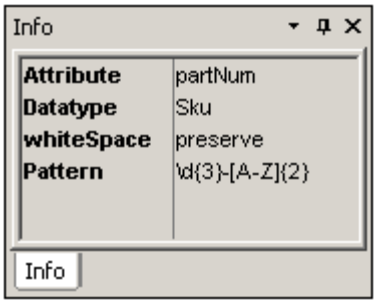
extensions for each folder (in the Properties dialog of that folder) to keep common files in one convenient place. When a file is added to the parent folder, it is automatically added to the sub-folder that has been defined to contain files of that file extension.

- In the Project Window, a folder can be dragged to another folder or to another location within the same folder, while a file can be dragged to another folder but cannot be moved within the same folder (within which files are arranged alphabetically). Additionally, files and folders can be dragged from Windows File Explorer to the Project Window.
- Each folder has a set of properties that are defined in the Properties dialog of that folder. These properties include file extensions for the folder, the schema by which to validate XML files, the XSLT file with which to transform XML files, etc.
- Batch processing of files in a folder is done by right-clicking the folder and selecting the relevant command from the context menu.

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

1.1.3 Info Window

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



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

1.1.4 Entry Helpers

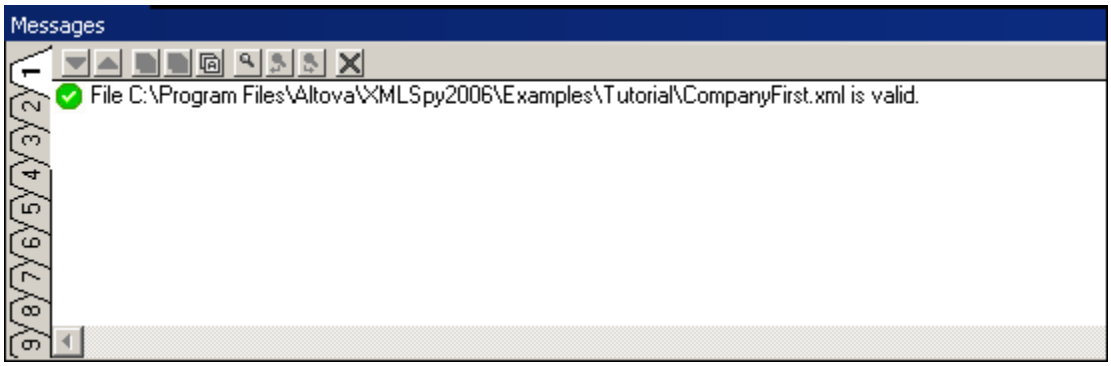
Entry helpers are an intelligent editing feature that helps you to create valid XML documents quickly. When you are editing a document, the entry helpers display structural editing options according to the current location of the cursor. The entry helpers get the required information from the underlying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

Note the following:

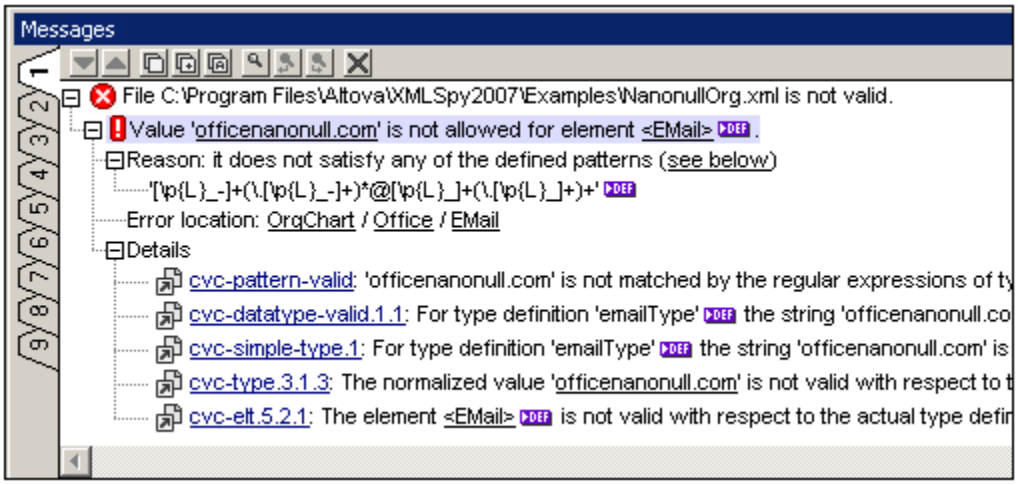
- You can turn the display of entry helpers on or off with the menu option **Window | Entry Helpers**.

1.1.5 Output Window: Messages

The Messages Window displays messages about actions carried out in Authentic Desktop as well as errors and other output. For example, if an XML, XML Schema, DTD, or XQuery document is validated and is valid, a successful validation message (*screenshot below*) is displayed in the Messages Window:



Otherwise, a message that describes the error (*screenshot below*) 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. Clicking the purple `Def` buttons, opens the relevant schema definition in Schema View.



The Messages Window is enabled in all views, but clicking a link to content in an XML document highlights that node in the XML document in Text View.

Note: 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** or **F8**. Invalid files in a project will be opened and made active in the Main Window, and the File Is Invalid error message will be displayed.

1.1.6 Menu Bar, Toolbars, Status Bar

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

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

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

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

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

1.2 The Application Environment

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

This section is organized as follows:

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

1.2.1 Settings and Customization

In Authentic Desktop, there are several settings and customization options that you can select. In this section, we point you to these options. This section is organized into the following parts.

- Settings
- Customization

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

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

The various customization options are described in the **User Reference** section.

1.2.2 Tutorials, Projects, Examples

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

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

```
C:\Documents and Settings\<username>\My Documents\
Altova\Authentic2012\AuthenticExamples\
```

The `My Documents\Altova\Authentic2012` 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 folder in a separate working area.

Note about the master Authentic Desktop folder
When Authentic Desktop is installed on a machine, a master `Altova\Authentic2012` folder is created at the following folder location:

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

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

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

1.2.3 Authentic Desktop Features and Help, and Altova Products

The Altova website, www.altova.com, 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.

2 Authentic View Tutorial

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

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

This tutorial takes you through the following steps:

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

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

Tutorial requirements
All **the files** you need for the tutorial are in the `C:\Documents and Settings\<username>\My Documents\Altova\<ProductName>\Examples` folder of your Altova application folder. These files are:

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

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

Caution: We recommend that you use a copy of `NanonullOrg.xml` for the tutorial, so that you can always retrieve the original should the need arise.

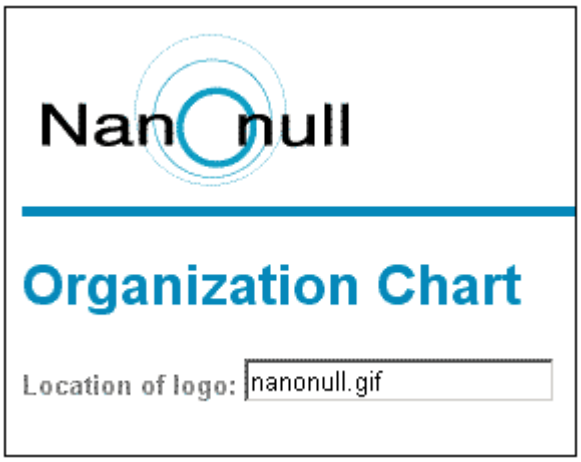
2.1 Opening an XML Document in Authentic View

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

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

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

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



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

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

Via File | New

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

Via Authentic | New Document

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

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

2.2 The Authentic View Interface

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

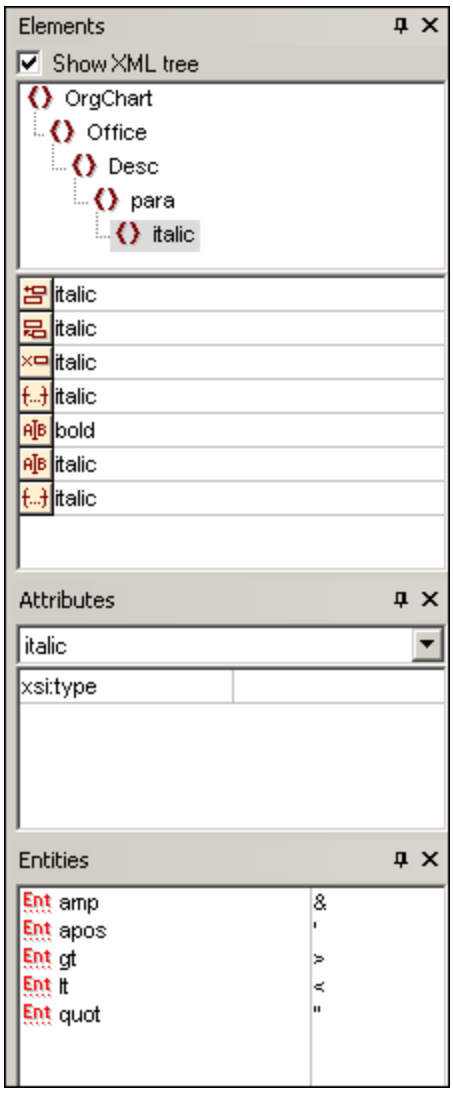
Displaying XML node tags (document markup)
An XML document is essentially a hierarchy of nodes. For example:

```
<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 Atributes entry helper. Selecting an element from the dropdown list of the combo box causes that element's attributes to be displayed in the entry helper, where they can then be edited.

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

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

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

Note: For more details about the interface, see Authentic View Interface

2.3 Node Operations

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

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

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

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

Inserting elements
Elements can be inserted at the following locations:

- The cursor location within an element node. The elements available for insertion at that location are listed in a submenu of the context menu's **Insert** command. In the Elements entry helper, elements that can be inserted at a location are indicated with the  icon. In the `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 AB 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 Using tables in Authentic View section.

2.4 Entering Data in Authentic View

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

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

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

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

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

- *Special characters that are used for XML markup* (ampersand, apostrophe, greater than, less than, and quotes). These characters are available as built-in entities and can be entered in the document by double-clicking the respective entity in the Entities entry helper. If these characters occur frequently (for example, in program code listings), then they can be entered within CDATA sections. To insert a CDATA section, right-click at the location where you wish to enter the CDATA section, and select **Insert CDATA Section** from the context menu. The XML processor ignores all markup characters within CDATA sections. This also means that if you want a special character inside a CDATA section, you should enter that character and not its entity reference.
- *Special characters that cannot be entered via the keyboard* should be entered by copying them from the character map of your system to the required location in the document.
- *A frequently used text string* can be defined as an entity, which appears in the Entities entry helper. The entity is inserted at the required locations by placing the cursor at each required location and double-clicking the entity in the entry helper. This is useful for maintenance because the value of the text string is held in one location; if the value needs to be changed, then all that needs to be done is to change the entity definition.

Note: When markup is hidden in Authentic View, an empty element can easily be overlooked. To make sure that you are not overlooking an empty element, switch large or small markup on.

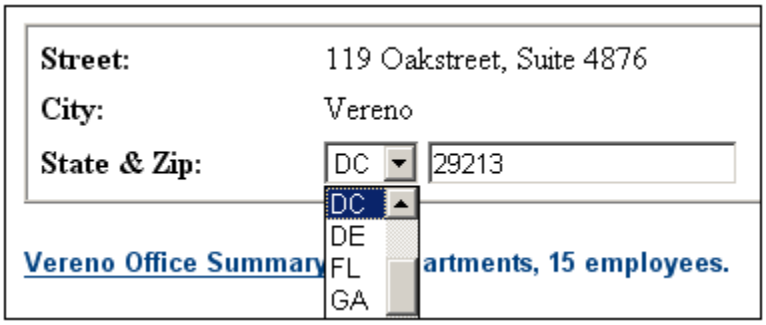
Try using each type of text content described above.

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.



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.

2.5 Entering Attribute Values

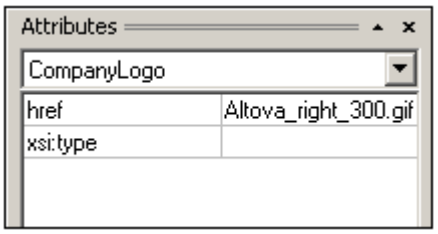
An attribute is a property of an element, and an element can have any number of attributes. Attributes have values. You may sometimes be required to enter XML data as an attribute value. In Authentic View, you enter attribute values in two ways:

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

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

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

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



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

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

2.6 Adding Entities

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

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

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

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

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

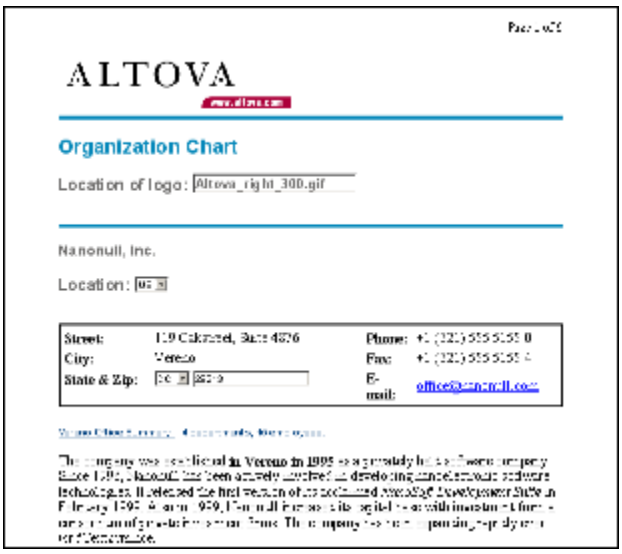
Defining your own entities
As a document editor, you can define your own document entities. How to do this is described in the section Defining Entities in Authentic View.

2.7 Printing the Document

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

To print `NanonullOrg.xml`, do the following:

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



   Notice that the formatting of the page is the same as that in Authentic View.
3. To print the file, click **File | Print**.

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

3 Authentic View

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:

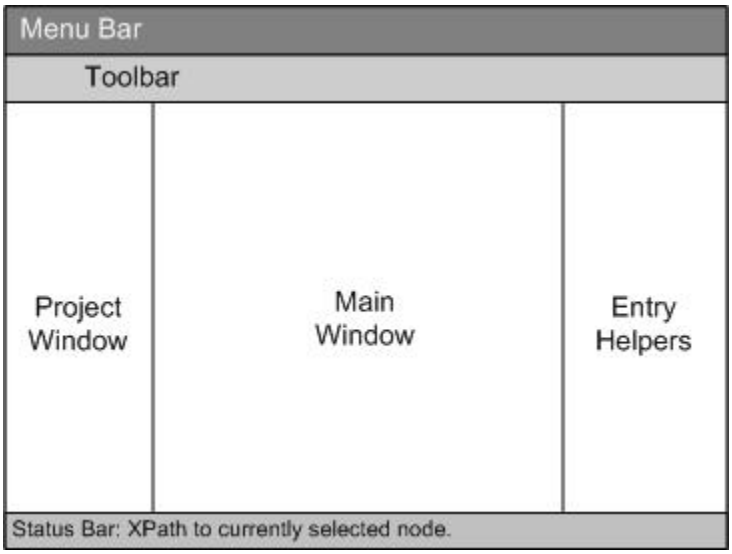
- An overview of the interface
- A description of the toolbar icons specific to Authentic View
- A description of viewing modes available in the main Authentic View window
- A description of the Entry Helpers and how they are to be used
- A description of the context menus available at various points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

3.1 Overview of the GUI

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



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

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

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

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

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

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

Context menus
These are the menus that appear when you right-click in the Main Window. The available

commands are context-sensitive editing commands, i.e. they allow you to manipulate structure and content relevant to the selected node. Such manipulations include inserting, appending, or deleting a node, adding entities, or cutting and pasting content.

3.2 Authentic View Toolbar Icons

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

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

Hide markup. All XML tags are hidden except those which have been collapsed. Double-clicking on a collapsed tag (which is the usual way to expand it) in Hide markup mode will cause the node's content to be displayed and the tags to be hidden.

Show small markup. XML element/attribute tags are shown without names.

Show large markup. XML element/attribute tags are shown with names.

Show mixed markup. In the StyleVision Power Stylesheet, each XML element or attribute can be specified to display (as either large or small markup), or not to display at all. This is called mixed markup mode since some elements can be specified to be displayed with markup and some without markup. In mixed markup mode, therefore, the Authentic View user sees a customized markup. Note, however, that this customization is created by the person who has designed the StyleVision Power Stylesheet. It cannot be defined by the Authentic View user.

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

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

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

Append row to table

 Insert row in table

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

 Move current row up by one row

 Move current row down by one row

 Delete the current row

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

Creating and editing XML tables

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

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

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

Text formatting icons

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

DB Row Navigation icons

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

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

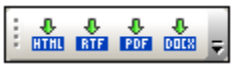
XML database editing

The **Select New Row with XML Data for Editing** command enables you to select a new row

from the relevant table in an XML DB, such as IBM DB2. This row appears in Authentic View, can be edited there, and then saved back to the DB.

Portable XML Form (PXF) toolbar buttons

The following PXF toolbar buttons are available in the Authentic View of XMLSpy and Authentic Desktop:



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

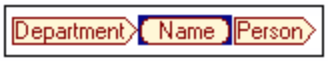
These buttons are enabled when a PXF file is opened in Authentic View. Individual buttons are enabled if the PXF file was configured to contain the XSLT stylesheet for that specific output format. For example, if the PXF file was configured to contain the XSLT stylesheets for HTML and RTF, then only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

3.3 Authentic View Main Window

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

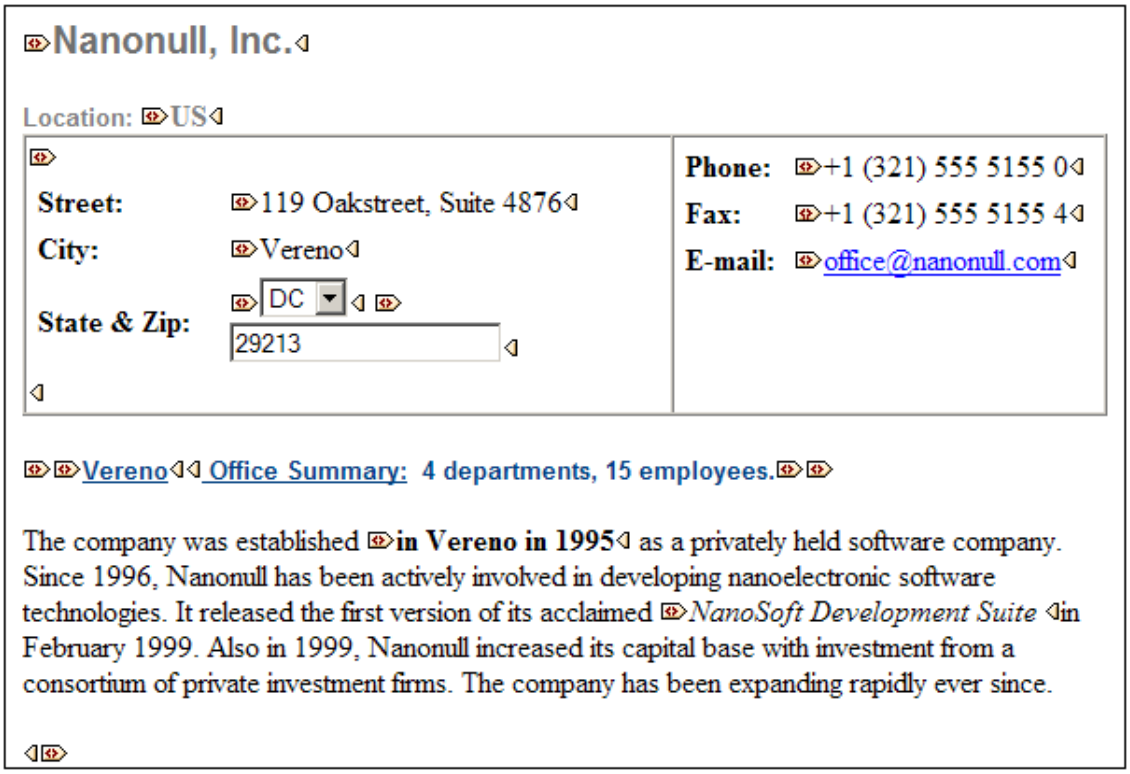
Large markup
This shows the start and end tags of elements and attributes with the element/attribute names in the tags:

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:



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

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

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

Hide all markup
All XML markup is hidden. Since the formatting seen in Authentic View is the formatting of the printed document, this viewing mode is a WYSIWYG view of the document.

Content display
In Authentic View, content is displayed in two ways:

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

- 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.

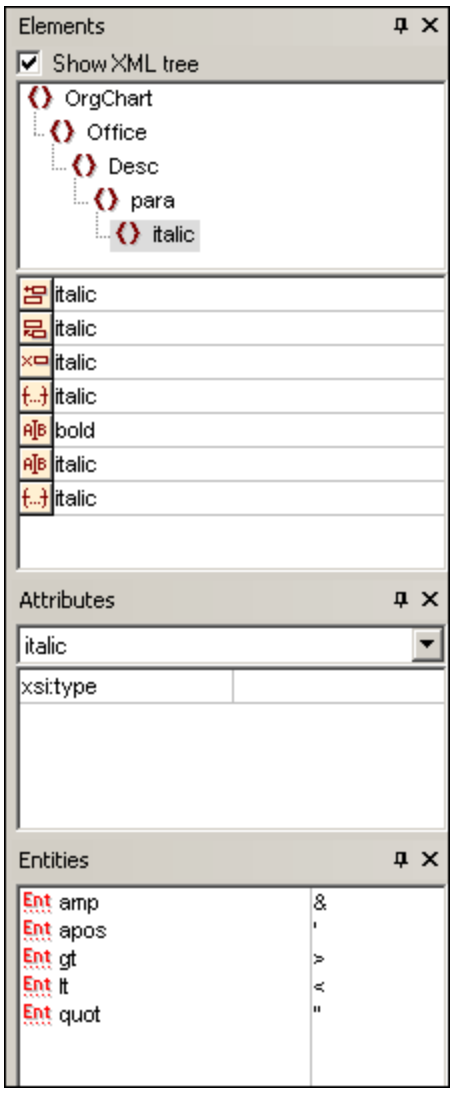
Optional nodes

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.

3.4 Authentic View Entry Helpers

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



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

Each of the entry helpers is described separately below.

Elements Entry Helper

The Elements Entry Helper consists of two parts:

- The upper part, containing an XML tree that can be toggled on and off using the **Show XML tree** check box. The XML tree shows the ancestors up to the document's root element for the current element. When you click on an element in the XML tree,

elements corresponding to that element (as described in the next item in this list) appear in the lower part of the Elements Entry Helper.

- The lower part, containing a list of the nodes that can be inserted within, before, and after; 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 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 Append After Element command, the element is inserted at the correct hierarchical level.

Remove Element
Removes the element and its content.

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

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

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

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

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

element of mixed content; the text range will be created as content of the applied element.

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

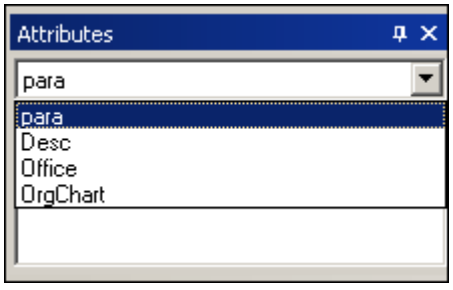
If a text range is selected rather than an element, applying an element to the selection will create the applied element at that location with the selected text range as its content. Applying an element when the cursor is an insertion point is not allowed.

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

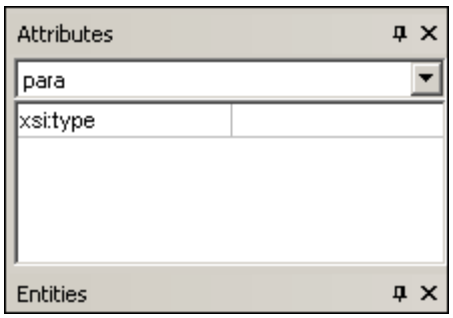
Clear Element (when insertion point selected)
This icon appears when the cursor is placed within an element that is a child of a mixed-content element. Clicking the icon clears the inline element.

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

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



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



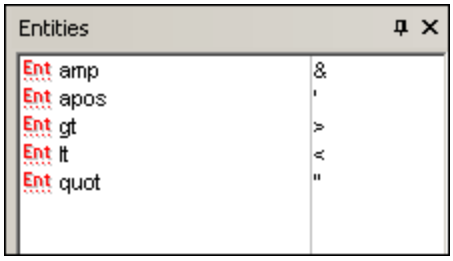
To enter a value for an attribute, click in the value field of the attribute and enter the value. This creates the attribute and its value in the XML document.

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

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

Entities Entry Helper

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



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

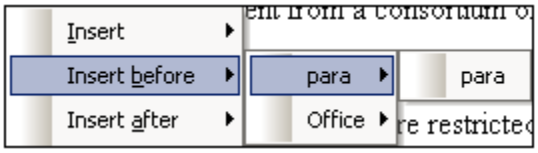
You can also **define your own entities** in Authentic View and these will also be displayed in the entry helper: see Define Entities in the Editing in Authentic View section.

3.5 Authentic View Context Menus

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

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

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



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

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

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

Remove node
Positioning the mouse cursor over the **Remove** command pops up a menu list consisting of the selected node and all its removable ancestors (those that would not invalidate the document) up to the document element. Click the element to be removed. This is a quick way to delete an element or any removable ancestor. Note that clicking an ancestor element will remove all its descendants, including the selected element.

Clear
The **Clear** command clears the element markup from around the selection. If the entire node is

selected, then the element markup is cleared for the entire node. If a text segment is selected, then the element markup is cleared from around that text segment only.

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

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

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

- An ***entire node together with its markup tags*** is highlighted in Authentic View and copied to the clipboard. (i) The node can be pasted as XML to any location where this node may validly be placed. It will not be pasted to an invalid location. (ii) If the node is pasted as text, then only the node's *text content* will be pasted (not the markup); the text content can be pasted to any location in the XML document where text may be pasted.
- A ***text fragment*** is highlighted in Authentic View and copied to the clipboard. (i) If this fragment is pasted as XML, then the XML markup tags of the text—even though these were not explicitly copied with the text fragment—will be pasted along with the text, but only if the XML node is valid at the location where the fragment is pasted. (ii) If the fragment is pasted as text, then it can be pasted to any location in the XML document where text may be pasted.

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

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

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

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

4 Editing in Authentic View

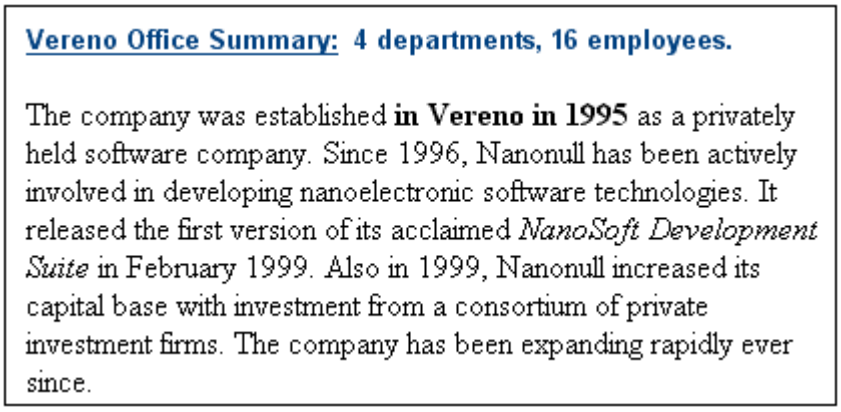
This section describes important features of Authentic View in detail. Features have been included in this section either because they are frequently used or because the mechanisms or concepts involved require explanation.

The section explains the following:

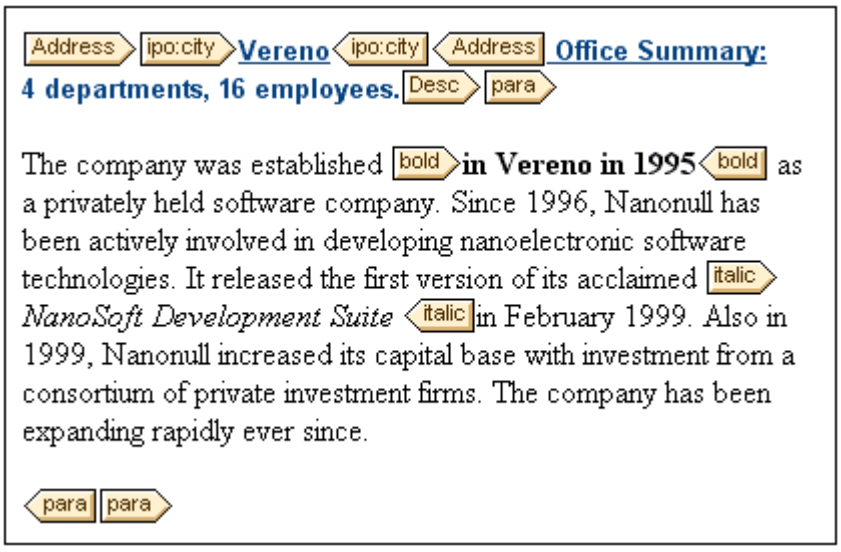
- There are three distinct types of tables used in Authentic View. The section Using tables in Authentic View explains the three types of tables (static SPS, dynamic SPS, and XML), and when and how to use them. It starts with the broad, conceptual picture and moves to the details of usage.
- The Date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

4.1 Basic Editing

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



An editable Authentic View document with no XML markup.

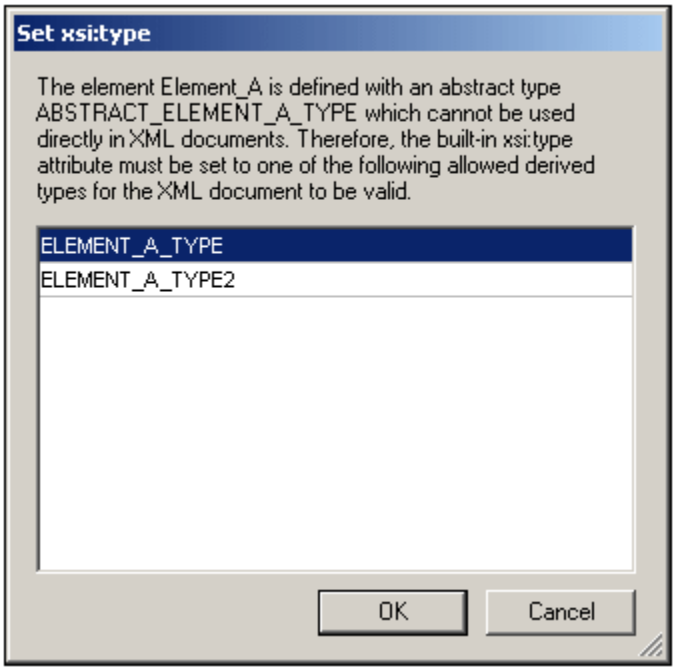


An editable Authentic View document with XML markup tags.

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.



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

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

The selected derived type can be changed subsequently by changing the value of the element's `xsi:type` attribute in the Attributes Entry Helper. When the element's type is changed in this way, all nodes of the previous type's content model are removed and nodes of the new type's content model are inserted.

Text editing
An Authentic View document will essentially consist of text and images. To edit the text in the document, place the cursor at the location where you wish to insert text, and type. You can copy, move, and delete text using familiar keystrokes (such as the **Delete** key) and drag-and-drop mechanisms. One exception is the **Enter** key. Since the Authentic View document is pre-formatted, you do not—and cannot—add extra lines or space between items. The **Enter** key in Authentic View therefore serves to append another instance of the element currently being edited, and should be used exclusively for this purpose.

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

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

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

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

Text formatting

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

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

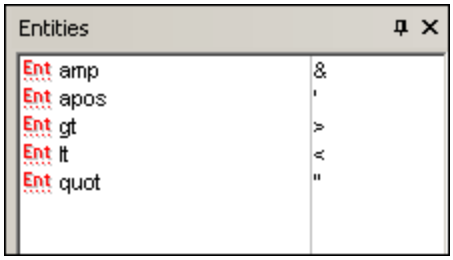
Using RichEdit in Authentic View

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

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

Inserting entities

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



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

Inserting CDATA sections

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

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

Editing and following links

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

4.2 Tables in Authentic View

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

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

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

4.2.1 SPS Tables

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

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

Nanonull, Inc.

Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

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

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

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

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	add Shares	25	7	18
Employees: 3 (20% of Office, 9% of Company)					**Shares: 1500 (13% of Office, 6% of Company)**			
Non-Shareholders: Frank Further, Loby Matise.								

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 a row creates a new row.

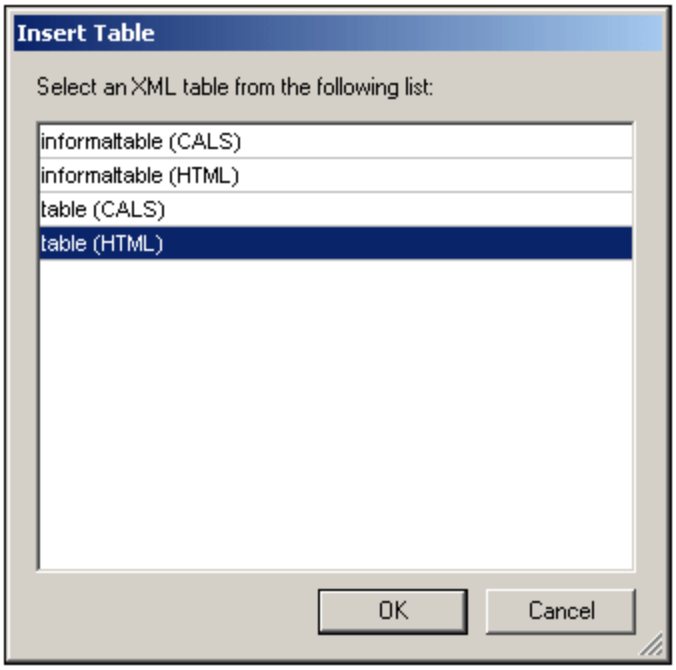
4.2.2 CALS/HTML Tables

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

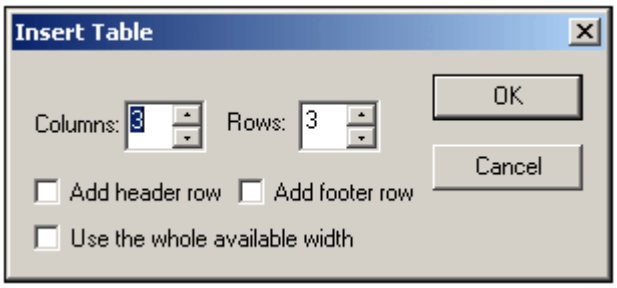
Inserting tables

To insert a CALS/HTML table do the following:

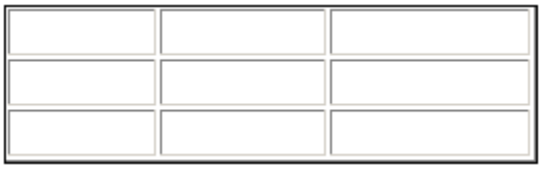
1. Place your cursor where you wish to insert the table, and click the  icon. (Note that where you can insert tables is determined by the schema.) The Insert Table dialog (*screenshot below*) appears. This dialog lists all the XML element data-structures for which a table structure has been defined. 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.

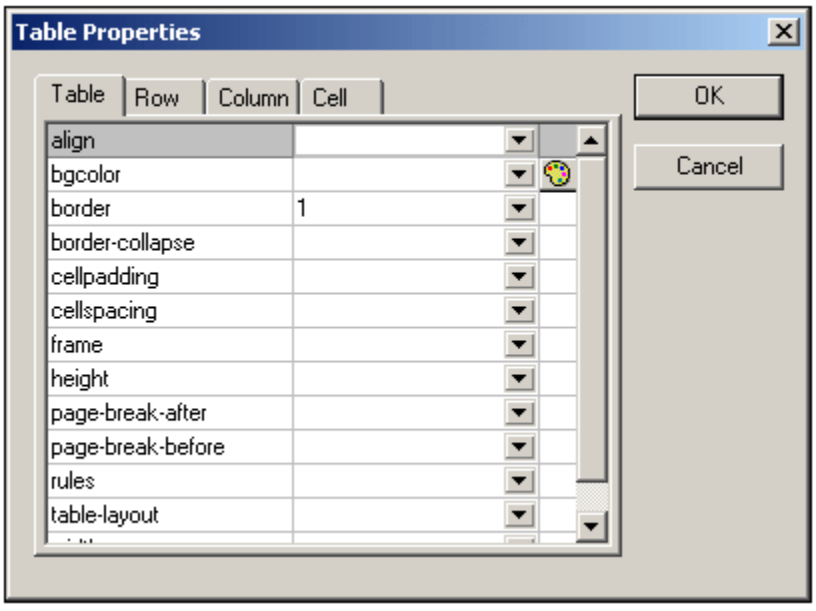
Formatting tables and entering data

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.

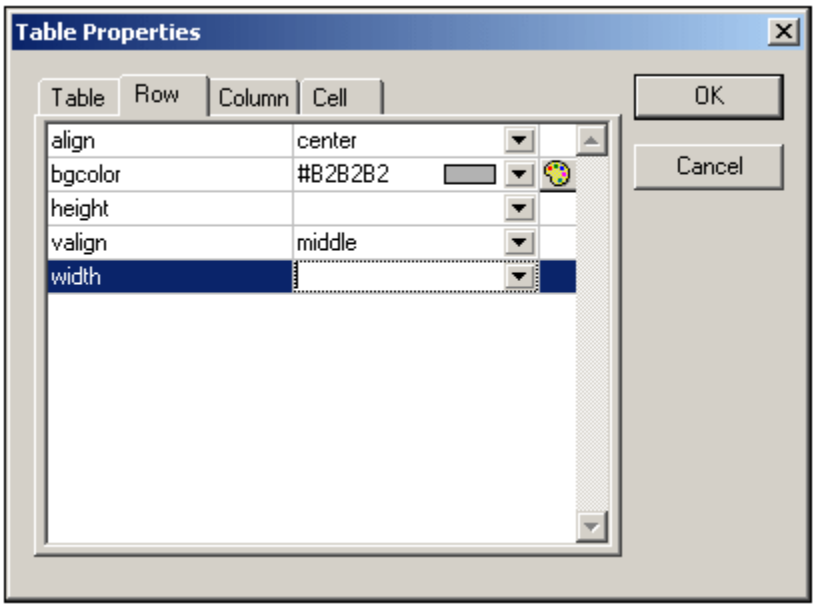
To format a table, row, column, or cell, do the following:

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.

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

Name	Telephone	Email

5. Now place the cursor in the cell below the cell containing "Telephone", and click the (Split horizontally) icon. Then type in the column headers "Office" and "Home". Your table will now look like this:

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.

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.

Name	Telephone		Email
	Office	Home	

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

Please note: For text formatting to be displayed in Authentic View, a global template with the required text formatting must have been created in StyleVision for the element in question.

4.2.3 CALS/HTML Table Editing Icons

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

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

Insert table

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

Delete table

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

Append row

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

Append column

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

Insert row

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

Insert column

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

Join cell left

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

Join cell right

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

Join cell below

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

Join cell above

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

Split cell horizontally

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

Split cell vertically

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

Align top

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

Center vertically

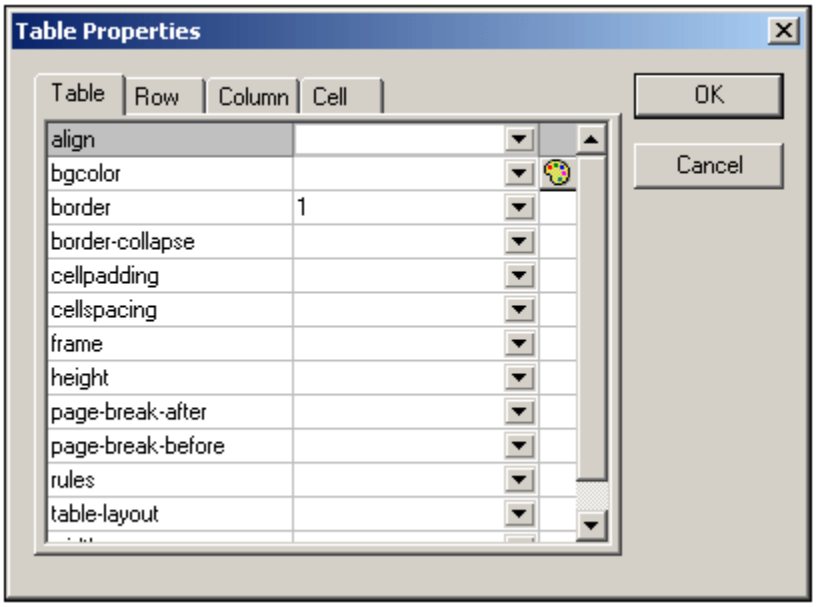
This command centers the cell contents.

Align bottom

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

Table properties

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



4.3 Editing a DB

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

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

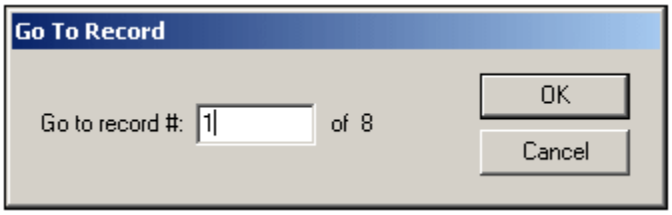
To open a DB-based StyleVision Power Stylesheet in Authentic View:

- Click **Authentic** | **Edit Database Data**, and browse for the required StyleVision Power Stylesheet.

4.3.1 Navigating a DB Table

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

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** menu command.

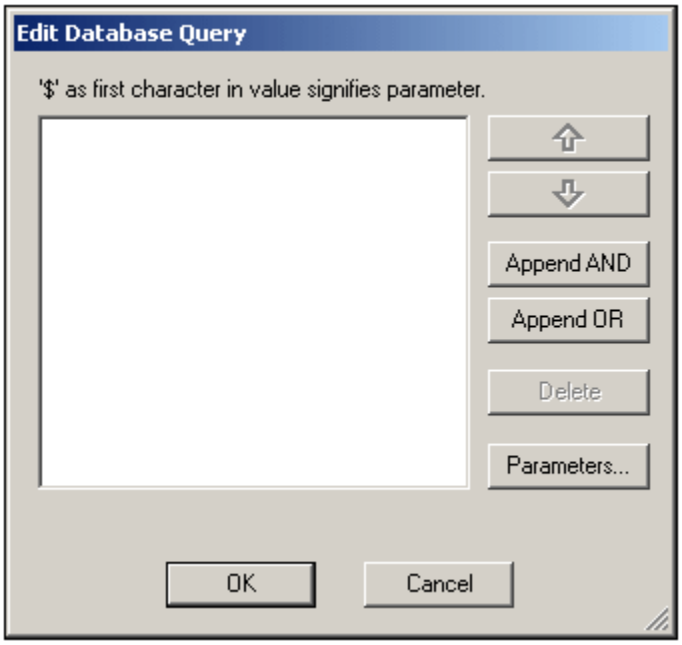
4.3.2 DB Queries

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

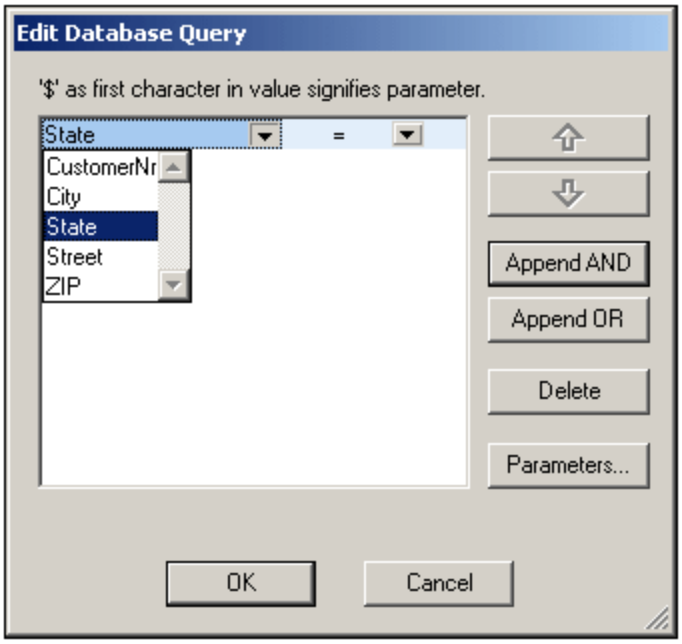
Please note: If you get a message saying that too many tables are open, then you can reduce the number of tables that are open by using a query to filter out some tables.

To create and submit a query:

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



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



4. Enter the expression for the criterion. An expression consists of: (i) a field name (available from the associated combo-box); (ii) an operator (available from the associated combo-box); and (iii) a value (to be entered directly). For details of how to construct expressions see the Expressions in criteria section.
5. If you wish to add another criterion, click the **Append AND** or **Append OR** button according to which logical operator (AND or OR) you wish to use to join the two criteria. Then add the new criterion. For details about the logical operators, see the section Re-ordering criteria in DB Queries.

Expressions in criteria

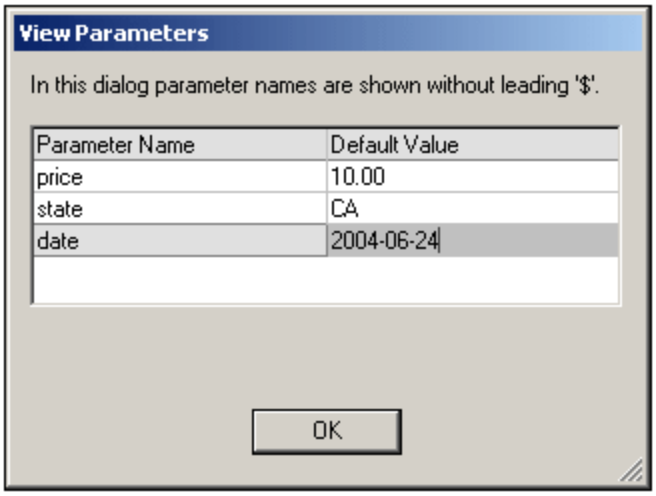
Expressions in DB Query criteria consist of a field name, an operator, and a value. The **available field names** are the child elements of the selected top-level data table; the names of these fields are listed in a combo-box (*see screenshot above*). The **operators** you can use are listed below:

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

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

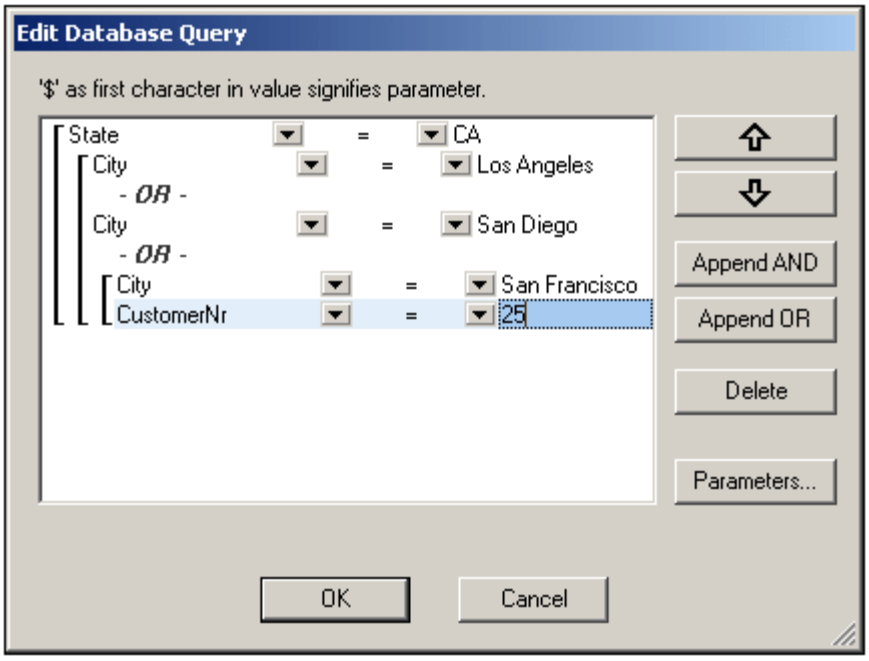
Using parameters with DB Queries
You can enter the name of a **parameter** as the value of an expression when creating queries. Parameters are variables that can be used instead of literal values in queries. When you enter it in an expression, its value is used in the expression. Parameters that are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

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



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

4.3.3 Modifying a DB Table

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

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

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

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

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

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

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

Please note:

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

Deleting a record

To delete a record:

1. Place the cursor in the row representing the record to be deleted and click the  icon. The record to be deleted is marked with a strikethrough. The `AltovaRowStatus` is set as follows: primary instances of the record are set to `D`; secondary instances to `d`; and records indirectly deleted to `X`. Indirectly deleted records are fields in the deleted record

that are held in a separate table. For example, an Address table might be included in a Customer table. If a Customer record were to be deleted, then its corresponding Address record would be indirectly deleted. If an Address record in the Customer table were deleted, then the Address record in the Customer table would be primarily deleted, but the same record would be secondarily deleted in an independent Address table if this were instantiated.

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

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

4.4 Working with Dates

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

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

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

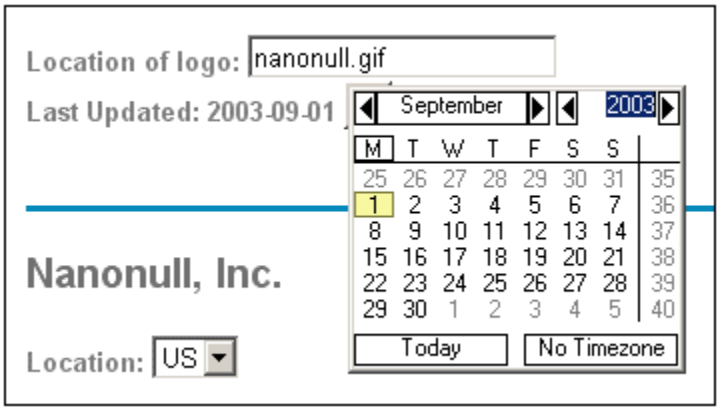
Note on date formats
In the XML document, dates can be stored in one of several date datatypes. Each of these datatypes requires that the date be stored in a particular lexical format in order for the XML document to be valid. For example, the `xs:date` datatype requires a lexical format of `YYYY-MM-DD`.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

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

4.4.1 Date Picker

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

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

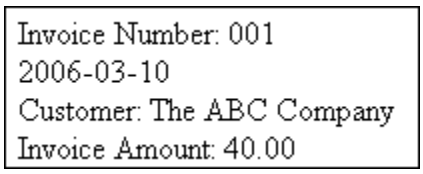


To select a date, click on the desired date, month, or year. The date is entered in the XML document, and the date in the display is modified accordingly. You can also enter a time zone if this is required.

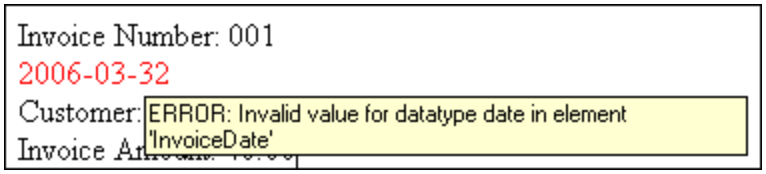
4.4.2 Text Entry

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

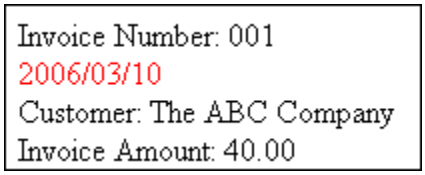
Please note: When editing a date, you must not change its format.



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



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



4.5 Defining Entities

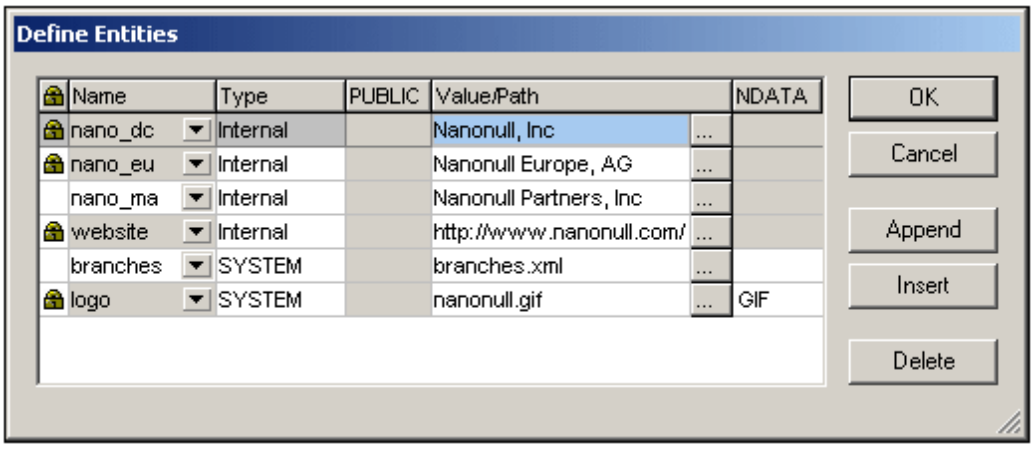
You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the **Insert Entity** submenu of the context menu. When you double-click on an entity in the Entities Entry Helper,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

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

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

To define an entity:

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



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

   - If the entity type is Internal, enter the text string you want as the value of your entity. Do not enter quotes to delimit the entry. Any quotes that you enter will be treated as

part of the text string.
- If the entity type is SYSTEM, enter the URI of the resource or select a resource on your local network by using the Browse button. If the resource contains parsed data, it must be an XML file (i.e., it must have a `.xml` extension). Alternatively, the resource can be a binary file, such as a GIF file.
- If the entity type is PUBLIC, you must additionally enter a system identifier in this field.

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

Dialog features
You can do the following in the Define Entities dialog:

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

Limitations of entities

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

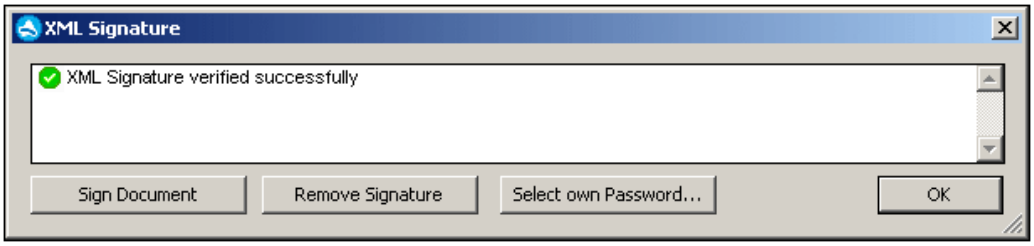
4.6 XML Signatures

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

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

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

- *Choosing the certificate/password:* Signatures are authenticated with either a certificate or a password. The authentication object (certificate or password) is required when the signature is created and again when it is verified. If an Authentic XML document has a signature-enabled SPS assigned to it, the SPS might specify a default certificate or password for the signature. Whether a default certificate or password has been specified or not, the signature can be configured to allow the Authentic View user to select an own certificate/password. The Authentic View user can do this at any time in the XML Signature dialog (s*creenshot 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.

4.7 Images in Authentic View

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

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

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

Relative paths
Relative paths are resolved relative to the SPS file.

4.8 Keystrokes in Authentic View

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

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

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

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

5 Authentic Scripting

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

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

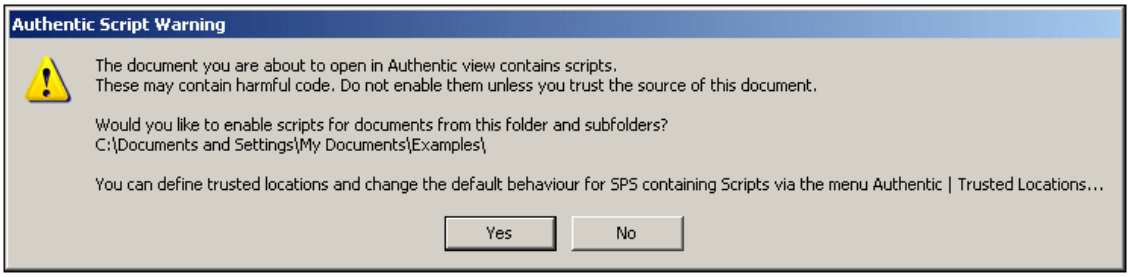
Altova Product	Authentic Scripts Creation	Authentic Scripts Enabled
StyleVision Enterprise	Yes	Yes
StyleVision Professional	Yes	Yes
StyleVision Standard *	No	No
XMLSpy Enterprise	No	Yes
XMLSpy Professional	No	Yes
XMLSpy Standard	No	No
AuthenticDesktop Community	No	No
AuthenticDesktop Enterprise	No	Yes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

* *No AuthenticView*
** *Both Trusted and Untrusted versions*
*** *Scripted designs displayed. No internal macro execution or event handling. External events fired.*

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

Authentic Script Warning Dialog

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



You can choose one of the following options:

- Click **Yes**. to add the folder containing the file to the Trusted Locations list for Authentic scripts. Subsequently, all files in the trusted folder will be opened In Authentic View without this warning dialog being displayed first. The Trusted Locations list can be accessed via the menu command **Authentic | Trusted Locations**, and modified.
- Click **No** to not add the folder containing the file to the Trusted Locations list. The file will be displayed in Authentic View with scripts disabled. The Authentic Script Warning dialog will appear each time this file is opened in Authentic View. To add the file's folder to the Trusted Locations list subsequently, open the Trusted locations dialog via the menu command **Authentic | Trusted Locations**, and add the folder or modify as required.

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

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

How Authentic Scripting works

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

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

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

Documentation for Authentic Scripting

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

6 Browser View

Browser View is typically used to view:

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

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

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

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

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

7 Altova Global Resources

Altova Global Resources is a collection of aliases for file, folder, and database resources. Each alias can have multiple configurations, and each configuration maps to a single resource

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.

A global resource can not only be used to switch resources within an Altova application, but also to generate and use resources from other Altova applications. So, files can be generated on-the-fly in one Altova application for use in another Altova application. All of this tremendously eases and speeds up development and testing.

Using Altova Global Resources involves two processes:

- Defining Global Resources: Resources are defined and the definitions are stored in an XML file. These resources can be shared across multiple Altova applications.
- Using Global Resources: Within an Altova application, files can be located via a global resource instead of via a file path. The advantage is that the resource being used can be instantly changed 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, and DatabaseSpy.

7.1 Defining Global Resources

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

- Click Tools in the menu bar to pop up the **Tools** menu (*screenshot below*), and select the command **Global Resources**. This pops up the Global Resources dialog.

- Click the menu command **Tools | Customize** to display the Customize dialog. Then select the **Toolbars** tab and check the Global Resources option. This switches on the display of the Global Resources toolbar (*screenshot below*).

  Once the toolbar is displayed, click the Manage Global Resources icon. This pops up the Global Resources dialog.

The Global Resources XML File
Information about global resources that you define is stored in an XML file. By default, this XML file is called `GlobalResources.xml`, and it is stored in the `\Altova\` sub-folder of the (My) documents folder. This file is set as the default Global Resources XML File for all Altova applications. As a result, a global resource defined in any application will be available to all Altova applications—assuming that all applications use this file.

You can also re-name the file and save it to any location, if you wish. Consequently, you may have multiple Global Resources XML files. However, only one of these Global Resources XML File can be active at any time, and only the definitions contained in this file will be available to the application.

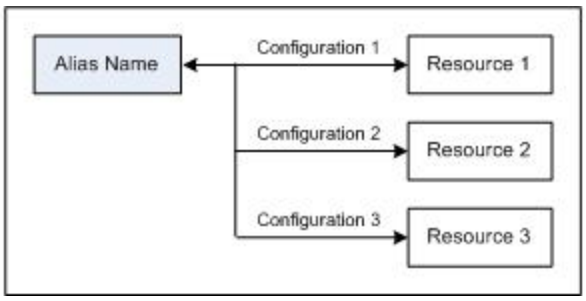
To select a Global Resources XML file to be the active file, in the Manage Global Resources dialog (*screenshot below*), browse for it in the Definitions File entry and select it.

Managing global resources: adding, editing, deleting
In the Manage Global Resources dialog (*screenshot above*), you can add a global resource to the selected Global Resources XML File, or edit or delete a selected global resource. The Global Resources XML File organizes the aliases you add into a list of several sections: files, folders, and databases (*see screenshot above*).

To add a global resource, click the **Add** button and define the global resource in the **Global Resource** dialog that pops up (*see description below*). After you define a global resource and save it, the global resource (or alias) is added to the library of global definitions in the selected Global Resources XML File. To edit a global resource, select it and click **Edit**. This pops up the **Global Resource** dialog, in which you can make the necessary changes (see the descriptions of files, folders, and databases in the sub-sections of this section). To delete a global resource, select it and click **Delete**.

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

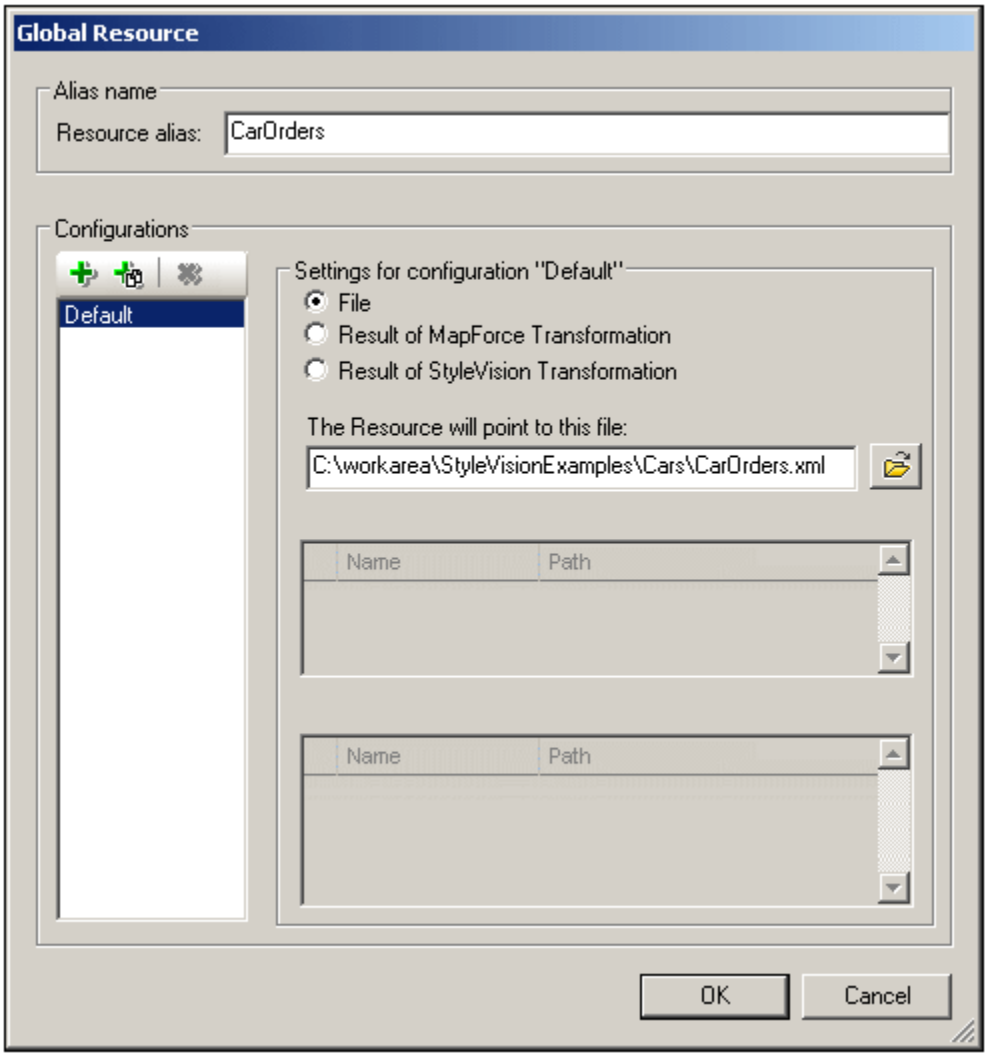
Adding a global resource
Creating a global resource involves mapping one alias name to one or more resources (file, folder, or database). Each mapping is called a configuration. A single alias name can therefore be associated with several resources via different configurations (*screenshot below*).



In the **Manage Global Resources** dialog (*screenshot above*), when you click the **Add** button, you can select whether you wish to add a file-type, folder-type, or database-type resource. How to add and edit each type of resource is described in the sub-sections of this section.

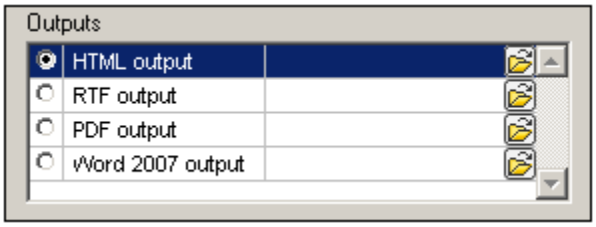
7.1.1 Files

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



1. Enter an alias name.
2. The Configurations pane will have a configuration named Default (*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** icon and, in the **Add Configuration** 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 required for this particular alias (global resource). You can also copy a configuration (using the Add Configuration as Copy icon) and then modify it.
3. Select one of the configurations in the Configurations pane and then define the resource to which this configuration will map. In the Settings for Configuration X pane, you can select whether the resource is a file, or the result of either an Altova MapForce or Altova StyleVision transformation. After selecting the resource type by clicking its radio button, browse for the file, MapForce file, or StyleVision file. Where multiple inputs or outputs for the transformation are possible, a selection of the options will be presented. For example, if the Result of StyleVision Transformation was selected as the resource type, the output options are displayed according to the what edition of

StyleVision is installed (*the screenshot below shows the outputs for Enterprise Edition*).



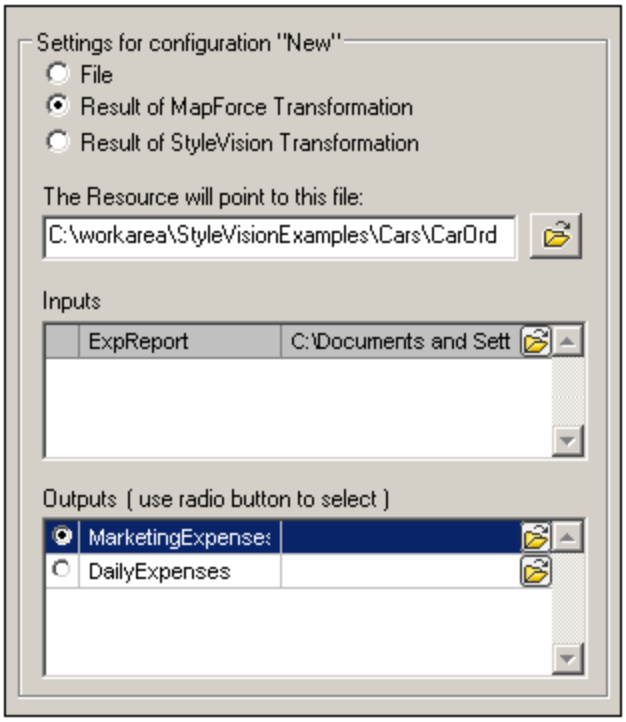
Select the radio button of the desired option (in the screenshot above, 'HTML output' is selected). The result of a transformation can itself be saved as a global resource or as a file path (click the icon and select, respectively, Global Resource or Browse). If neither of these two saving options is selected, the transformation result will be loaded as a temporary file when the global resource is invoked.

4. Specify a resource for each configuration (that is, repeat Step 3 above for the various configurations you have created).
5. 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 Files in the Manage Global Resources dialog.

Selecting Result of MapForce transformations as a global resource

Altova MapForce maps one or more (already existing) schemas to one or more (new) schemas designed by the MapForce user. XML files corresponding to the input schemas are used as data sources, and an output XML file based on the user-designed schema can be generated by MapForce. This generated output file (Result of MapForce Transformation) is the file that will be used as a global resource.

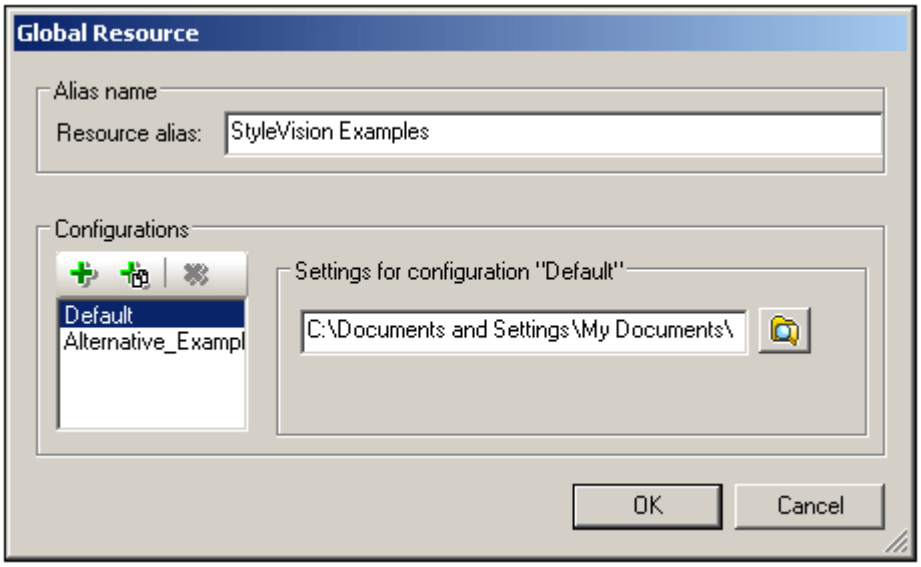
In a MapForce transformation that has multiple output schemas, you can select which one of the output schemas should be used for the global resource by clicking its radio button (*screenshot below*). The XML file that is generated for this schema can be saved as a global resource or as a file path (click the icon and select, respectively, Global Resource or Browse). If neither of these options is selected, a temporary XML file is created when the global resource is used.



Note that each Input can also be saved as a global resource or as a file path (click the icon and select, respectively, Global Resource or Browse).

7.1.2 Folders

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

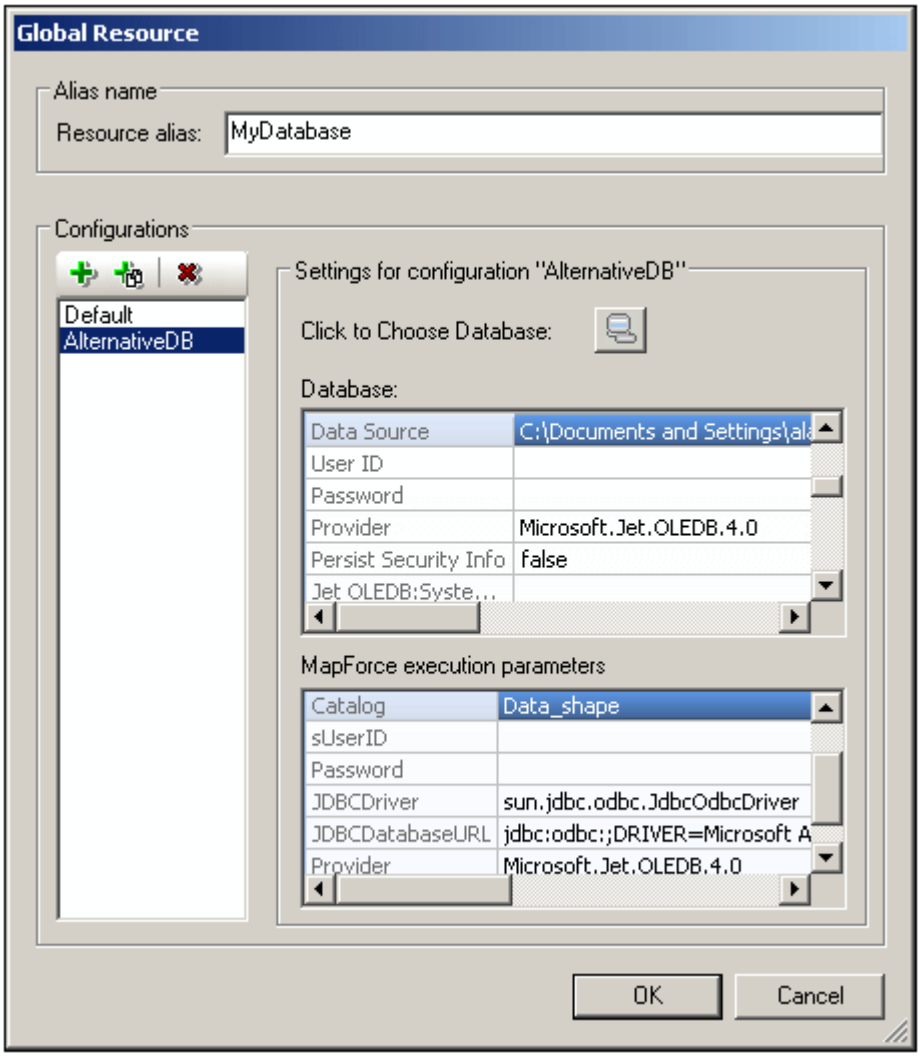


Enter an alias name.

1. The Configurations pane will have a configuration named Default (*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** icon and, in the **Add Configuration** 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 required for this particular alias (global resource).
2. Select one of the configurations in the Configurations pane and browse for the folder you wish to create as a global resource.
3. Specify a folder resource for each configuration (that is, repeat Step 3 above for the various configurations you have created).
4. Click **OK** in the Global Resource dialog to save the alias and all its configurations as a global resource. The global resource will be listed under Folders in the Manage Global Resources dialog.

7.1.3 Databases

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



To define a database-type global resource:

1. Enter an alias name.
2. The Configurations pane will have a configuration named Default (*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** icon and, in the **Add Configuration** 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 required for this particular alias (global resource).
3. Select one of the configurations in the Configurations pane and click the **Choose Database** icon. This pops up the Create Global Resources Connection dialog (*screenshot below*).

4. Select whether you wish to create a connection to the database using the Connection Wizard, an existing connection, an ADO Connection, or an ODBC Connection.
5. If you connect to a database server, you will be prompted to select a DB Root Object on the server. 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, then you can select the Root Object at the time the global resource is loaded.
6. Specify a database resource for each configuration (that is, repeat Steps 3 and 4 above for the various configurations you have created).
7. Click **OK** in the **Global Resource** dialog to save the alias and all its configurations as a global resource. The global resource will be listed under databases in the Manage Global resources dialog.

7.1.4 Copying Configurations

The Manage Global resources dialog allows you to duplicate existing configurations for all types of resources. To do so, select a configuration and click the **Copy Configuration** icon  . Then select or enter a configuration name and click **OK**. This creates a copy of the selected configuration which you can now change as required.

7.2 Using Global Resources

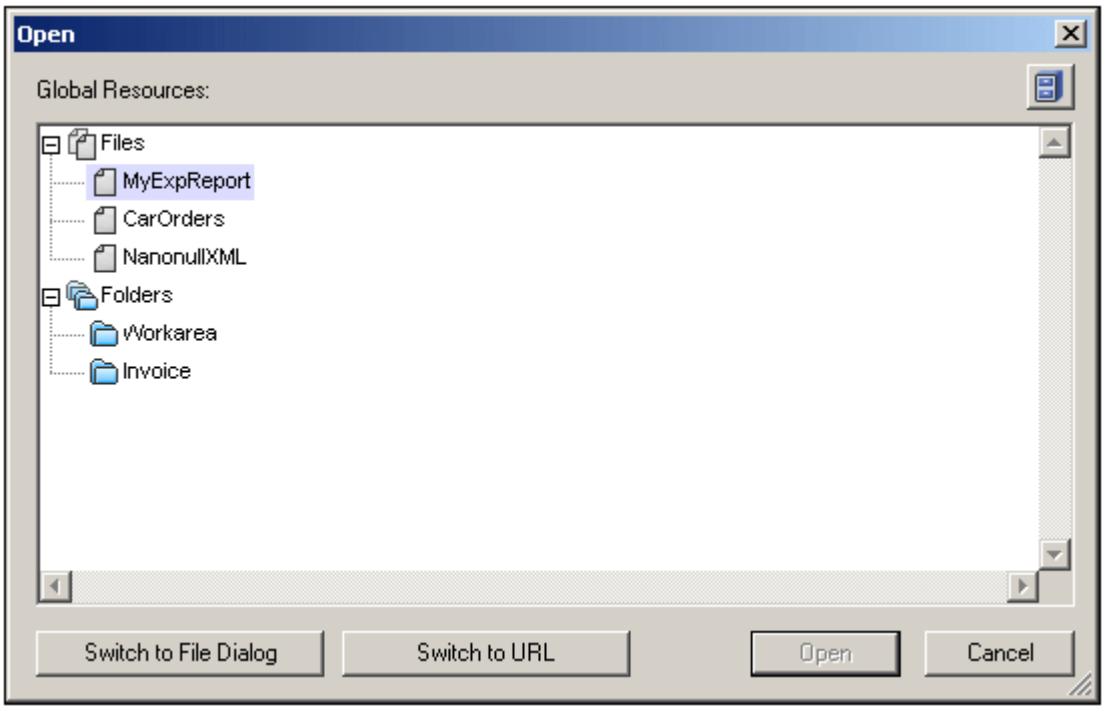
There are several types of global resources (file-type, folder-type , and database-type). The various scenarios in which you can use global resources in Authentic Desktop are listed in this section: Files and Folders.

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

- *The active Global Resources XML File* is selected in the Global Resource dialog. The global-resource definitions that are present in the active Global Resources XML File are available to all files that are open in the application. Only the definitions in the active Global Resources XML File are available. The active Global Resources XML File can be changed at any time, and the global-resource definitions in the new active file will immediately replace those of the previously active file. The active Global Resources XML File therefore determines: (i) what global resources can be assigned, and (ii) what global resources are available for look-up (for example, if a global resource in one Global Resource XML File is assigned but there is no global resource of that name in the currently active Global Resources XML File, then the assigned global resource (alias) cannot be looked up).
- *The active configuration* is selected via the menu item **Tools | Active Configuration** or via the Global Resources toolbar. Clicking this command (or drop-down list in the toolbar) pops up a list of configurations across all aliases. Selecting a configuration makes that configuration active application-wide. This means that wherever a global resource (or alias) is used, the resource corresponding to the active configuration of each used alias will be loaded. The active configuration is applied to all used aliases. If an alias does not have a configuration with the name of the active configuration, then the default configuration of that alias will be used. The active configuration is not relevant when assigning resources; it is significant only when the resources are actually used.

7.2.1 Assigning Files and Folders

In this section, we describe how file-type and folder-type global resources are assigned. File-type and folder-type global resources are assigned differently. In any one of the usage scenarios below, clicking the **Switch to Global Resources** button pops up the Open Global Resource dialog (*screenshot below*).



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 brows 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.

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

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

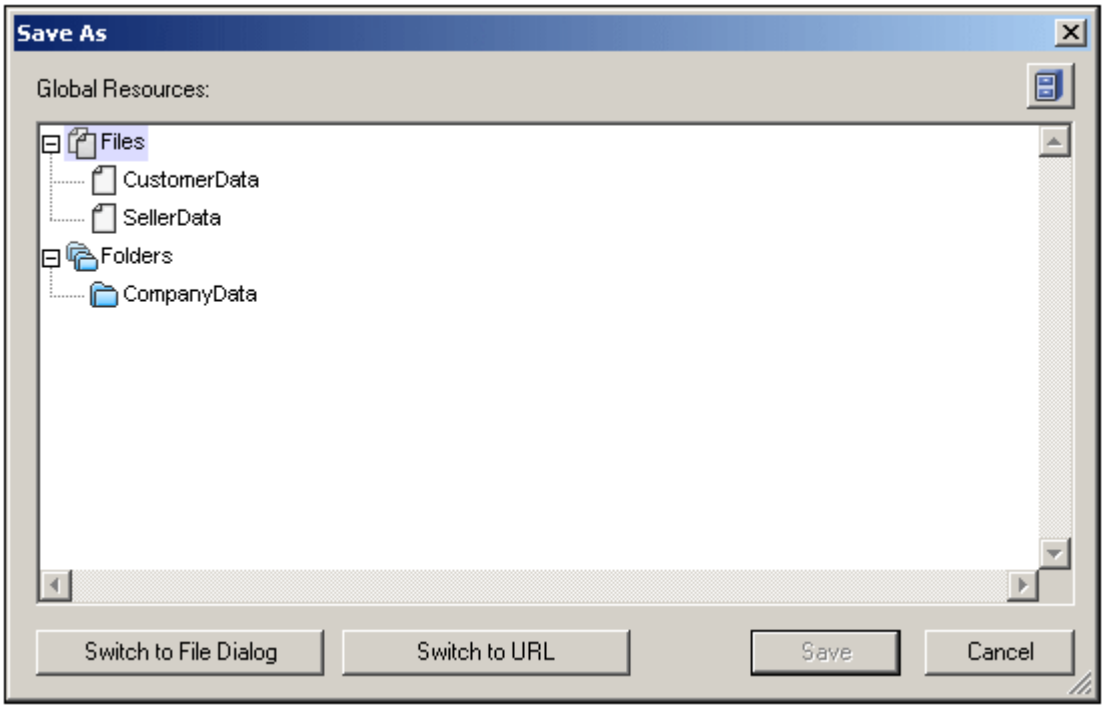
- Opening global resources
- Saving as global resource
- Assigning files for XSLT transformations
- XSLT transformation and XQuery executions
- Assigning an SPS

Opening global resources
A global resource can be opened in Authentic Desktop with the **File | Open (Switch to Global Resource)** command and can be edited. In the case of a file-type global resource, the file is

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

Saving as global resource

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



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

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

Assigning files for XSLT transformations

XSLT files can be assigned to XML documents and XML files to XSLT documents via global resources.. When the commands for assigning XSLT files (**XSL/XQuery | Assign XSL** and **XSL/XQuery | Assign XSL:FO**) and XML files (**XSL/XQuery | Assign Sample XML**) are clicked the assignment dialog pops up. Clicking the **Browse** button pops up the Open dialog, in

which you can switch to the Open Global Resource dialog and select the required global resource. A major advantage of using a global resource to specify files for XSLT transformations is that the XSLT file (or XML file) can be changed for a transformation merely by changing the active configuration in Authentic Desktop; no new file assignments have to be made each time a transformation is required with a different file. When an XSLT transformation is started, it will use the file/s associated with the active configuration.

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

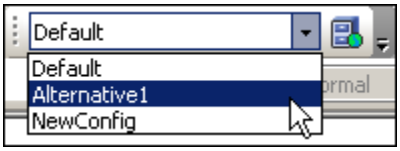
Assigning an SPS
When assigning a StyleVision stylesheet to an XML file (**Authentic | Assign StyleVision Stylesheet**), you can select a global resource to locate the stylesheet. Click the **Browse** button and then the **Switch to Global Resource** button to pop up the Open Global Resource dialog (*screenshot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

7.2.2 Changing Configurations

One global resource configuration can be active at any time, and it is active application-wide. This means that the active configuration is active for all aliases in all currently open files. 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 an XSLT file has been assigned to an XML document via a global resource with multiple configurations. The XSLT file can be switched merely by changing the configuration of the global resource. This can be done in two ways:

- When you hover over the menu command **Tools | Active Configuration**, a submenu with a list of all configurations in the Global Resources XML File pops out. Select the required configuration.
- In the combo box of the Global Resources toolbar (*screenshot below*), select the required configuration. (The Global Resources toolbar can be toggled on and off with the menu command **Tools | Customize | Toolbars | Global Resources**.)



The XSLT file will be changed immediately.

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

8 Source Control

Project files can be placed under source control. A variety of source control systems are supported and Altova has tested support with a large number of drivers and source control systems. The tested systems are listed in the section, Supported Source Control Systems.

Since Authentic Desktop implements the Microsoft Source Code Control Interface (MSSCCI) v1.1 – v1.3, multiple source control systems, including Microsoft SourceSafe and other compatible repositories, are supported.

Note: The **64-bit** version of Authentic Desktop automatically supports any of the supported 32-bit source control programs mentioned in this documentation. When using a 64-bit version of Authentic Desktop with a 32-bit source control program, the "Perform background status updates every... ms" option (Tools | Options) is automatically **grayed-out** and cannot be selected!

Overview of the Source Control feature
The mechanism for placing files in a Authentic Desktop project under source control is as follows:

1. In Authentic Desktop, a project folder containing the files to be placed under source control is created. Typically, the 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. Project files are added to source control using the command **Project | Source Control | Add to Source Control**.
4. Source control actions, such as checking in to, checking out from , and removing files from source control, can be carried out by using the commands in the **Project | Source Control** submenu. The commands in this submenu are listed and described in the Project menu section of the User Reference.

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

Note: Note that a Source Control project is not the same as an Authentic Desktop project. Source Control projects are directory dependent, while Authentic Desktop projects are logical constructions without direct directory dependence.

Registry entry and plug-ins
Microsoft has defined a Registry entry, where all Source Control-compatible programs can register themselves. This is the entry for Authentic Desktop:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders
```

Note that Source Control (SC) plug-ins are not automatically installed by all SC products. Please read the documentation supplied with your specific SC software for more information about plug-ins.

In this section

This section as organized as follows:

- Supported Source Control Systems lists the Source Control Systems (SCSs) that are supported by Authentic Desktop. Systems are listed alphabetically by server, and for each server the supported clients are also listed.
- Installing Source Control Systems contains notes about how to install the various SCSs.
- SCSs and Altova DiffDog Differencing describes how various Source Control Systems can be set up to carry out differencing with Altova DiffDog.

Menu commands
The menu commands for using Source Control Systems are in the submenu **Project | Source Control** and are described in the User Reference section.

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 xxx seconds" field in the Source Control tab accessed through **Tools | Options**.

8.1 Supported Source Control Systems

The list below shows the Source Control Servers (SCSs) supported by Authentic Desktop, together with their respective Source Control Client/s (SCCs). The list is organized alphabetically by SCS. Please read the notes following the list for information about support levels.

AccuRev

Version: AccuRev 4.7.0 Windows

Clients:
- AccuBridge for Microsoft SCC 2008.2

Bazaar

Version: Bazaar 1.9 Windows

Clients:
- Aigenta Unified SCC 1.0.6

Borland StarTeam 2008

Version: StarTeam 2008 Release 2

Clients:
- Borland StarTeam Cross-Platform Client 2008 R2

Codice Software Plastic SCM

Version: Codice Software Plastic SCM Professional 2.7.127.10 (Server)

Clients:
- Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)

Collabnet Subversion 1.5

Version: 1.5.4

Clients:
- Aigenta Unified SCC 1.0.6
- PushOK SVN SCC 1.5.1.1
- PushOK SVN SCC x64 version 1.6.3.1
- TamTam SVN SCC 1.2.24

ComponentSoftware CS-RCS (PRO)

Version: ComponentSoftware CS-RCS (PRO) 5.1

Clients:
- ComponentSoftware CS-RCS (PRO) 5.1

Dynamsoft SourceAnywhere for VSS

Version: Dynamsoft SourceAnywhere for VSS 5.3.2 Standard/Professional Server

Clients:
- Dynamsoft SourceAnywhere for VSS 5.3.2 Client

Dynamsoft SourceAnywhere Hosted

Version: Server hosted in a Bell Data Center

Clients:
- Dynamsoft SourceAnywhere Hosted Client (22252)

Dynamsoft SourceAnywhere Standalone

Version: SourceAnywhere Standalone 2.2 Server

Clients:
- Dynamsoft SourceAnywhere Standalone 2.2 Client

IBM Rational ClearCase 7

Version: 7.0.1 (LT)

Clients:
- IBM Rational ClearCase 7.0.1 (LT)

March-Hare CVSNT 2.5

Version: 2.5.03.2382

Clients:
- Aigenta Unified SCC 1.0.6

March-Hare CVS Suite 2008

Version: Server 2008 [3321]

Clients:
- Jalindi Igloo 1.0.3
- March-Hare CVS Suite Client 2008 (3321)
- PushOK CVS SCC NT 2.1.2.5
- PushOK CVS SCC x64 version 2.2.0.4
- TamTam CVS SCC 1.2.40

Mercurial

Version: Mercurial 1.0.2 for Windows

Clients:
- Sergey Antonov HgSCC 1.0.1

Microsoft SourceSafe 2005

Version: 2005 with CTP

Clients:
- Microsoft SourceSafe 2005 with CTP

Microsoft Visual Studio Team System 2008 Team Foundation Server

Version: 2008

Clients:
- Microsoft Team Foundation Server 2008 MSSCCI Provider

Perforce 2008

Version: P4S 2008.1

Clients:
- Perforce P4V 2008.1

PureCM

Version: PureCM Server 2008/3a

Clients:
- PureCM Client 2008/3a

QSC Team Coherence Version Manager

Version: QSC Team Coherence Server 7.2.1.35

Clients:
- QSC Team Coherence Client 7.2.1.35

Qumasoft QVCS Enterprise

Version: QVCS Enterprise 2.1.18

Clients:
- Qumasoft QVCS Enterprise 2.1.18

Qumasoft QVCS Pro

Version: 3.10.18

Clients:
- Qumasoft QVCS Pro 3.10.18

Reliable Software Code Co-Op

Version: Code Co-Op 5.1a

Clients:
- Reliable Software Code Co-Op 5.1a

Seapine Surround SCM

Version: Surround SCM Client/Server for Windows 2009.0.0

Clients:
- Seapine Surround SCM Client 2009.0.0

Serena Dimensions

Version: Dimensions Express/CM 10.1.3 for Win32 Server

Clients:
- Serena Dimensions 10.1.3 for Win32 Client

Softimage Alienbrain

Version: Alienbrain Server 8.1.0.7300

Clients:
- Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300

SourceGear Fortress

Version: 1.1.4 Server

Clients:
- SourceGear Fortress 1.1.4 Client

SourceGear SourceOffsite

Version: SourceOffsite Server 4.2.0

Clients:
- SourceGear SourceOffsite Client 4.2.0 (Windows)

SourceGear Vault

Version: 4.1.4 Server

Clients:
- SourceGear Vault 4.1.4 Client

VisualSVN Server 1.6

Version: 1.6.2

Clients:
- Aigenta Unified SCC 1.0.6
- PushOK SVN SCC 1.5.1.1
- PushOK SVN SCC x64 version 1.6.3.1
- TamTam SVN SCC 1.2.24

Note the following:

- Altova has implemented the Microsoft Source Code Control Interface (MSSCCI) v1.1 – v1.3 in Authentic Desktop, and has tested support for the drivers and revision control systems listed above. It is expected that Authentic Desktop will continue to support these products if, and when, they are updated.
- Source Control plugins not listed in the table above, but that adhere to the MSCCI 1.1-1.3 specification, should also work together with Authentic Desktop.

8.2 Installing Source Control Systems

This section gives information on how to install and set up the various supported Source Control Systems.

AccuBridge for Microsoft SCC 2008.2
http://www.accurev.com/

1. Install AccuRev client software, run the installer and specify the server you want to connect to (hostname and port) then create a workspace.
2. Install the AccuBridge SCC provider. Extract the ZIP archive into the <AccuRev installation dir>\bin directory.

   Register the AccuRev.dll and SccAcc.dll as follows:
3. Open a command prompt window (if you work with Vista, start Windows Explorer, go to C:\Windows\System32, right click and run cmd.exe "As administrator").
4. Go to the <installation AccuRev dir>\bin directory.
5. Enter the following command at the command prompt:
   - Regsvr32 AccuRev.dll
   - Regsvr32 SccAcc.dll
6. Run the SwitchScc.exe program and set AccuRev as the provider.
7. Perform a Windows log off and log in again.

Aigenta Unified SCC 1.0.6
http://aigenta.com/products/UnifiedScc.aspx

Requirements: source control client. Aigenta Unified SCC works with:

- subversion command line client 1.5.4 at http://subversion.tigris.org
- CVSNT 2.5 (client) at http://www.cvsnt.org
- Bazaar 1.9 Windows (and related pre-requisites)at http://bazaar-vcs.org/ (http://bazaar-vcs.org/WindowsInstall)

A standard installation will work correctly with Altova products.

Borland StarTeam Cross-Platform Client 2008 R2
http://www.borland.com/us/products/starteam

To install the Borland StarTeam Microsoft SCC integration run the setup program and choose to install the SCC API Integration. Altova products can now connect to the repository by specifying the server address, the end point, user and password to log on, your project and working path.

Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)
http://www.codicesoftware.com/xpproducts.aspx

A standard installation will work correctly with Altova products. It is sufficient to install the "client" and the "Visual Studio SCC plug-in" components.

ComponentSoftware CS-RCS (PRO) 5.1
http://www.componentsoftware.com/Products/RCS

1. To install ComponentSoftware CS-RCS (PRO) start the setup and choose the option “Workstation Setup”.
2. Specify your repository tree root and when the installation is finished, restart your machine as requested.
3. Use the “ComponentSoftware RCS Properties” to choose, or create, a project and to specify a work folder.

Dynamsoft SourceAnywhere for VSS 5.3.2 Client
http://www.dynamsoft.com/Products/SAW_Overview.aspx

A standard installation will work correctly with Altova products. To integrate with Altova products you do not need to install the plug-in for Adobe DreamWeaver CS3. After the installation, establish a server connection and set a working folder.

Dynamsoft SourceAnywhere Hosted Client (22252)
http://www.dynamsoft.com/Products/SourceAnywhere-Hosting-Version-Control-Source-Control.aspx

A standard installation will work correctly with Altova products. To integrate with the Altova products you do not need to install the plug-in for Adobe DreamWeaver CS3.

Dynamsoft SourceAnywhere Standalone 2.2 Client
http://www.dynamsoft.com/Products/SourceAnywhere-SourceSafe-VSS.aspx

A standard installation will work correctly with Altova products. To integrate with the Altova products you do not need to install the plug-in for Adobe DreamWeaver CS3. After the installation, establish a server connection and set a working folder.

IBM Rational ClearCase 7.0.1 (LT)
http://www-01.ibm.com/software/awdtools/clearcase/

To install IBM Rational ClearCase LT run the setup.

- You will be asked to update the version of the InstallShield scripting engine if it is older than version 10.5, choose “Update it if necessary”. The update runs prior to the installation starting.
- Choose the default option “Enterprise deployment, create a network release area and customize it using Siteprep”.

To integrate with Altova products, it is sufficient to install only the client. Check only the client check box.

- Provide a server name and the license server element(s) following the examples provided by the installer (port@server_name).
- Provide a configuration description name by editing a name you like and insert the path to a Release area. This path must specify a shared folder.
- You can create a new folder on your machine, share it, and use it as a Release Area. (In Vista, you must set the Network discovery to "on" in Network and Sharing Center to set this path.) The Release Area is now created, some files are copied into it and a shortcut is created with the name **sitedefs.lnk**.
- When all files are copied, continue by clicking the shortcut from Windows Explorer. A

new setup will start to install the client.
- When setup starts, choose the option “Install IBM Rational ClearCase LT”.
- Keep clicking “Next”, accept the “Software License Agreement” and start the installation.

In **Vista**, the second setup could generate the internal error: 2739. In this case, start Windows Explorer and go to C:\Windows\System32.
- Right click and run “cmd.exe” “As Administrator”. A command window pops up.
- Type “regsvr32 jscript.dll”.
- Launch the setup again.

To work with files stored in ClearCase, you should create a view that points to your ClearCase project.

Jalindi Igloo 1.0.3
http://www.jalindi.com/igloo/

To use Jalindi Igloo with Altova products it is sufficient to run the setup to install Jalindi Igloo. Note that if you uninstall Jalindi Igloo, all other installed SCC Provider Windows registry keys (if any) are deleted as well and are not longer available.

When working with Altova products, setting the “Auto Commit” Mode is recommended.
- Auto Commit Mode is found in the advanced Source Control options.
- After defining a workspace, you can start to work.

March-Hare CVS Suite Client 2008 (3321)
http://www.march-hare.com/cvsnt/en.asp

A “typical” installation will work correctly with Altova products.

Mercurial
see under Sergey Antonov HgScc 1.0.1

Microsoft SourceSafe 2005 with CTP
http://msdn.microsoft.com/en-us/vstudio/aa718670.aspx

A standard installation of Microsoft Source Safe 2005 will work correctly with Altova products.

Microsoft Team Foundation Server 2008 MSSCCI Provider
http://www.microsoft.com/downloads

Requirements: Visual Studio 2008 Team Explorer, or Visual Studio 2008 with Team Explorer 2008. A standard installation will work correctly with Altova products.

Perforce P4V 2008.1

http://www.perforce.com/

The Perforce Visual Client (P4V) offers a choice:

- To install all client features (default behavior)
- To install only the “SCC Plug-in (P4SCC)” feature.

The default installation will work correctly with all Altova products.

If the “SCC Plug-in (P4SCC)” feature is chosen:

- Two SCC functions “Show differences” and “Source control manager” will not work.
- The “Show differences” functionality and the possibility to launch the source control manager will not work, because they rely on the non-installed features” Visual Merge Tool” and “Visual Client (P4V)” respectively.
- The differencing functionality will need 3rd party software, while the launch of the source control manager will only be possible after the explicit installation of the “Visual Client (P4V)".
- After starting your Perforce Visual Client installation, specify your own client configuration settings (server name, Text Editing Application, User Name).
- When the installation is finished, do not forget to create a new workspace or to select an existing one.

PureCM Client 2008/3a

http://www.purecm.com/

A standard installation will work correctly with Altova products. After the installation, start the PureCM client to register a server.

PushOK CVS SCC NT 2.1.2.5

http://www.pushok.com/soft_cvs.php

A standard installation is sufficient for using PushOK CVS SCC NT.

- After installation is complete, make sure your copy of the CVS proxy plug-in is correctly registered.
- After defining a workspace, you can start to work.

PushOK CVS SCC x64 version 2.2.0.4

http://www.pushok.com/soft_cvs.php

A standard installation is sufficient for using PushOK CVS SCC.

- After installation is complete, make sure your copy of the CVS proxy plug-in is correctly registered.
- After defining a workspace, you can start to work.

PushOK SVN SCC 1.5.1.1

http://www.pushok.com/soft_svn.php

A standard installation of PushOK SVN SCC is sufficient for use with Altova products. When installing under Vista, it is possible that the COM library **svncom.dll** cannot be registered. In this case, finish the installation, and then register the library manually by following these steps:

1. Start a command window using the option "Run as administrator".
2. Enter: cd “C:\Program Files\PushOK Software\SVNSCC\svn”

3. Type the command > regsvr32 svncom.dll.

PushOK SVN SCC x64 version 1.6.3.1
http://www.pushok.com/soft_svn.php

A standard installation of PushOK SVN SCC is sufficient for use with Altova products. When installing under Vista, it is possible that the COM library **svncom.dll** cannot be registered. In this case, finish the installation, and then register the library manually by following these steps:

1. Start a command window using the option "Run as administrator".
2. Enter: cd "C:\Program Files\PushOK Software\SVNSCC\svn"
3. Type the command > regsvr32 svncom.dll.

QSC Team Coherence Client 7.2.1.35
http://www.teamcoherence.com

A standard installation will work correctly with Altova products.

- If the server is installed on the client machine, a default connection is created after the client installation.
- If the server resides on a different machine, you need to change the "HOSTNAME" property in the Connection Properties dialog of Team Coherence client, to point to the relevant machine.

Qumasoft QVCS Enterprise 2.1.18
http://www.qumasoft.com/

Requirements: J2SE 1.5 or later http://www.oracle.com/technetwork/java/index.html

To install Qumasoft QVCS-Enterprise client, run the installer. If your operating system is **Vista**, you must modify the installation directory from the default value "C:\Program Files\QVCS-Enterprise Client" to "C:\QVCS-Enterprise Client". This must be done as Vista does not let applications write to the C:\Program Files area. Edit the "**setEnv.cmd**" file that resides in the installation directory so that the JAVA_HOME environment variable points to the location of your JVM.

If you work with **Vista** you might have problem when saving the file.

1. If this is the case, start Windows Explorer and go to C:\Windows\System32.
2. Right click and run "cmd.exe" "As Administrator".
3. A command window pops up.
4. Type "cd <installation folder of the QVCS –Enterprise client>"
5. Type "Notepad setEnv.cmd" and then edit the file and save it.
6. From the installation directory of the Qumasoft QVCS-Enterprise client run the batch file "gui.bat".
7. Add a server from the "Server menu" specifying the requested name, IP address and ports, log in and define a local workspace.

Qumasoft QVCS Pro 3.10.18
http://www.qumasoft.com/

To install Qumasoft QVCS-Pro run the installer.

- If your operating system is **Vista**, you must modify the installation directory from the default value “C:\Program Files\QVCSBin“ to “C:\QVCSBin”. This must be done as Vista does not let applications write to the C:\Program Files area.
- After installation is finished, launch the QVCS 3.10 client, create a new user and enable **Ide integration** by selecting the submenu “Ide Integration” in the Admin menu and adding QVCS as a Version Control Tool.
- Create a project and set a workspace.

Reliable Software Code Co-Op 5.1a
http://www.relisoft.com/co_op/index.htm

A standard installation will work correctly with Altova products.

Seapine Surround SCM Client 2009.0.0
http://www.seapine.com/surroundscm.html

A standard installation will work correctly with Altova products. After installation, a server connection must be established.

Serena Dimensions 10.1.3 for Win32 Client
http://www.serena.com/products/dimensions-cm/index.html

Supported Versions: Dimensions Express/CM 10.1.3 for Win32 Client

- Perform a “Typical” installation of the Serena Dimension client.
- Specify the WEB client hostname and port number as requested.

Sergey Antonov HgSCC 1.0.1 for Mercurial SCS
http://www.newsupaplex.pp.ru/hgscc_news_eng.html

A standard installation will work correctly with Altova products.

Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300
http://www.alienbrain.com/

- Perform a “typical” installation of the Alienbrain Client Software, then install the Alienbrain Microsoft Visual Studio Integration. To work with Altova products you do not need to install Microsoft Visual Studio.
- The first time you try to open a project from VCS, or to add a project to VCS you will be asked to enter some user settings, e.g. to specify your server and choose the project database you want to connect to.

SourceGear Fortress 1.1.4 Client
http://www.sourcegear.com/fortress

A standard installation of SourceGear Fortress client will work with Altova products.

SourceGear SourceOffsite Client 4.2.0 (Windows)
http://www.sourcegear.com/sos/

A standard installation of SourceOffsite client will work with Altova products.

SourceGear Vault 4.1.4 Client
http://www.sourcegear.com/vault

A standard installation of SourceGear Vault client will work correctly with Altova products.

TamTam CVS SCC 1.2.40
http://www.daveswebsite.com/software/tamtam/

Requirements: CVSNT 2.5 (client) at http://www.cvsnt.org. A standard installation will work correctly with Altova products.

- To connect to the CVS repository you need to install CVSNT.
- In the Altova product open the "Source control" Advanced options and enter the path to the **cvs.exe** executable.

TamTam SVN SCC 1.2.24
http://www.daveswebsite.com/software/tamtamsvn/

Requirements: subversion command line client 1.5.4 at http://subversion.tigris.org . A standard installation will work correctly with Altova products.

- To connect to the SVN repository you need to install the subversion command line client and specify the path to the executable **svn.exe** in the Altova product Source control options.
- After starting Authentic Desktop, you must register the SCC provider.

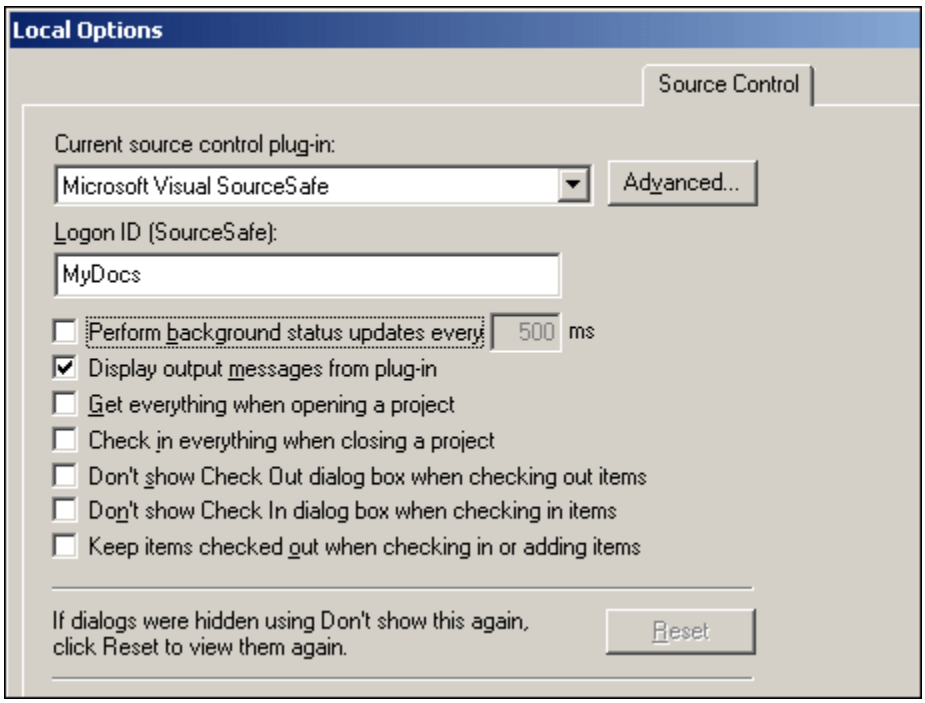
On a **Vista** machine the SCC registration could fail.

- If this is the case, use Windows explorer and browse to the directory that contains the Altova application executable.
- Right click and run the Altova executable "As Administrator".
  The SCC registration will now be successful.

8.3 SCSs and Altova DiffDog Differencing

You can configure certain Source Control Systems so that they use Altova DiffDog as their differencing tools. The systems that support this feature are listed below, together with the setup steps for each. In Authentic Desktop, you access the setup process via the Source Control tab of the Options dialog (**Tools | Options**, *screenshot below*).

When using a 64-bit version of Authentic Desktop with a 32-bit source control program, the "Perform background status updates every... ms" option (Tools | Options) is automatically **grayed-out** and cannot be selected!



The "Perform background status updates every xx ms" check box is unchecked per default, which means that status updates are not performed at all. Activate the check box and enter a value in the field, if you want to perform status updates every xx ms. For 64-bit versions using 32-bit source control plugins, this option has no effect.

In the Source Control tab, select the required Source Control System and then click the **Advanced** button. The dialog box that opens will be different for each Source Control System. For the setup process, note the following:

- If you have performed a standard installation of Altova DiffDog, the file path to the Altova DiffDog executable is:
  `c:\program files\altova\diffdog2012\DiffDog.exe`
  If Altova DiffDog is installed elsewhere on your system, insert the appropriate value when the filepath is required.

Aigenta Unified SCC 1.0.6
http://aigenta.com/products/UnifiedScc.aspx

The following steps will integrate Altova DiffDog into Aigenta Unified SCC:

1. Click the **Advanced** button of the Source Control tab.
2. Select the *Comparison and merging tab* and specify the full DiffDog filepath as comparison tool.

Borland StarTeam Cross-Platform Client 2008 R2
http://www.borland.com/us/products/starteam

The following steps integrate Altova DiffDog into Borland Star Team:
1. Use the StarTeam client personal options (**Tools | Personal options | File | Alternate applications**)
2. Compare utility: Enter the DiffDog full path.
3. Compare utility options: `$file1 $file2.`

ComponentSoftware CS-RCS (PRO) 5.1
http://www.componentsoftware.com/Products/RCS

The following steps will integrate Altova DiffDog into ComponentSoftware CS-RCS (Pro):
1. Go to the ComponentSoftware CS-RCS Properties.
2. In the *File Types* tab, choose a file extension and edit it.
3. Enter/select the value *Custom Tool* for the "Difference Analysis Tool", and browse to insert the DiffDog full path.

Dynamsoft SourceAnywhere for VSS 5.3.2 Client
http://www.dynamsoft.com/Products/SAW_Overview.aspx

The following steps will integrate Altova DiffDog into Dynamsoft SourceAnywhere for VSS:
1. Go to the Dynamic SourceAnywhere For VSS client Options.
2. Specify the DiffDog full path as External application for diff/merge, with the arguments: `%FIRST_FILE%" "%SECOND_FILE%.`

Warning: Do not perform these settings from the Altova product options, as there is no possibility of inserting the external application parameters.

Dynamsoft SourceAnywhere Hosted Client (22252)
http://www.dynamsoft.com/Products/SourceAnywhere-Hosting-Version-Control-Source-Control.aspx

Dynamsoft SourceAnywhere Standalone 2.2 Client
http://www.dynamsoft.com/Products/SourceAnywhere-SourceSafe-VSS.aspx

The following steps will integrate Altova DiffDog into Dynamsoft SourceAnywhere Hosted and Dynamsoft SourceAnywhere Standalone:
1. Click the **Advanced** button of the Source Control tab.
2. Specify the DiffDog full path as External program application for diff/merge with arguments `%FIRST_FILE%" "%SECOND_FILE%.`

Jalindi Igloo 1.0.3
http://www.jalindi.com/igloo/

The following steps will integrate Altova DiffDog into Jalindi Igloo:
1. Start the **Show differences** command in Authentic Desktop.

2. Open the **Show Differences or Merge Files** panel.
3. Set the *External Diff Command* by entering the DiffDog full file path as the External Diff EXE path.

Warning: When using the default diff editor CvsConflictEditor, you might have problems comparing files with excessively long lines. We recommended that you "pretty print" all files (particularly `.ump` files) before storing them in the repository. This limits the line length, thus avoiding problems with the CVSConflictEditor.

March-Hare CVS Suite Client 2008 (3321)
http://www.march-hare.com/cvsnt/en.asp

The following steps will integrate Altova DiffDog into Marc-Hare CVS Suite 2008:

1. Go to the TortoiseCVS Preferences and choose the Tools tab.
2. Specify the DiffDog full path as Diff application, and the parameters `%1 %2` as two-way differencing parameters.

Mercurial
see under Sergey Antonov HgScc 1.0.1

Microsoft SourceSafe 2005 with CTP
http://msdn.microsoft.com/en-us/vstudio/aa718670.aspx

The following steps will integrate Altova DiffDog into Microsoft SourceSafe 2005:

1. Click the **Advanced** button of the Source Control tab.
2. Click the Custom Editors tab and enter `C:\Program Files\Altova\DiffDog2012\DiffDogexe %1 %2` in the Command Line field.
3. In the Operation combo box, select *File Difference*.

Microsoft Team Foundation Server 2008 MSSCCI Provider
http://www.microsoft.com/downloads

Requirements: Visual Studio 2008 Team Explorer *or* Visual Studio 2008 *with* Team Explorer 2008. The following steps will integrate Altova DiffDog into Microsoft Visual Studio Team System 2008 Team Foundation Server MSSCCI Provider:

1. In the manager (Visual Studio 2008 Team Explorer or Visual Studio 2008) options, configure Altova DiffDog as new user tool
2. Choose Visual Studio Team Foundation Server source as the plug-in.
3. Configure a new user tool specifying: (i) the extensions of the files you wish to compare with DiffDog; and (ii) the DiffDog full file path.

Perforce P4V 2008.1
http://www.perforce.com/

The following steps will integrate Altova DiffDog into Perforce 2008:

1. Click the **Advanced** button of the Source Control tab.
2. Choose the tab Diff in the Preferences panel.
3. Check as default differencing application the field “Other application” and enter the DiffDog full file path.

PushOK CVS SCC NT 2.1.2.5,
PushOK SVN SCC 1.5.1.1
PushOK CVS SCC x64 version 2.2.0.4
PushOK SVN SCC x64 version 1.6.3.1
http://www.pushok.com/soft_cvs.php

The following steps will integrate Altova DiffDog into PushOK CVS NT and PushOK SVN SCC:

1. Click the **Advanced** button of the Source Control tab.
2. Choose the CVS Executables tab.
3. Select the value *External merge/compare tool* into the Diff/Merge field.
4. Insert the DiffDog full file path.
5. Edit the value `%first %second` into the "2 way diff cmd" field.

Warning: When using the default differencing editor CvsConflictEditor, you might have problems comparing files with excessively long lines. We recommended that you "pretty print" all files (particularly `.ump` files) before storing them in the repository. This limits the line length, thus avoiding problems with the CVSConflictEditor.

QSC Team Coherence Client 7.2.1.35
http://www.teamcoherence.com

The following steps will integrate Altova DiffDog into Team Coherence Version Manager:

1. Go to Team Coherence client Options “Difference Viewer”.
2. Specify as the Default Difference Viewer application, the DiffDog full file path.
3. Specify as parameters: `"$LF $RF"`.

Warning: It is possible that the new settings will only be applied after a Windows log off.

Qumasoft QVCS Enterprise 2.1.18
http://www.qumasoft.com/

The following steps will integrate Altova DiffDog into Qumasoft QVCS-Enterprise:

1. Add the Qumasoft QVCS-Enterprise installation directory to the Path environment variable.
2. Use the QVCS Enterprise User Preferences.
3. In Utilities, enable the checkbox, Use External Visual Compare Tool.
4. Specify as Visual Compare Command Line:
   `<DiffDog full path> “file1Name file2Name”`

Qumasoft QVCS Pro 3.10.18
http://www.qumasoft.com/

The following steps will integrate Altova DiffDog into Qumasoft QVCS-Pro:

1. Use the QVCS 3.10 client preferences.
2. In Utilities, specify the DiffDog full path as the visual compare utility with parameters “%s %s “.

Seapine Surround SCM Client 2009.0.0
http://www.seapine.com/surroundscm.html

The following steps will integrate Altova DiffDog into Seapine Surround SCM:
1. Go to the Surround SCM client user options (Diff/Merge) section.
2. Edit the Diff/Merge settings to compare with a selected application.
3. Enter the DiffDog full path with the parameters “%1” “%2”.
4. Restart the Surround SCM client and the Altova products.

Sergey Antonov HgSCC 1.0.1
http://www.newsupaplex.pp.ru/hgscc_news_eng.html

The following steps will integrate Altova DiffDog into Mercurial:
1. Click the **Advanced** button of the Source Control tab.
2. Select differencing tool “custom”, and specify the DiffDog full path.

SourceGear Fortress 1.1.4 Client
http://www.sourcegear.com/fortress

SourceGear Vault 4.1.4 Client
http://www.sourcegear.com/vault

The following steps will integrate Altova DiffDog into SourceGear Fortress and SourceGear Vault:
1. Click the **Advanced** button of the Source Control tab.
2. Set the Diff/Merge Vault options by specifying as the differencing program the DiffDog full path and using the Arguments:
   /ro1 /ro2 /title1:"%LEFT_LABEL%" /title2:"%RIGHT_LABEL%" "%LEFT_PATH%" "%RIGHT_PATH%"

SourceGear SourceOffsite Client 4.2.0 (Windows)
http://www.sourcegear.com/sos/

The following steps will integrate DiffDog into SourceGear SourceOffsite:
1. Click the **Advanced** button of the Source Control tab.
2. Specify as “External Programs”, “Application for comparing files” the DiffDog full path.

TamTam CVS SCC 1.2.40,
TamTam SVN SCC 1.2.24
http://www.daveswebsite.com/software/tamtam/

The following steps will integrate Altova DiffDog into TamTam CVS SCC and TamTam SVN SCC:
1. Click the **Advanced** button of the Source Control tab.
2. Specify the DiffDog full file path as the external tool for Diff/Merge and Conflict.

Warning: The default differencing editor CvsConflictEditor, has problems comparing files with excessively long lines. We recommended that you "pretty print" all files (particularly `.ump` files) before storing them in the repository. This limits the line length, avoiding problems with the CVSConflictEditor.

9 Authentic Desktop in Visual Studio

Authentic Desktop can be integrated into the Microsoft Visual Studio IDE versions 2005, 2008, and 2010. This unifies the best of both worlds, integrating XML editing capabilities with the advanced development environment of Visual Studio.

In this section, we describe:

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

9.1 Installing the Authentic Plugin

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

- Install Microsoft Visual Studio
- Install Authentic Desktop
- 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. (**Please note:** You must use the integration package corresponding to your Authentic Desktop version (to see the version and release number, click the menu command **Help | About Authentic**).)

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

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

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

9.2 Differences with Standalone Version

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

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

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

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

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

Additional Notes
Some additional notes and tips are given below:

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

10 Authentic Desktop in Eclipse

Eclipse 3.x is an open source framework that integrates different types of applications delivered in the form of plugins.

The Authentic Plugin for Eclipse enables you to access the functionality of Authentic Desktop from within the Eclipse 3.5 / 3.6 / 3.7 Platform. It is available on Windows platforms. In this section, we describe how to install the Authentic Plugin for Eclipse and how to set up the Authentic perspective. After you have done this, components of the Authentic Desktop GUI and Authentic Desktop menu commands will be available within the Eclipse GUI.

10.1 Installing the Authentic Desktop Plugin for Eclipse

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

- Authentic Desktop Edition.
- Java Runtime Environment (JRE) version 1.5 or higher, which is required for Eclipse. JRE5 is recommended. See the Eclipse website for more information.
- Eclipse Platform 3.5 / 3.6 / 3.7.

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

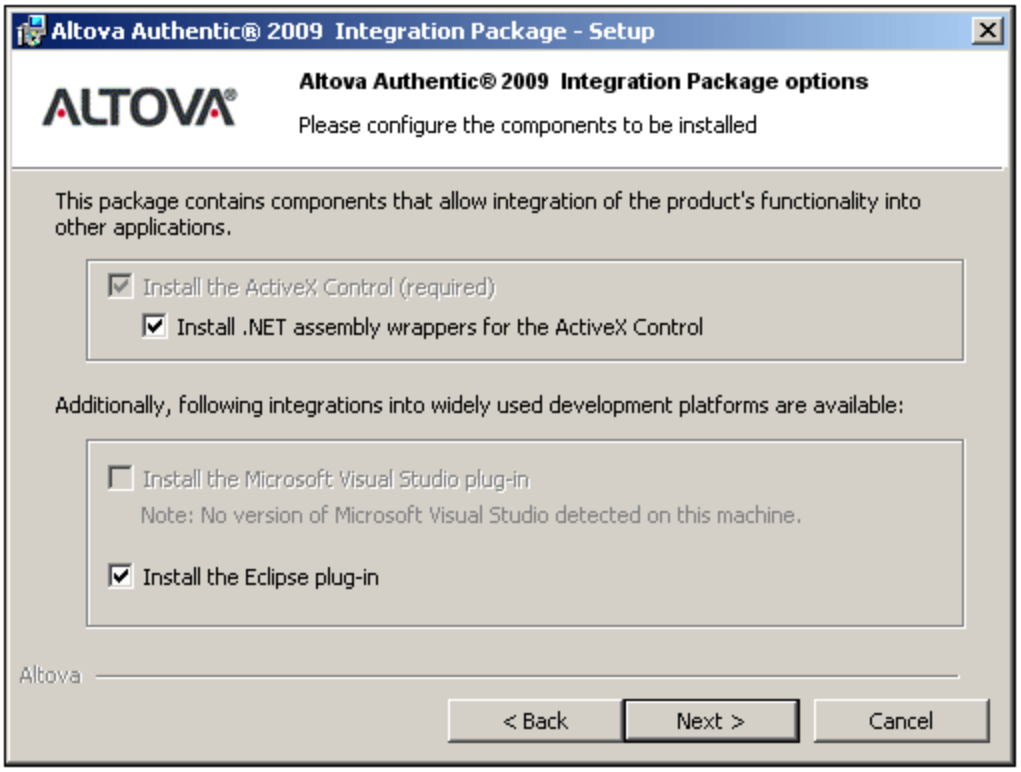
Note on JRE
If, on opening a document in Eclipse, you receive the following error message:

```
java.lang.UnsupportedClassVersionError: com/altova/....
(Unsupported major.minor version 49.0)
```

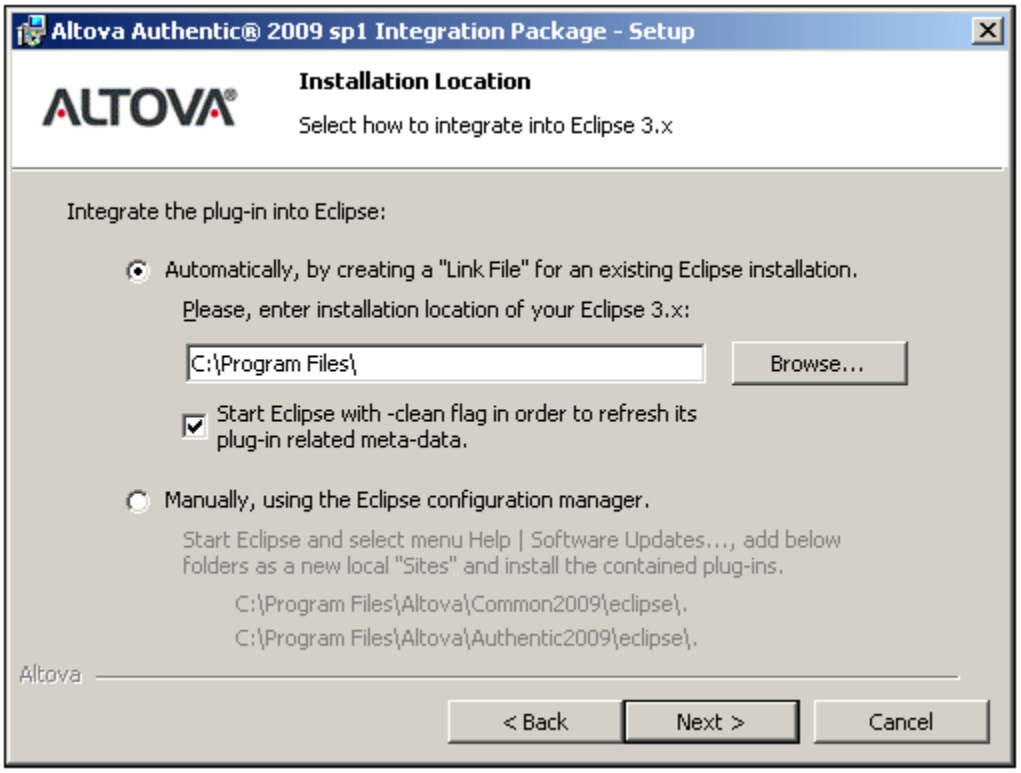
it indicates that Eclipse is using an older JRE. Since Eclipse uses the `PATH` environment variable to find a `javaw.exe`, the problem can be solved by fixing the `PATH` environment variable so that a newer version is found first. Alternatively, start Eclipse with the command line parameter `-vm`, supplying the path to a `javaw.exe` of version 1.5 or higher.

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

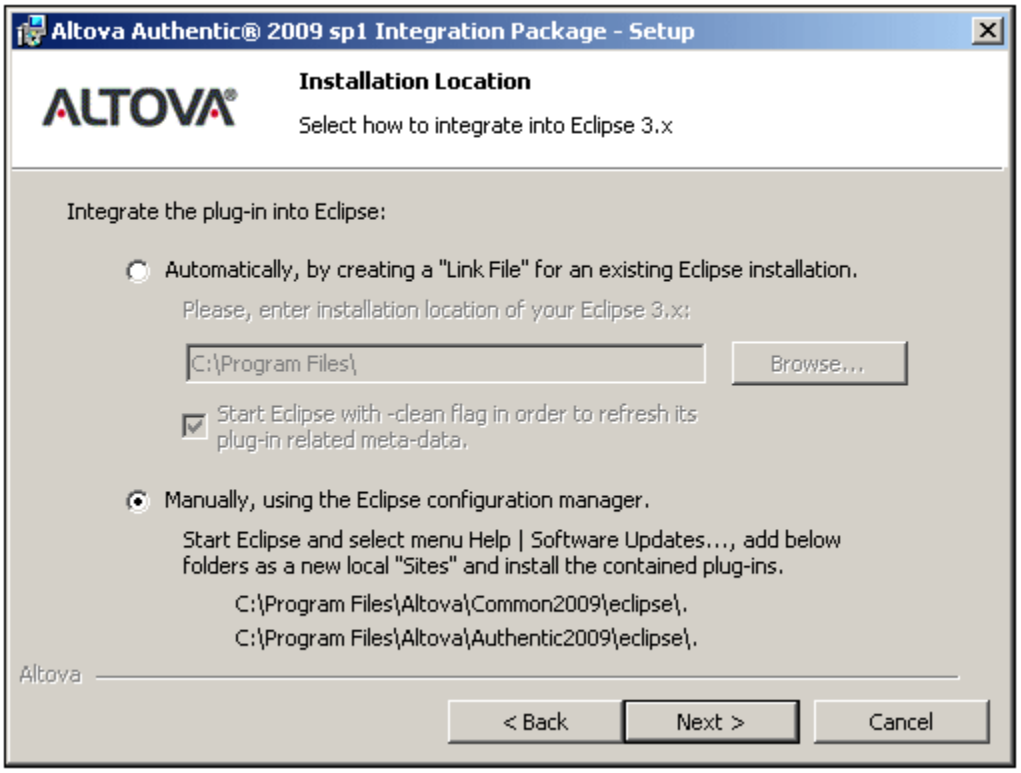
1. Ensure that Authentic Desktop, JRE, and Eclipse are already installed (*see above*).
2. From the Components Download page of the Altova website, download and install the Authentic Integration Package. There are two important steps during the installation; these are described in Steps 3 and 4 below.
3. During installation of the Authentic Integration Package, a dialog will appear asking whether you wish to install the Authentic Plugin for Eclipse (*see screenshot below*). Check the option and then click **Next**.



4. In the next dialog ((Eclipse) Installation Location, *screenshot below*), you can choose whether the Install Wizard should integrate theAuthentic Plugin into Eclipse during the installation (the "Automatic" option) or whether you will integrate the Authentic Plugin into Eclipse (via the Eclipse GUI) at a later time.



We recommend that you let the Installation Wizard do the integration. Do this by checking the Automatic option and then browsing for the folder in which the Eclipse executable (`eclipse.exe`) is located. Click **Next** when done. If you choose to manually integrate Authentic Plugin for Eclipse in Eclipse, select the Manually option (*screenshot below*). See the section below for instructions about how to manually integrate from within Eclipse.

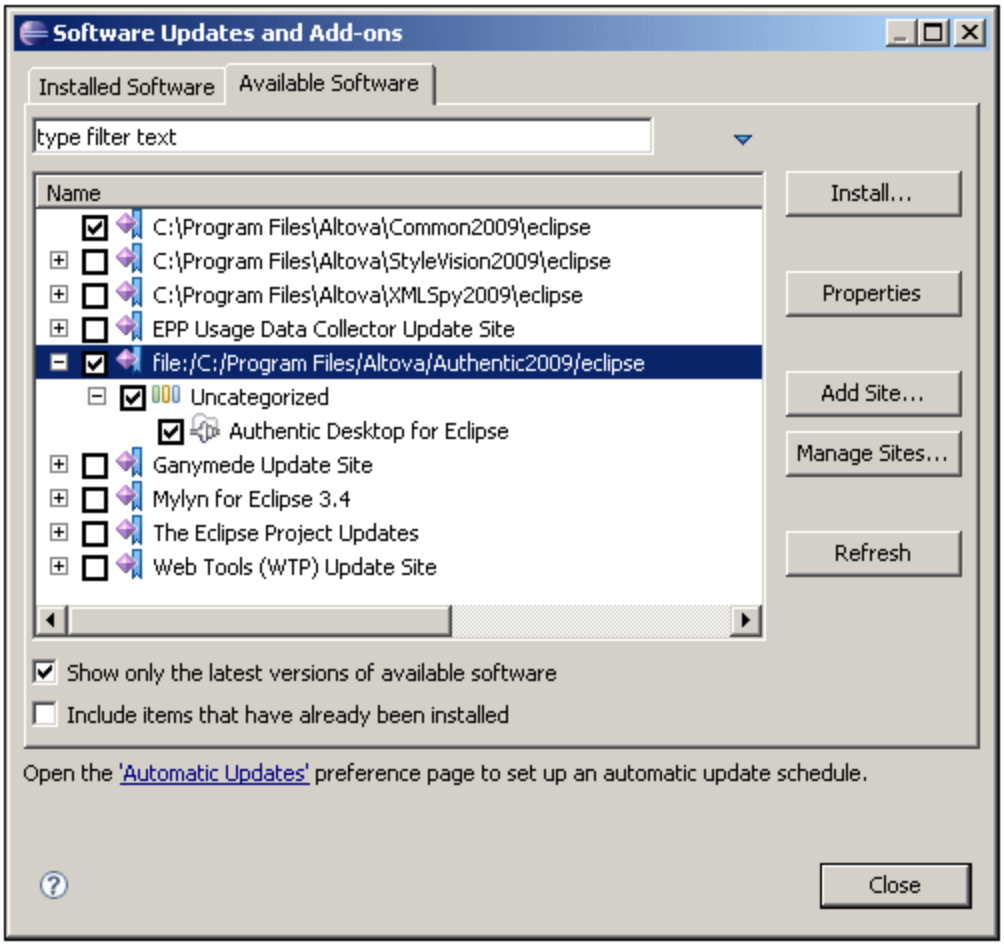


5. Complete the installation. If you set up automatic integration, the Authentic Plugin for Eclipse will be integrated in Eclipse and will be available when you start Eclipse the next time.

Manually integrating the Authentic plugin in Eclipse

To manually integrate the Authentic Plugin for Eclipse, do the following:

1. In Eclipse, click the menu command **Help | Software Updates**.
2. In the Software Updates and Add-Ons dialog that pops up, click the Available Updates tab (*screenshot below*).
3. Click the **Add Site** button.
4. In the Add Site dialog that pops up, click the **Local** button.
5. Browse for the folder `c:\Program Files\Altova\Common2012\eclipse`, select it, and click **OK**.
6. Repeat Steps 3 to 5, this time selecting the folder `c:\Program Files\Altova\Authentic2012\eclipse`.
7. The two added folders are displayed in the Available Software tab (*screenshot below*). Check the top-level check box of each folder to select the plug-ins, and click the **Install** button.



8. An Installation review dialog box allowing you to confirm that the checked items will be installed opens.
9. Click **Next** to continue.The Review License dialog opens.
10. Read the license terms and, if you accept them, click *I accept the terms...*. Then click **Finish** to complete the installation.

If there are problems with the plug-in (missing icons, for example), start Eclipse with the `-clean` flag.

Currently installed version
To check the currently installed version of the Authentic Plugin for Eclipse, select the Eclipse menu option **Help | About Eclipse Platform**. Then select the Authentic icon.

10.2 Authentic Desktop Entry Points in Eclipse

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

- Authentic Desktop Perspective, which provides Authentic Desktop's GUI features within the Eclipse GUI.
- Authentic Desktop toolbar buttons, which provides access to Authentic Desktop Help and the Create New Document functionality.

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

When a file having a filetype associated with Authentic Desktop is opened (`.sps` or `.xml`), this file can be edited in the Authentic perspective. If a filetype is not associated with the Authentic Desktop application, it can be opened in the Autnetic perspective as follows: (i) right-click the file in the Projects sidebar; (ii) select the command **Edit With**; and (iii) browse and select the Authentic Desktop application.

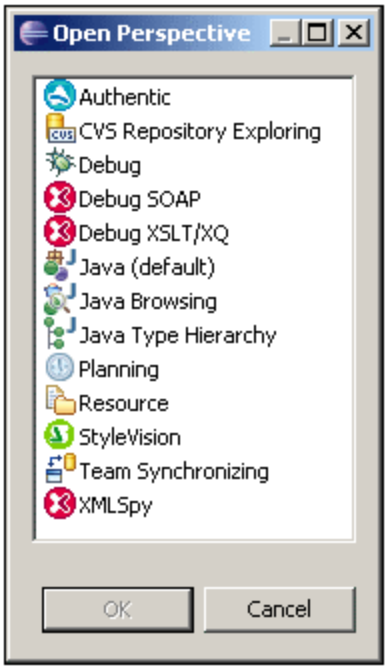
Similarly, a file of another filetype (and associated with another application) can be opened in another perspective in Eclipse. Additionally, for any active file, you can switch the perspective, thus allowing you to edit or process that file in another environment. There are therefore two main advantage of perspectives:

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

Working with the Authentic perspective involves the following:

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

Switching to the Authentic perspective
In Eclipse, select the command **Window | Open Perspective | Other**. In the dialog that pops up (*screenshot below*), select **Authentic**, and click **OK**.

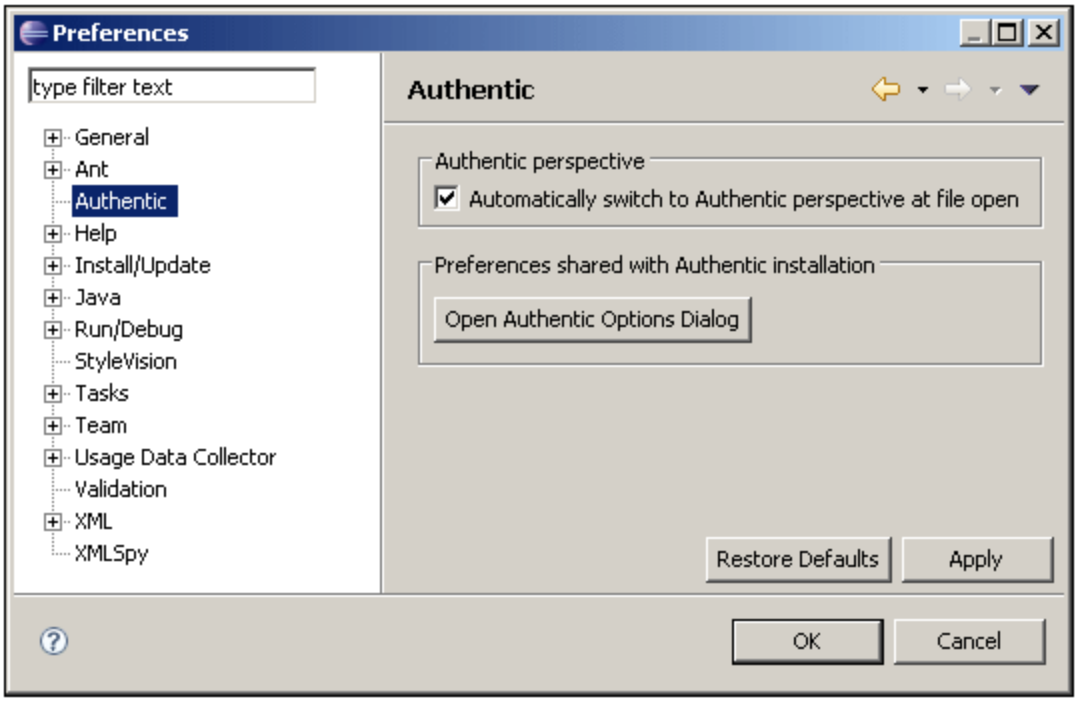


The empty window or the active document will now have the Authentic perspective. This is how the user switches the perspective via the menu. To access a perspective faster from another perspective, the required perspective can be listed in the **Open Perspective** submenu, above the **Other** item; this setting is in the customization dialog (*see below*).

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

Setting preferences for the Authentic perspective

The preferences of a perspective include: (i) a setting to automatically change the perspective when a file of an associated filetype is opened (*see above*), and (ii) options for including or excluding individual Authentic Desktop toolbars. To access the Preferences dialog (*screenshot below*), select the command **Window | Preferences**.



In the list of perspectives in the left pane, select Authentic, then select the required preferences. Finish by clicking **OK**.

Customizing the Authentic 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 the perspective active, and select the command **Window | Customize Perspective**.

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.

In the Commands tab, you can add command groups. To display the commands in a command group, select the required command group from among the available command groups (displayed in the Command Groups pane). The commands in this group are displayed in a tree in the right-hand side pane, ordered hierarchically in the menu in which it will appear. If you wish to include the command group, check its check box.

Click **OK** to complete the customization and for the changes to take effect.

Authentic Desktop toolbar buttons
Two Authentic Desktop-related buttons are created automatically in the toolbar (*screenshot below*).

Thes are for: (i) opening the Authentic Desktop Help, and (ii) editing Athentic documents.

11 User Reference

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

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

11.1 File Menu

The **File** menu contains commands relevant to manipulating files, in the order common to most Windows software products.

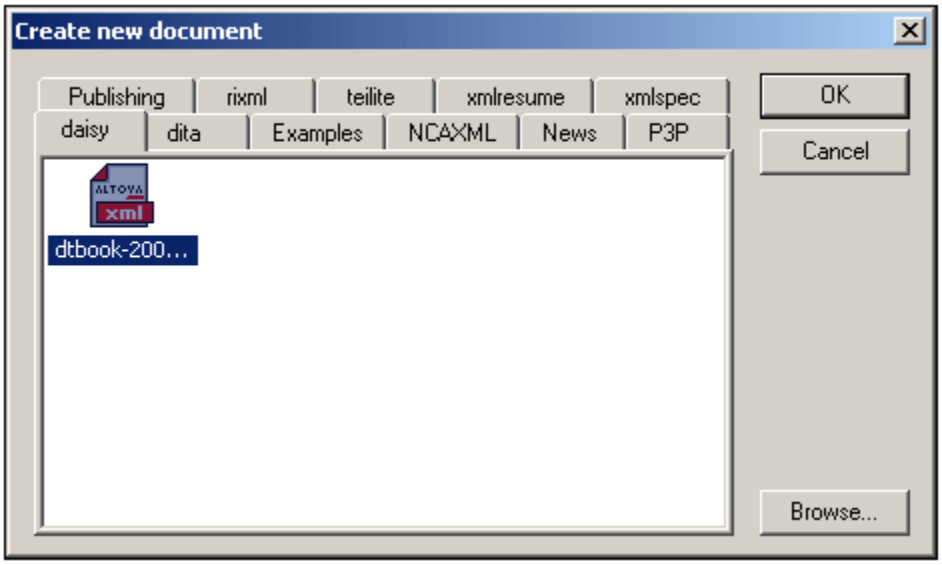
In addition to the standard New, Open, Save, Print, Print Setup, and Exit commands, Authentic Desktop offers a range of XML- and application-specific commands.

11.1.1 New

Ctrl+N

The **New** command is used to create a new document.

Assigning a StyleVision Power Stylesheet when creating a new document
When a new XML document is created, you must associate a StyleVision Power Stylesheet (`.sps` file) to view the document in Authentic View. When you click **New**, the Create New Document dialog (*shown below*) appears.



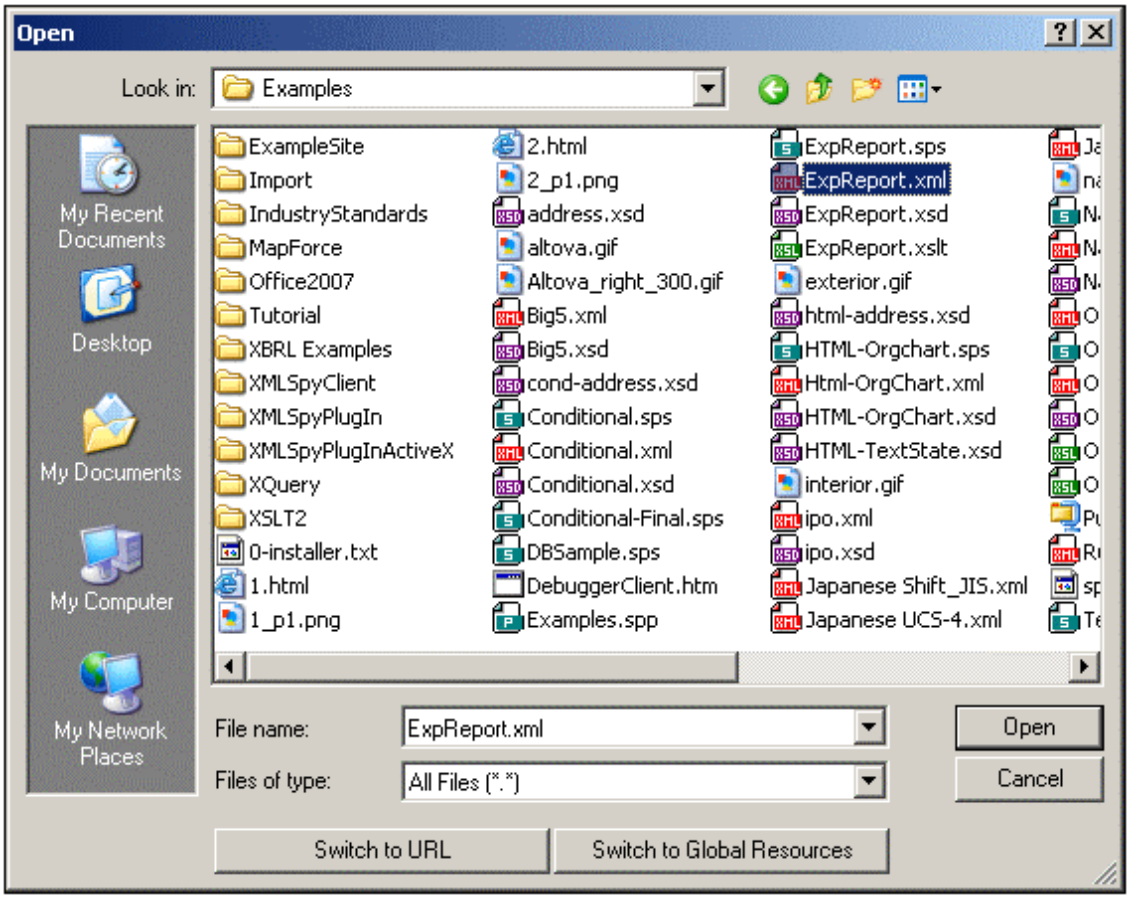
You can browse for the required StyleVision Power Stylesheet in the folder tabs displayed in the New dialog. Alternatively, you can click the **Browse** button to navigate for and select the StyleVision Power Stylesheet. The tabs that appear in the New dialog correspond to folders in the `sps/Template` folder of your application folder.

11.1.2 Open

Ctrl+O

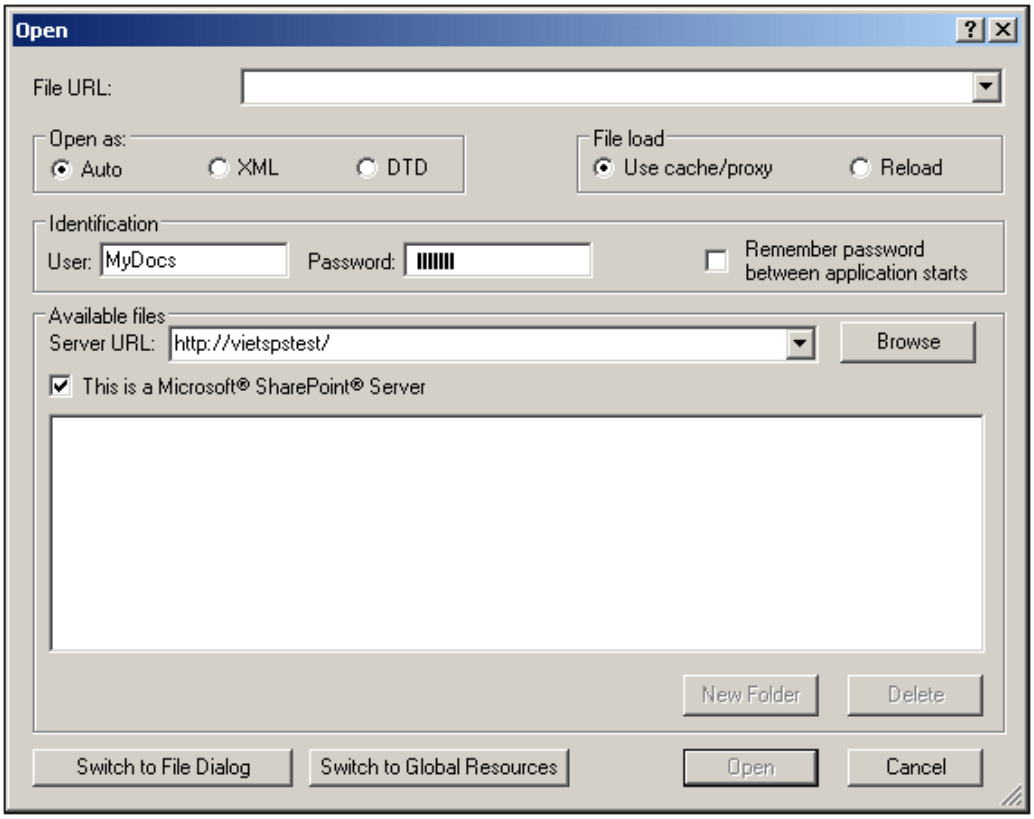
The **Open** command pops up the familiar Windows Open dialog, and allows you to open any XML-related document or text document. In the Open dialog, you can select more than one file to open. Use the Files of Type combo box to restrict the kind of files displayed in the dialog box. (The list of available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

Selecting 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*). Select the **Switch to URL** or **Switch to Global Resource** to go to one of these selection processes.

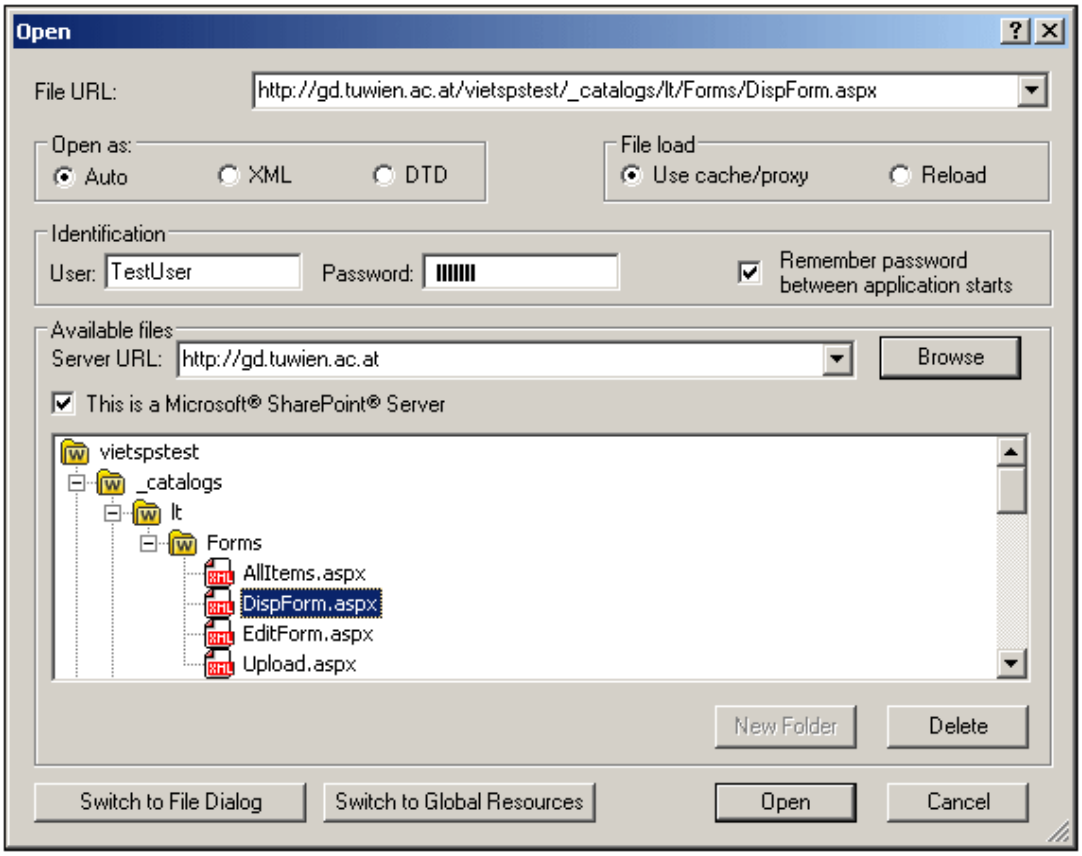


Selecting files via URLs
To select a file via a URL, do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open dialog (*screenshot below*).



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 (*screenshot above*). The **Open** button only becomes active at this point.

6. Click the **Open** button to load the file. The file you open appears in the main window.

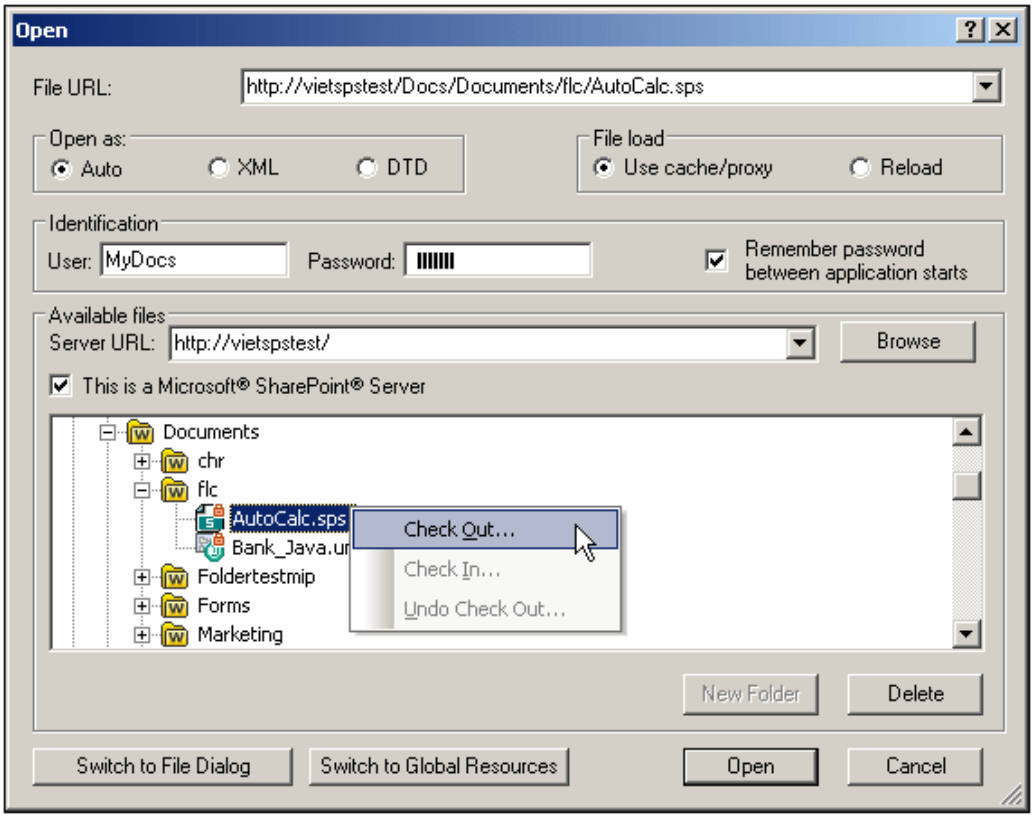
Note: The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.

Note: To give you more control over the loading process, 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 click the file tab in the Main Window of your application (*screenshot below*).

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

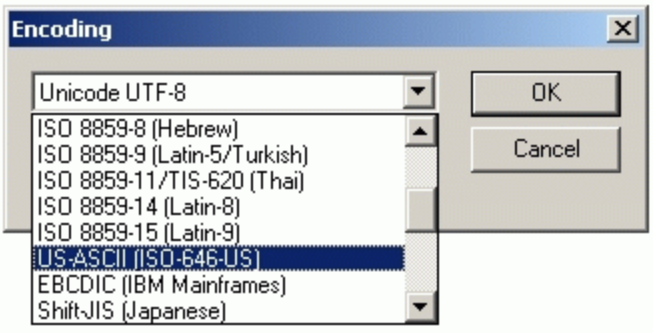
Opening and saving files via Global Resources
To open or save a file via a global resources, click **Switch to Global Resource**. This pops up a dialog in which you can select the global resource. These dialogs are described in the section, Using Global Resources. For a general description of Global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

11.1.3 Reload

The **Reload** command allows you to reload open documents. This is useful if an open document has been modified outside Authentic Desktop. If a modification occurs, Authentic Desktop asks whether you wish to reload the file. If you reload, then any changes you may have made to the file since the last save will be lost. This option can be changed in the Options dialog (Tools | Options).

11.1.4 Encoding

The **Encoding** command lets you view the current encoding of the active document (XML or non-XML) and to 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 in use before, the encoding specification in the XML declaration will be adjusted 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 tab of the Options dialog.

Note: When saving a document, Authentic Desktop automatically checks the encoding specification and opens a dialog box if it cannot recognize the encoding entered by the user. Also, if your document contains characters that cannot be represented in the selected encoding, you will get a warning message when you save your file.

11.1.5 Close, Close All, Close All But Active

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.

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.

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

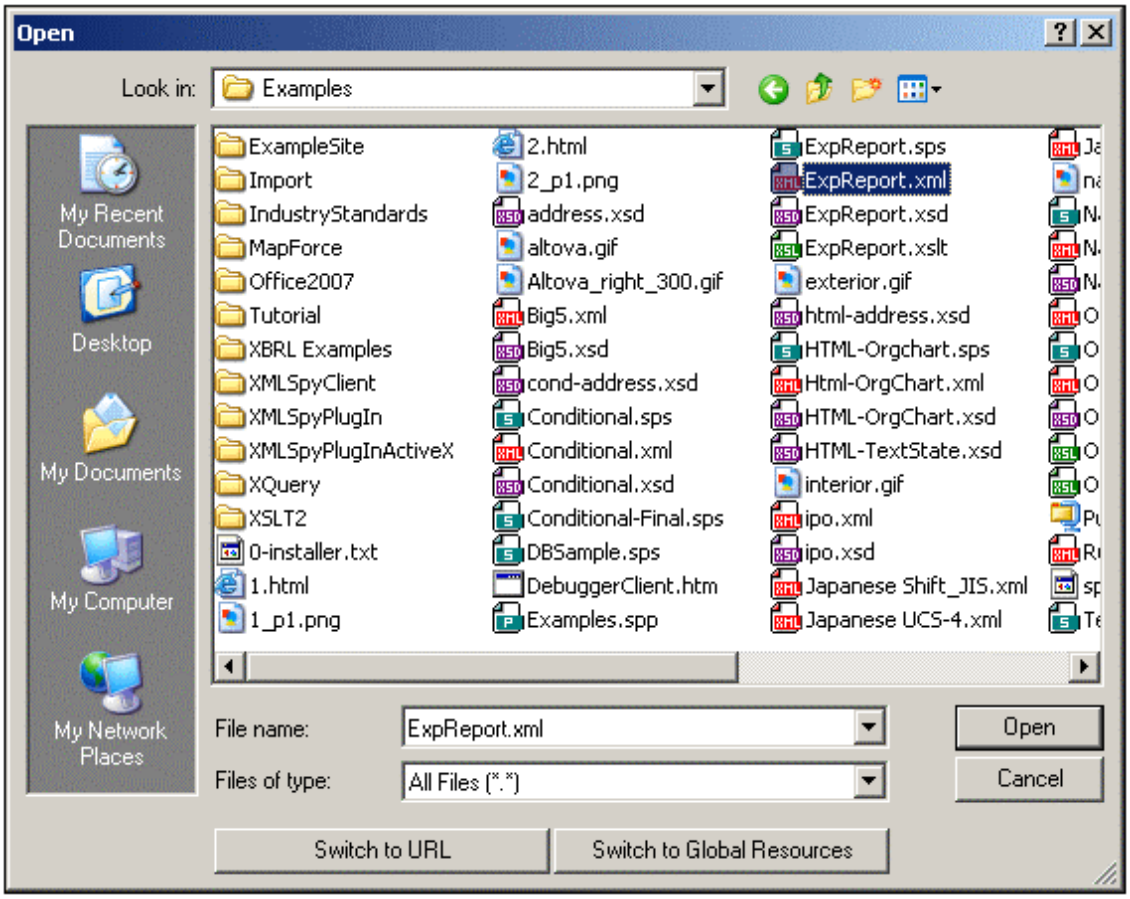
11.1.6 Save, Save As, Save All

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

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.

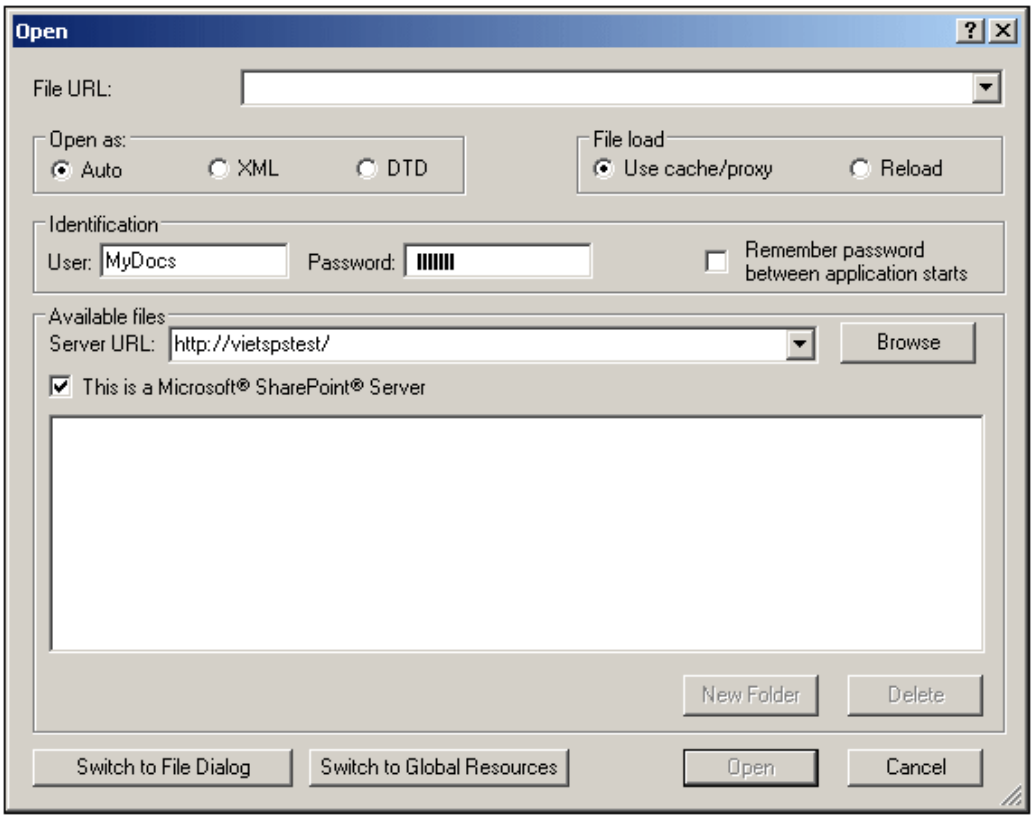
The **Save All** command saves all modifications that have been made to any open documents. The command is useful if you edit multiple documents simultaneously. If a document has not been saved before (for example, after being newly created), the Save As dialog box is presented for that document.

Selecting 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*). Select the **Switch to URL** or **Switch to Global Resource** to go to one of these selection processes.

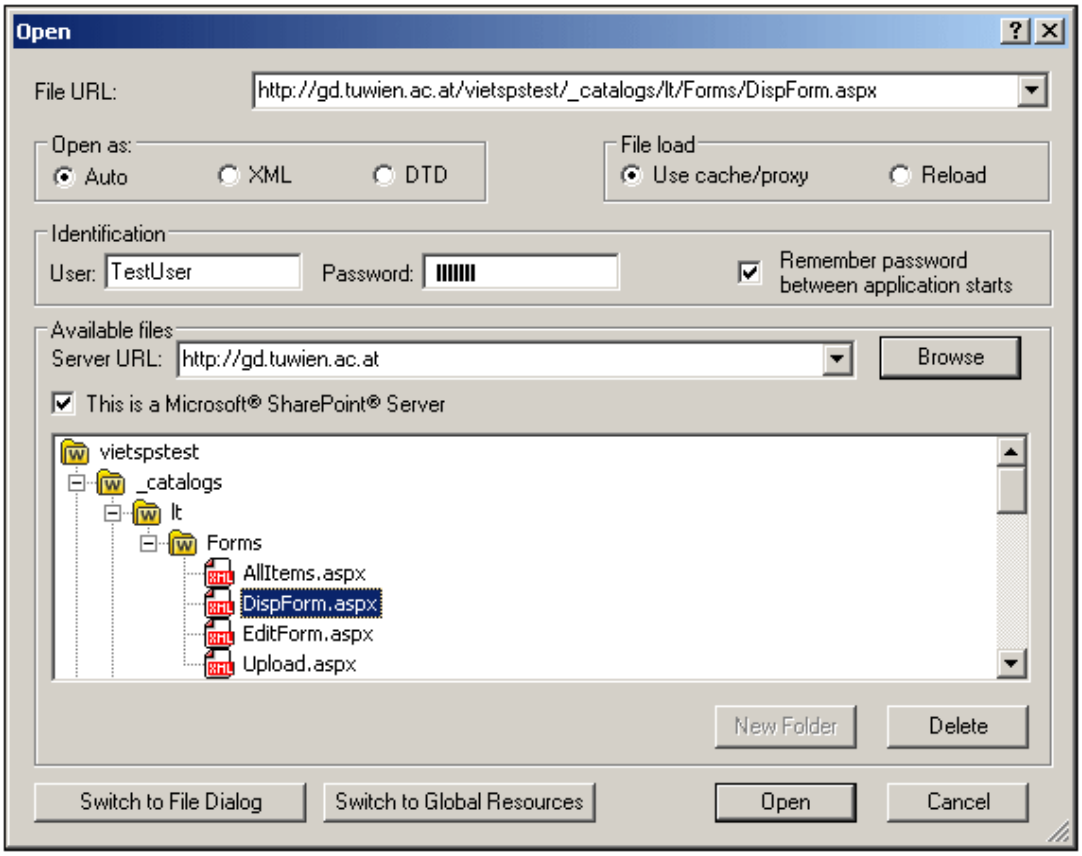


Selecting files via URLs
To select a file via a URL, do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open dialog (*screenshot below*).



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 (*screenshot above*). The **Open** button only becomes active at this point.

6. Click the **Open** button to load the file. The file you open appears in the main window.

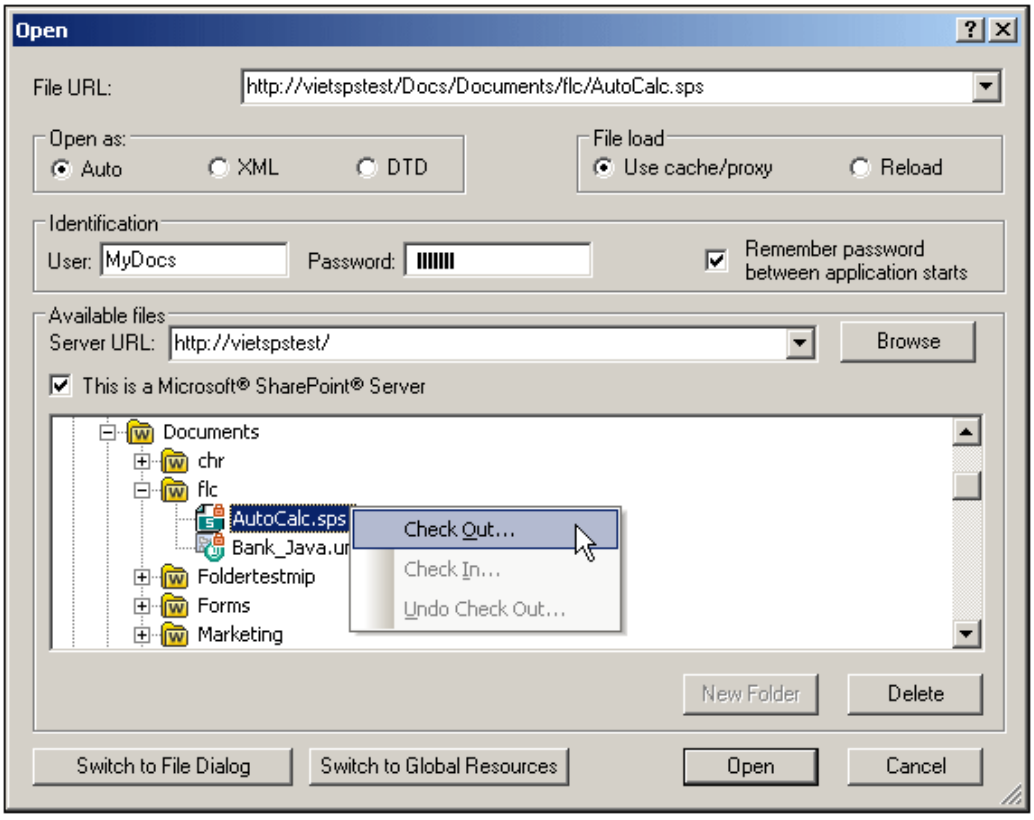
Note: The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.

Note: To give you more control over the loading process, 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 click the file tab in the Main Window of your application (*screenshot below*).

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

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

11.1.7 Send by Mail

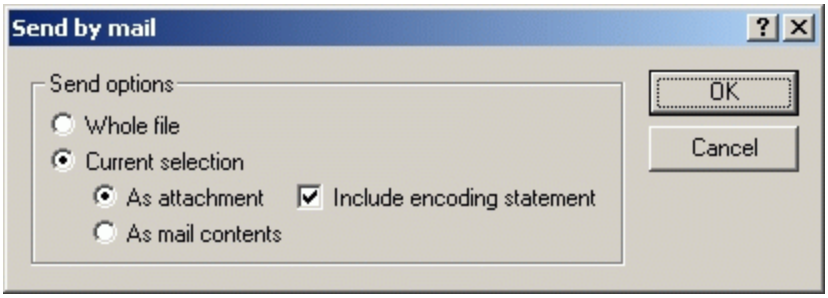
The **Send by Mail...** command lets you send XML document/s or selections from an XML document by e-mail. You can do any of the following:

- Send an XML document as an attachment or as the content of an e-mail.
- Send a selection in an XML document as an attachment or as the content of an e-mail.
- Send a group of files (selected in the Project Window) as an attachment to an e-mail.
- Send a URL (selected in the Project Window) as an attachment or as a link.

Please note: To use this function you must have a MAPI compliant e-mail system.

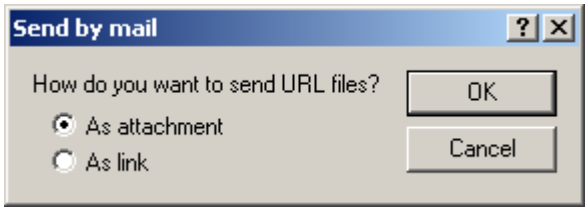
Sending documents and document fragments
To send an XML document:

1. Make that document the active document in the Main Window. If you wish to send a selection or a group of files from a project, make the selection in the document or select the required files in the Project.
2. Click **Send by Mail...**. The following dialog opens:



3. Make the required choices and click **OK**. The selected documents/contents/URLs are attached to the e-mail or content is inserted into the e-mail.

Sending URLs by mail
To send one or more URLs, select the URLs in the Project Window, and click **Send by Mail...**. The following dialog opens:



Select how the URL is to be sent and click **OK**.

11.1.8 Print

 **Ctrl+P**

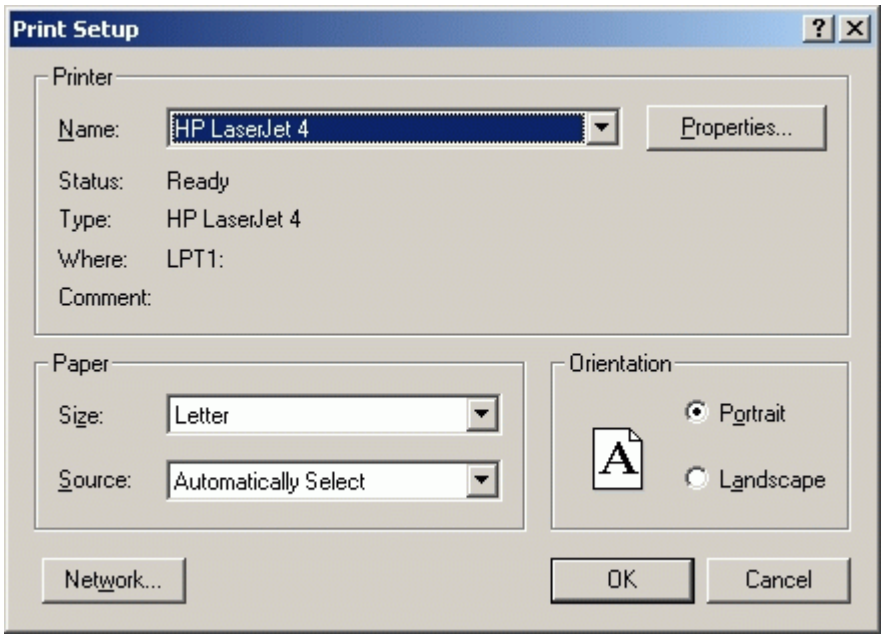
The **Print** command opens the Print dialog box, in which you can select printer options. The currently active document, as seen in the current view, can then be printed.

11.1.9 Print Preview, Print Setup

The **Print Preview** command opens the Print dialog box. Click the **Preview** button to display a print preview of the currently active document. In Print Preview mode, the Print Preview toolbar at top left of the preview window provides print- and preview-related options.

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

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.



The screenshot above shows the Print Setup dialog in which an HP LaserJet 4 printer attached to a parallel port (LPT1) is selected.

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**.

11.1.10 Recent Files, Exit

The **File** menu displays 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.

The **Exit** command is used to quit Authentic Desktop. If you have any open files with unsaved changes, you are prompted to save these changes. Authentic Desktop also saves modifications to program settings and information about the most recently used files.

11.2 Edit Menu

The **Edit** menu contains commands for editing documents in Authentic Desktop.

11.2.1 Undo, Redo

The **Undo** (**Ctrl+Z**) 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.

The **Redo** (**Ctrl+Y**) command  allows you to redo previously undone commands, thereby giving you a complete history of work completed. You can step back and forward through this history using the Undo and Redo commands.

11.2.2 Cut, Copy, Paste, Delete

The **Cut** (**Shift+Del** or **Ctrl+X**) command  copies the selected text or items to the clipboard and deletes them from their present location.

The **Copy** (**Ctrl+C**) 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.

The **Paste** (**Ctrl+V**) command  inserts the contents of the clipboard at the current cursor position.

The **Delete** (**Del**) command  deletes the currently selected text or items without placing them in the clipboard.

11.2.3 Select All

Ctrl+A

The **Select All** command selects the contents of the entire document.

11.2.4 Find, Find Next

The **Find** command (**Ctrl+F**)  pops up 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 What text box or use the combo box to select from one of the last 10 search criteria, and then specify the options for the search.

The **Find Next** command (**F3**)  repeats the last Find command to search for the next occurrence of the requested text.

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.

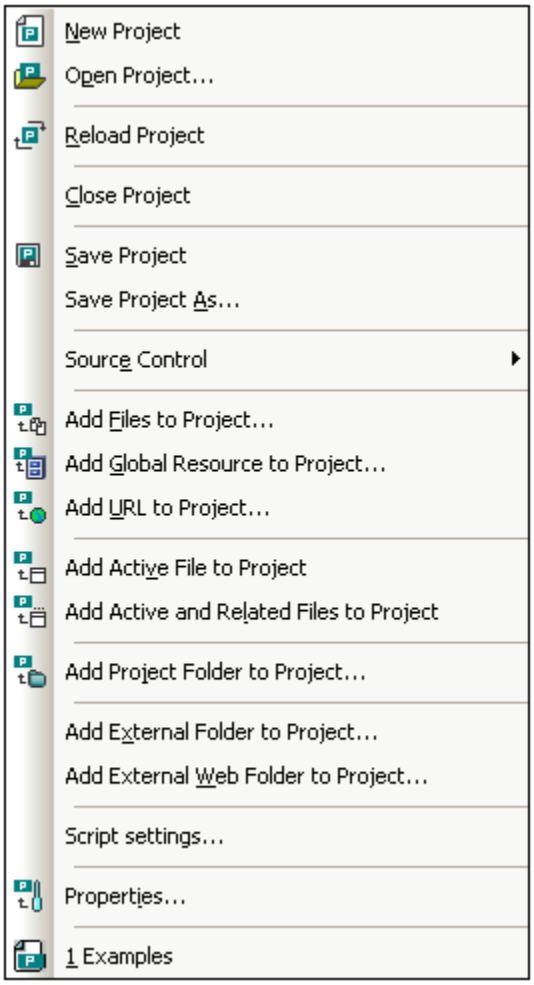
11.2.5 Replace

 **Ctrl+H**

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

11.3 Project Menu

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



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

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

```
| -Folder1
|      |
|      | -Folder2
|             |
|             | -Folder3
|                    |
|                    | -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.

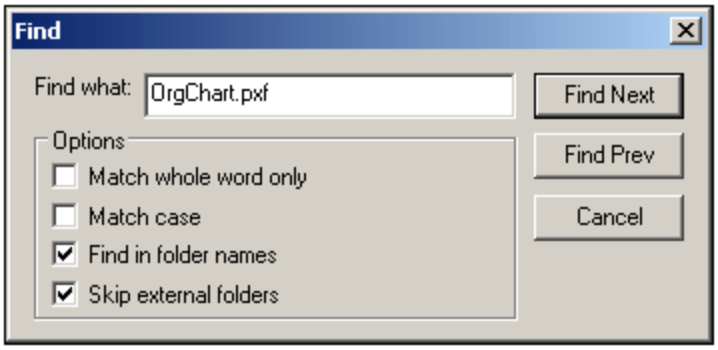
Drag-and-drop

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 can be included or excluded from the search. External folders are actual folders on the system or network, as opposed to project folders, which are created within the project and not on the system.

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

buttons in the Find dialog.

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

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

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

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

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

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

11.3.1 New Project

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

11.3.2 Open Project

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

11.3.3 Reload Project

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

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

11.3.4 Close Project

The **Close Project** command **closes** the active project. If the project has been modified, you will be asked whether you want to save the project first. When a project is modified in any way, an asterisk is added to the project name in the Project Window.

11.3.5 Save Project, Save Project As

The **Save Project** command **saves** the current project. You can also save a project by making the project window active and clicking the  icon.

The **Save Project As** command **saves** the current project with a new name that you can enter when prompted for one.

11.3.6 Source Control

Authentic Desktop supports Microsoft SourceSafe and other compatible repositories. The Source Control Systems supported by Authentic Desktop are listed in the section Supported Source Control Systems. How to install theses systems is described in the section, Installing Source Control Systems. This section describes the commands in the **Project | Source Control** submenu, which are used to work with the Source Control System from within Authentic Desktop.

Overview of the Source Control feature
The mechanism for placing files in a Authentic Desktop project under source control is as follows:

1. In Authentic Desktop, an application project folder containing the files to be placed under source control is created. Typically, the application project folder will correspond to a local folder in which the project files are located. The path to the local folder is referred to as the local path.
2. In the source control system's database (also referred to as source control or repository), a folder is created that will contain the files to be placed under source control.
3. Application project files are added to source control using the command **Project | Source Control | Add to Source Control**.
4. Source control actions, such as checking in to, checking out from, and removing files from source control, can be carried out by using the commands in the **Project | Source Control** submenu. The commands in this submenu are listed in the sub-sections of this section.

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

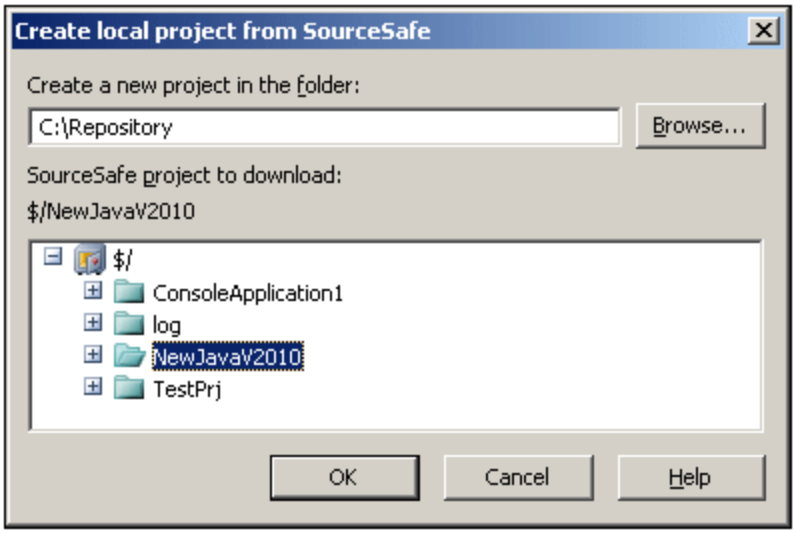
Note: Note that a Source Control project is not the same as an Authentic Desktop project. Source Control projects are directory-dependent, while Authentic Desktop projects are logical constructions without direct directory dependence.

For additional information, see the section, Source Control.

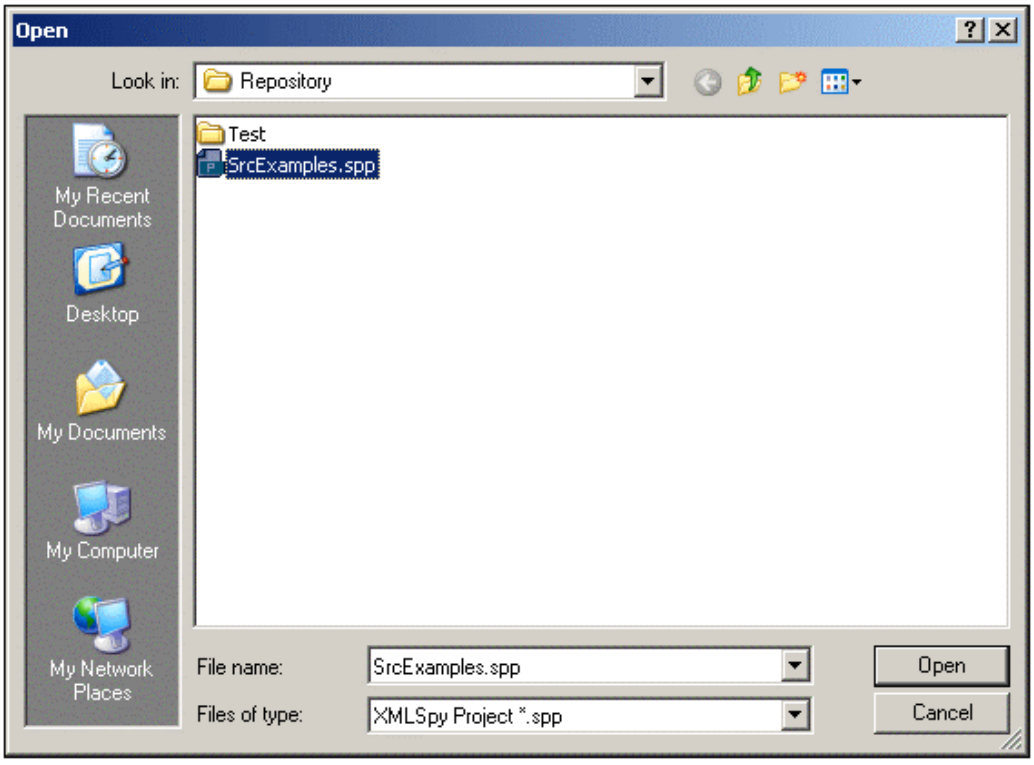
Open from Source Control

The **Open from Source Control** command creates an existing source control project locally. This command is enabled only if an Authentic Desktop project has previously been added to source control using the menu option Add to Source Control.

1. Select **Project | Source Control | Open from Source Control**. Enter your login details in the Login dialog that appears.
2. The Create Local Project from SourceSafe dialog box appears (*screenshot below*). Select the folder to contain the local project. This folder becomes the Working Folder or Checkout Folder (in the screenshot below: `C:\Repository`).



3. Select the source control project you want to download. In the screenshot above, the source control project folder, `NewJavaV2010`, has been selected. Files in the source control project folder will be downloaded to the local folder. If the local folder you have specified does not exist, a dialog box opens prompting you to create it. Click **Yes** to confirm the new folder.
4. The Open dialog now pops up (*screenshot below*).



5. Click the application project file you wish to create and click **Open**. The selected application project will open in Authentic Desktop, and the file is placed under source control.

Source control symbols

Folders and files 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.

Enable Source Control

The **Enable Source Control** command allows you to enable or disable source control for a Authentic Desktop project. Selecting this option on any file or folder, enables/disables source control for the whole project.

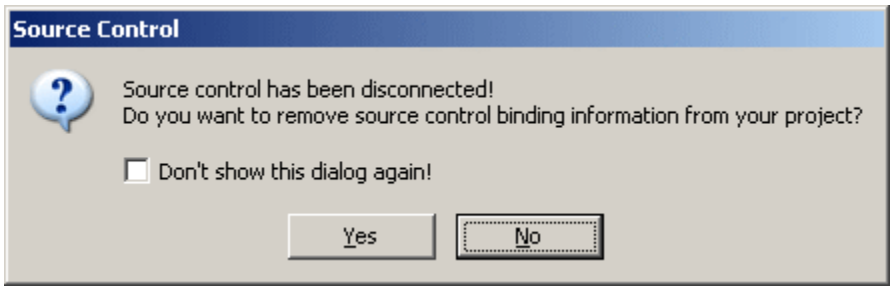
Enabling Source Control for a project

To enable source control for a project, do the following:

1. Click on any file or folder in the Project window.
2. Select the menu option **Project | Source Control | Enable Source Code Control**. The previous check in/out status of the various files are retrieved and displayed in the Project window.

Disabling Source Control for a project

To disable source control for a project, select the menu option **Project | Source Control | Enable Source Control**. You are now prompted if you want to remove the binding information from the project (*see screenshot below*).



To provisionally disable source control for the project select **No**. To permanently disable source control for the project select **Yes**.

Get Latest Version

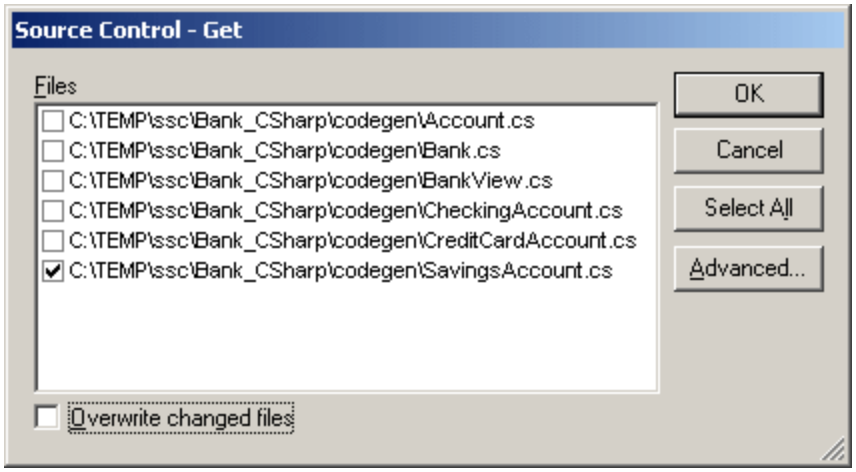
The **Get Latest Version** command retrieves and places the latest source control version of the selected file(s) in the working directory. The files are retrieved as read-only and are not checked out. No further options are available when using this command.

To get the latest version of a file, do the following:

1. Select the file(s) you want to get the latest version of in the Model Tree.
2. Select **Project** | **Source Control** | **Get Latest Version**.

Get

The **Get** command retrieves read-only copies of the selected files and places them in the working folder. By default, the files are not checked-out for editing. To get a file, select it in the Project window (multiple files can be selected) and then select the **Get** command. This pops up the Get dialog (*screenshot below*). Check the files you want to get.



Overwrite changed files check box
Overwrites those files that have been changed locally with those from the source control database.

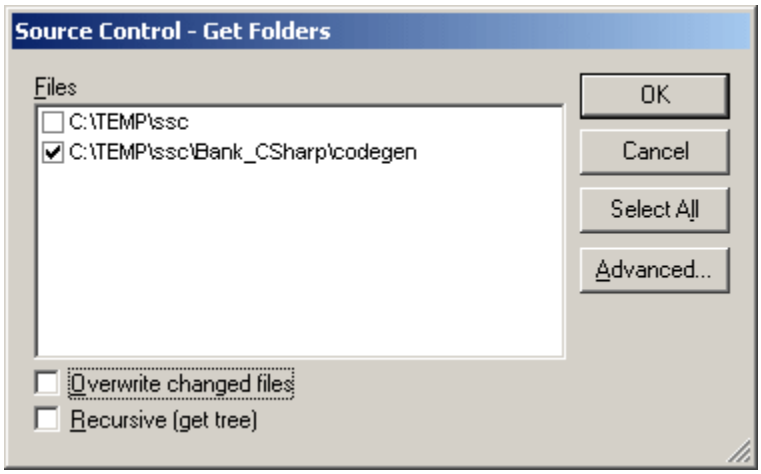
Select All
Selects all the files in the list box.

Advanced
Allows you to define the *Replace writable* and *Set timestamp* options in the respective combo boxes.

The *Make writable* check box removes the read-only attribute of the retrieved files.

Get Folders

The **Get Folders** command retrieves read-only copies of files in the selected folders and places them in the working folder. By default, the files are not checked-out for editing. To get a folder, s elect it in the Project window and then select the **Get Folders** command. This pops up the Get Folders dialog (*screenshot below*). Check the foldersyou want to get.



Overwrite changed files check box
Overwrites those files that have been changed locally with those from the source control database.

Recursive (get tree) check box
Retrieves all files of the folder tree below the selected folder.

Select All
Selects all the files in the list box.

Advanced
Allows you to define the *Replace writable* and *Set timestamp* options in the respective combo boxes.

The *Make writable* check box removes the read-only attribute of the retrieved files.

Check Out

The **Check Out** command checks out the latest version of the selected files and places writable copies in the working directory. The files are flagged as checked out for all other users. To check out files, do the following:

1. Select the file or folder you want to check out in the Model Tree.
2. Select **Project | Source Control | Check Out**.
3. In the Check Out dialog that pops up, select the files to check out, then click **OK**.

Note the following points:

- You can change the number of files to check out by activating the individual check boxes in the Files list box.
- The *Checkout local version* option checks out only the local versions of files, not those from the source control database.
- The following items can be checked out: (i) single files (in the Project window, click the required file to select it; use **Ctrl+Click** to select multiple files, use **Ctrl+Click**); (ii) folders (in the Project window, click the required folder to select it; use **Ctrl+Click** to select multiple folders).
- Checked out files and folders are indicated with a red check mark.

Advanced
Allows you to define the *Replace writable* and *Set timestamp* options in the respective combo boxes.

The *Make writable* check box removes the read-only attribute of the retrieved files.

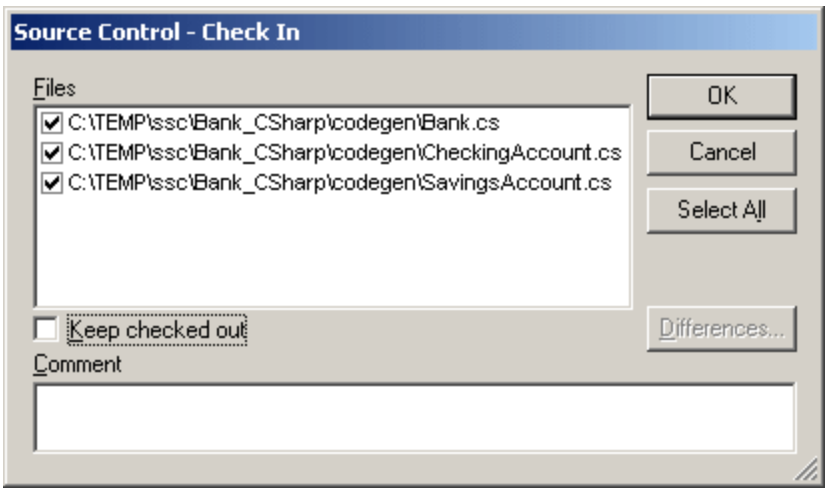
Source control symbols
Folders and files display certain symbols, the meanings of which are given below.

	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.

Check In

The Check In command checks in previously checked out files (that is, your locally updated files) and places them in the source control database. To check in files, do the following:

1. Select the files in the Model Tree.
2. Select **Project | Source Control | Check In**.
3. In the Check In dialog that pops up, select the files to check in, then click **OK**.



Note the following points:

- As a shortcut for the Check In command, right-click a checked out item in the project window and select **Check In** from the context menu.
- The following items can be checked in: (i) single files (in the Project window, click the required file to select it; use **Ctrl+Click** to select multiple files, use **Ctrl+Click**); (ii) folders (in the Project window, click the required folder to select it; use **Ctrl+Click** to select multiple folders).
- The lock symbol denotes that the file/folder is under source control, but is currently not checked out.
- The **Differences** button is enabled when a line in the Files pane is selected. Clicking it enables you to see differences between two file versions.

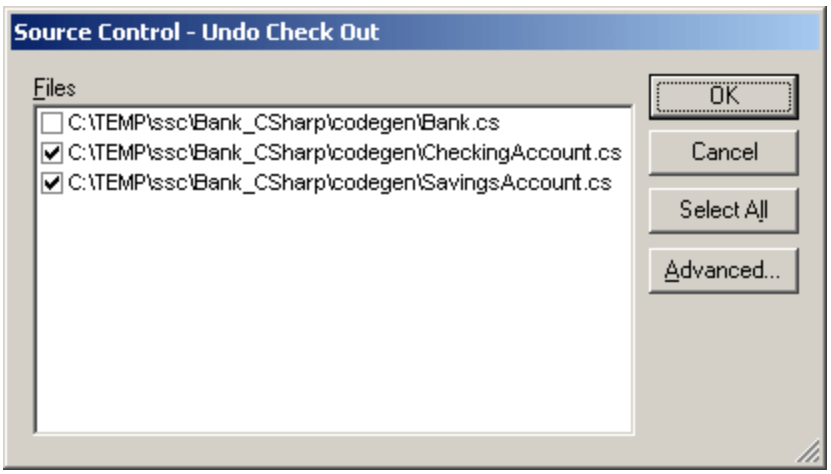
Source control symbols
Folders and files display certain symbols, the meanings of which are given below.

	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.

Undo Check Out

The **Undo Check Out** command rejects changes made to previously checked out files (that is, your locally updated files) and retains the old files from the source control database. To undo a check out, do the following:

1. Select the files in the Project window.
2. Select **Project | Source Control | Undo Check Out**.
3. In the Undo Check Out dialog that pops up, select the files for which check out should be undone, then click **OK**.



Check out of the following items can be undone: (i) single files (in the Project window, click the required file to select it; use **Ctrl+Click** to select multiple files, use **Ctrl+Click**); (ii) folders (in the Project window, click the required folder to select it; use **Ctrl+Click** to select multiple folders).

Advanced
Allows you to define the *Replace writable* and *Set timestamp* options in the respective combo boxes.

The *Make writable* check box removes the read-only attribute of the retrieved files.

Source control symbols
Folders and files display certain symbols, the meanings of which are given below.

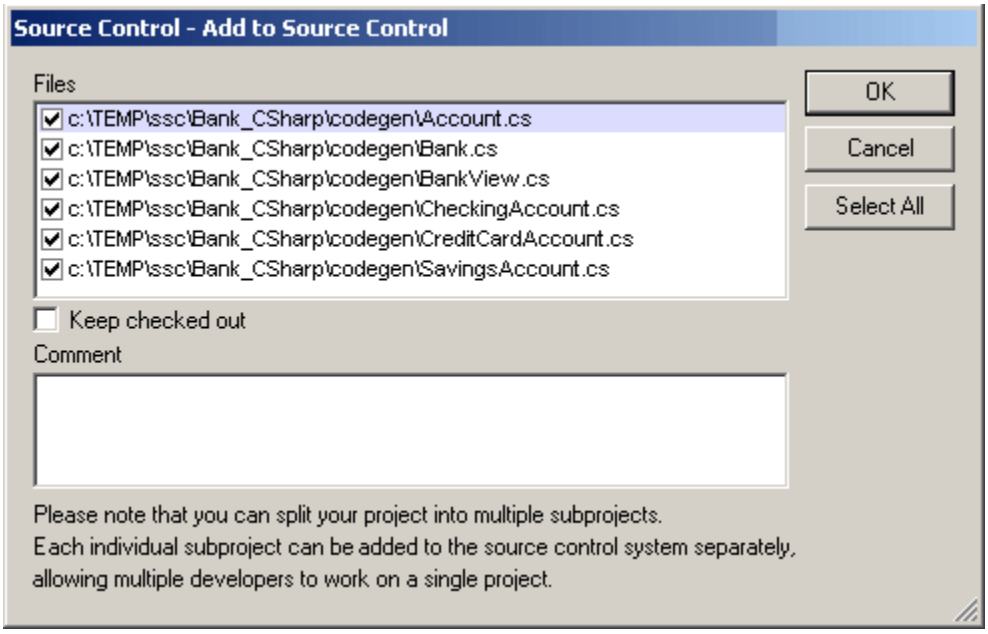
	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Add to Source Control

The **Add to Source Control** command adds the selected files in a project folder to the source control database and places them under source control. To add files to source control, do the following:

1. In the Project window, click a file and select the menu option **Project | Source Control | Add to Source Control**. If you select a folder, the files in that folder will be added to source control.
2. In the Add to Source Control dialog that pops up, ensure that the files to be added are

checked. If the file to be added should be kept checked out, check the *Keep checked out* check box. Then click **OK**.



The lock symbol now appears next to each of the files placed under source control. If the files are checked out they will be shown with a red check symbol.

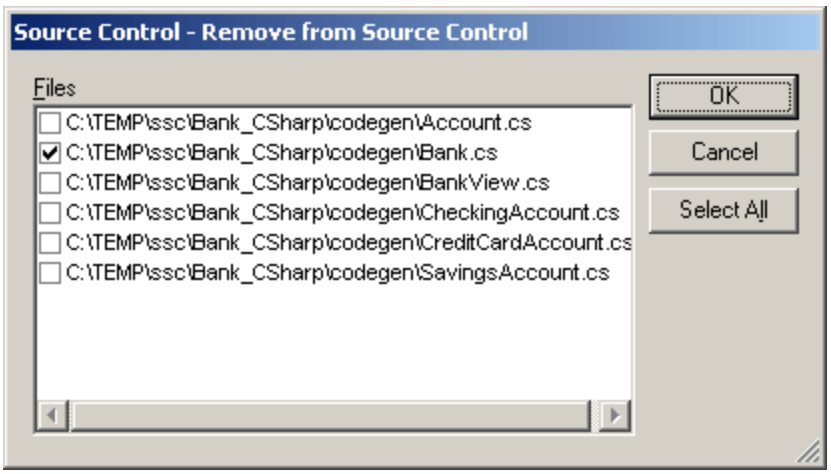
Source control symbols
Folders and files display certain symbols, the meanings of which are given below.

	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.

Remove from Source Control

The **Remove from Source Control** command removes previously added files from the source control database. These files will be visible in the Project window but cannot be checked in or out. Use the **Add to Source Control** command to place them back under source control. To remove files from the source control provider, do the following:

1. In the Project window, select the files you wish to remove.
2. Select **Project | Source Control | Remove from Source Control**.
3. In the Remove from Source Control dialog that pops up, select the files to remove, then click **OK**.

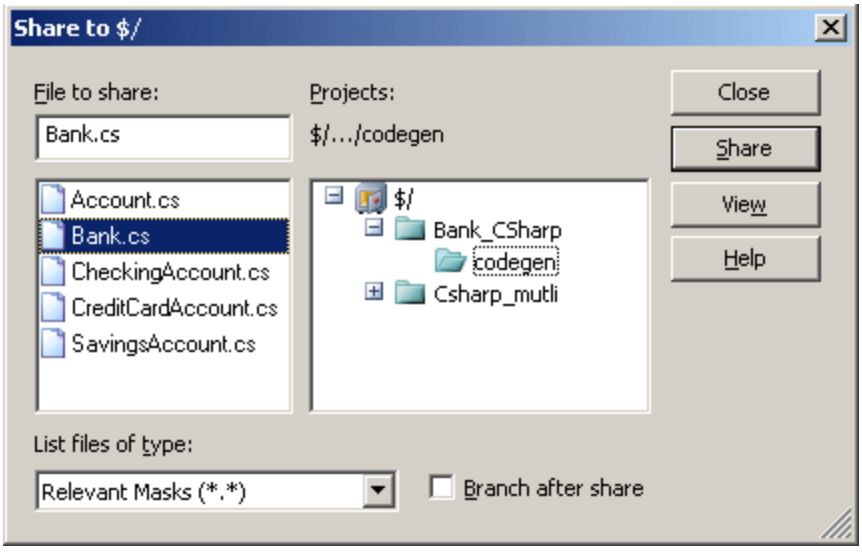


The following items can be removed from source control: (i) single files (in the Project window, click the required file to select it; use **Ctrl+Click** to select multiple files, use **Ctrl+Click**); (ii) folders (in the Project window, click the required folder to select it; use **Ctrl+Click** to select multiple folders).

Share from Source Control

To use the **Share from Source Control** command you must have the Check in/out rights to the project you are sharing from. To share a file from source control, do the following:

1. In the Project window, select the file you want to share and select **Project | Source Control | Share from Source Control**.
2. In the *Projects* list, select the project folder that contains the file you want to share.



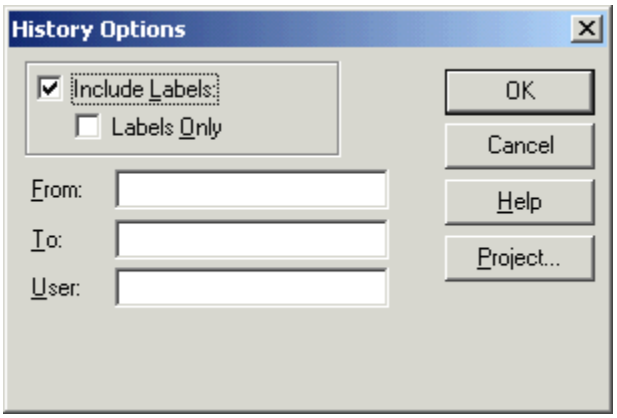
3. In the *Files to share* list box, select the file you want to share and click the **Share** button. The file will be removed from the *Files to share* list.
4. Click the **Close** button to continue.

Branch after share
The *Branch After Share* option shares the file and creates a new branch to create a separate version.

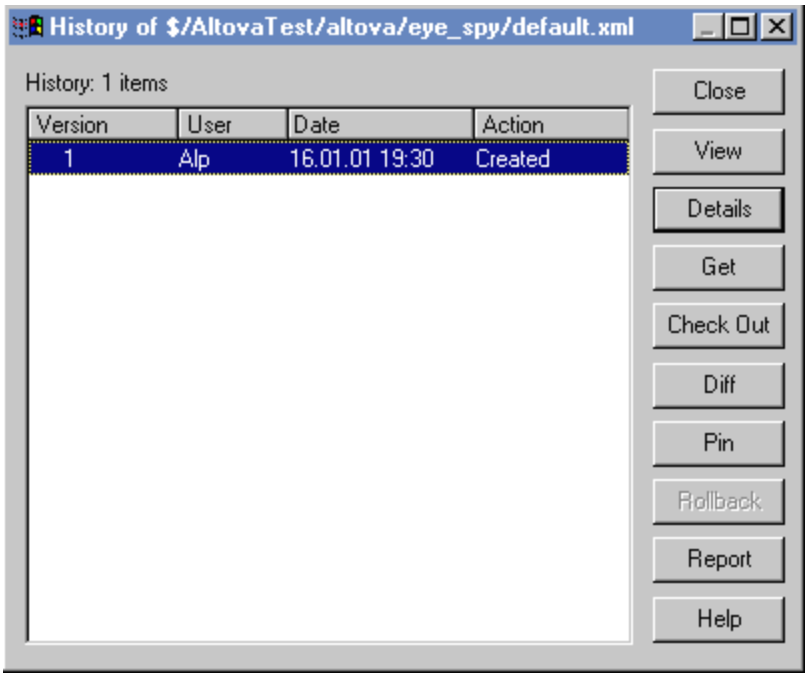
Show History

The **Show History** command displays the history of a file under source control and allows you to view detailed history info of previous versions of a file, view differences and retrieve previous versions of the file. To show the history of a file, do the following:

1. Click on the file in the Project window.
2. Select the menu option **Project | Source control | Show History**. A dialog box prompting for more information may appear (this example uses Visual Source-Safe).



3. Select the appropriate entries and confirm with **OK**.



This dialog box provides various way 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 dialog provide the following functionality:

- ***Close:*** closes this dialog box.
- ***View:*** opens a further dialog box in which you can select the type of viewer with which you wish to see the file.
- ***Details:*** opens a dialog box in which you can see the properties of the currently

active file.

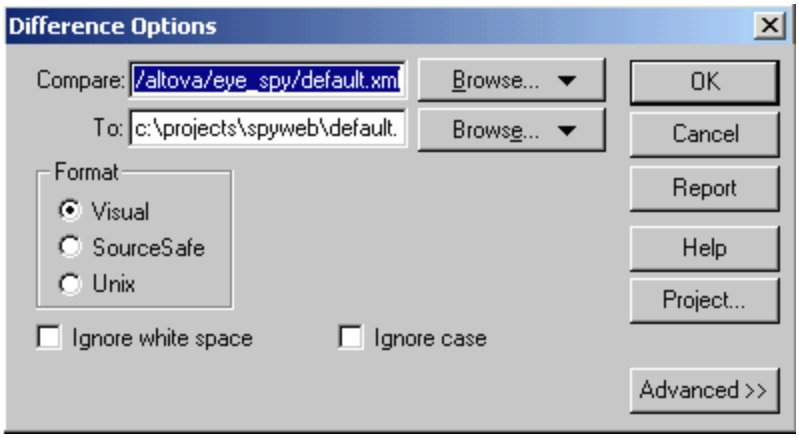
- ***Get:*** allows you to retrieve one of the previous versions of the file in the version list and place it in the working directory.
- ***Check Out:*** allows you to check out a previous version of the file.
- ***Diff:*** opens the Difference options dialog box, which allows you to define the difference options when viewing the differences 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 which you can send to the printer, file, or clipboard.
- ***Help:*** opens the online help of the source control provider plugin.

Show Differences

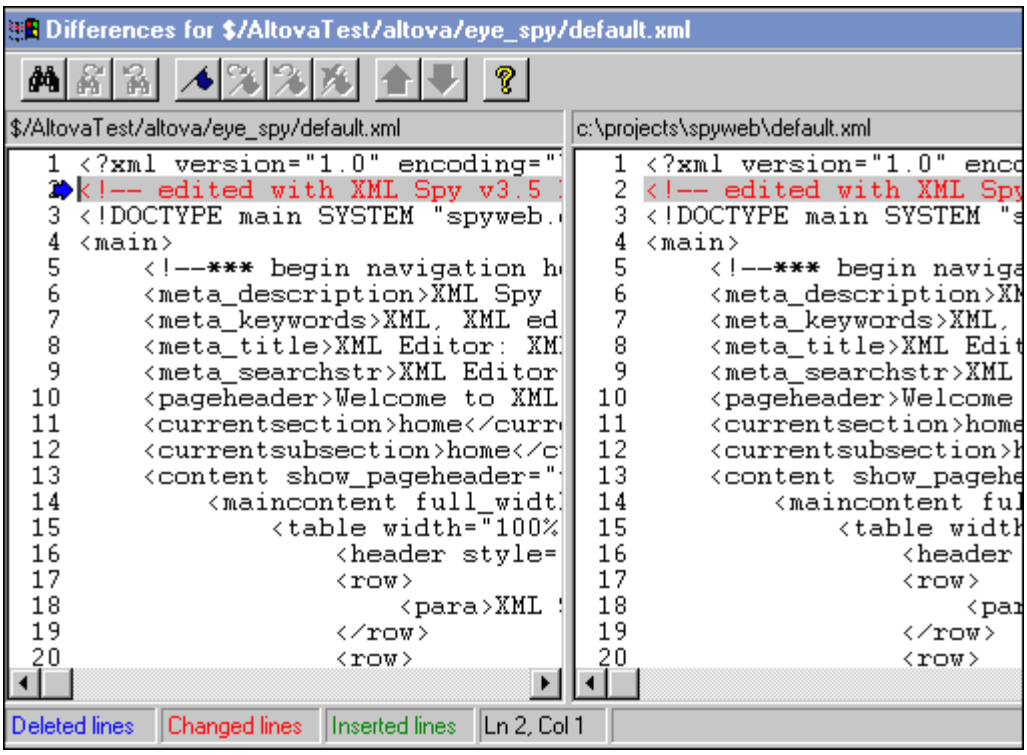
The **Show Differences** command displays the differences between the file currently in the source control repository and the **checked in/out** file of the same name in the working directory. If you have "pinned" one of the files in the history dialog box, then the pinned file will be used in the "Compare" text box. Any two files can be selected using the Browse buttons.

To show the differences between two files, do the following:

1. Check out a file from your project. Click on the file in the project window.
2. Select the menu option **Project | Source control | Show Differences**. A dialog box prompting for more information may appear at this time.



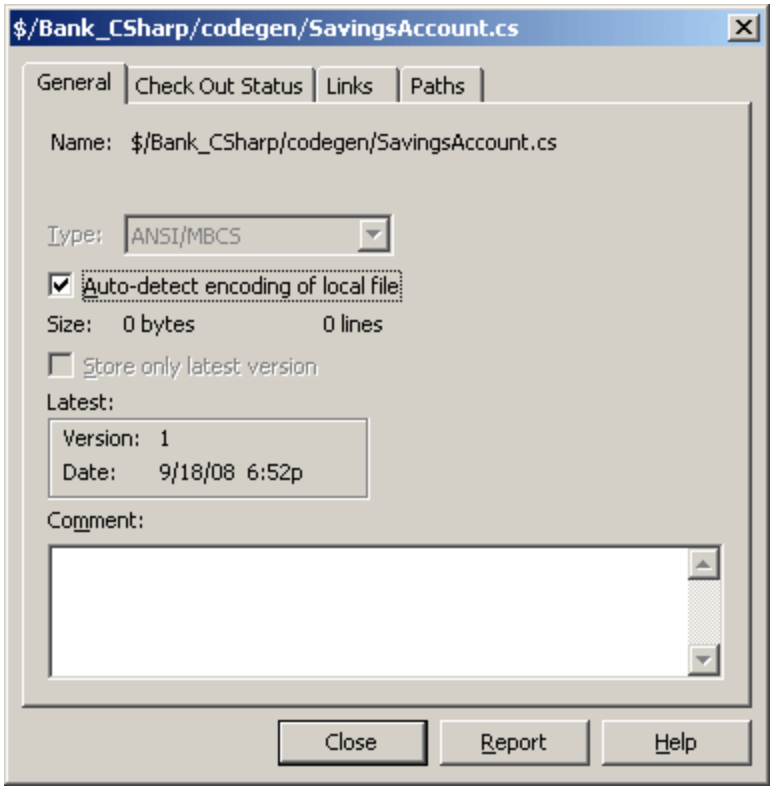
3. Select the appropriate entries and confirm with **OK**.



The differences between the two files are highlighted in both windows (this example uses MS Source-Safe).

Show Properties

The **Show Properties** command displays the properties of the currently selected file (*screenshot below*). The properties displayed depends on the source control provider you use.



Note that this command can only be used on single files.

Refresh Status

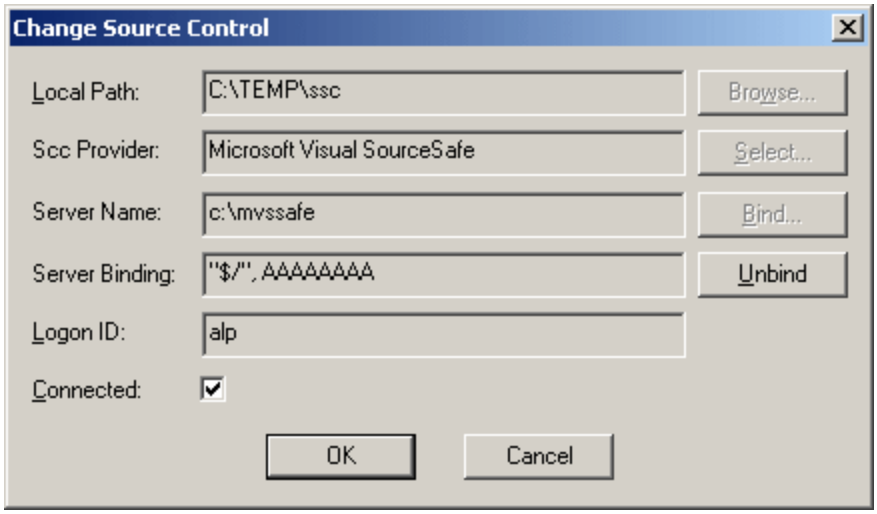
The **Refresh Status** command refreshes the status of all project files independent of their current status.

Source Control Manager

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

Change Source Control

The **Change Source Control** command pops up the Change Source Control dialog box (*screenshot below*), which enables you to change the source control provider that you are using. Click the **Unbind** button first, then click the **Select** button to select the new source control provider.



After you have selected the source control provider, you will need to create a binding between the local folder and the folder on the source control server. The local folder is that entered in the Local Path text box. To select a folder on the source control server, click the **Bind** button. This enables you to log on to the server and then navigate to the required folder on the source control server. Click **OK** when done.

11.3.7 Add Files to Project

The **Project** | **Add Files to Project** command adds files to the current project. Use this command to add files to any folder in your project. You can either select a single file or any group of files (using **Ctrl+ click**) in the Open dialog box. If you are adding files to the project, they will be distributed among the respective folders based on the File Type Extensions defined in the Project Properties dialog box.

11.3.8 Add Global Resource to Project

The **Project** | **Add Global Resource to Project** command pops up the Choose Global Resource dialog, in which you can select a global resource of file or folder type to add to the project. If a file-type global resource is selected, then the file is added to the appropriate folder based on the File Type Extensions defined in the Project Properties dialog box. If a folder-type global resource is selected, that folder will be opened in a file-open dialog and you will be prompted to select a file; the selected file is added to the appropriate folder based on the File Type Extensions defined in the Project Properties dialog box. For a description of global resources, see the Global Resources section in this documentation.

11.3.9 Add URL to Project

The **Project** | **Add URL to Project** command adds a URL to the current project. URLs in a project cause the target object of the URL to be included in the project. Whenever a batch operation is performed on a URL or on a folder that contains a URL object, Authentic Desktop retrieves the document from the URL, and performs the requested operation.

11.3.10 Add Active File to Project

The **Project** | **Add Active File to Project** command adds the active file to the current project. If you have just opened a file from your hard disk or through an URL, you can add the file to the current project using this command.

11.3.11 Add Active And Related Files to Project

The **Project** | **Add Active and Related Files to Project** command adds the currently active XML document and all related files to the project. When working on an XML document that is based on a DTD or Schema, this command adds not only the XML document but also all related files (for example, the DTD and all external parsed entities to which the DTD refers) to the current project.

Please note: Files referenced by processing instructions (such as XSLT files) are not considered to be related files.

11.3.12 Add Project Folder to Project

The **Project** | **Add Project Folder to Project** command adds a new folder to the current project. Use this command to add a new folder to the current project or a sub-folder to a project folder. You can also access this command from the context-menu when you right-click on a folder in the project window.

Note: A project folder can be dragged and dropped into another project folder or to any other location in the project. Also, a folder can be dragged from Windows (File) Explorer and dropped into any project folder.

Note: Project folders are green, while external folders are yellow.

11.3.13 Add External Folder to Project

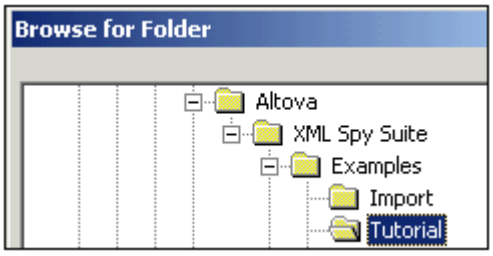
The **Project** | **Add External Folder to Project** command adds a new external folder to the current project. Use this command to add a local or network folder to the current project. You can also access this command from the context-menu when you right-click a folder in the project window.

Note: External folders are yellow, while project folders are green.

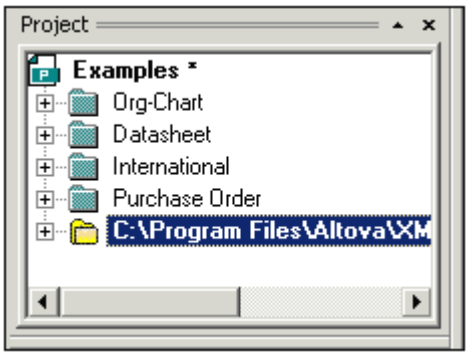
Note: Files contained in external folders cannot be placed under source control.

Adding external folders to projects
To add an external folder to the project:

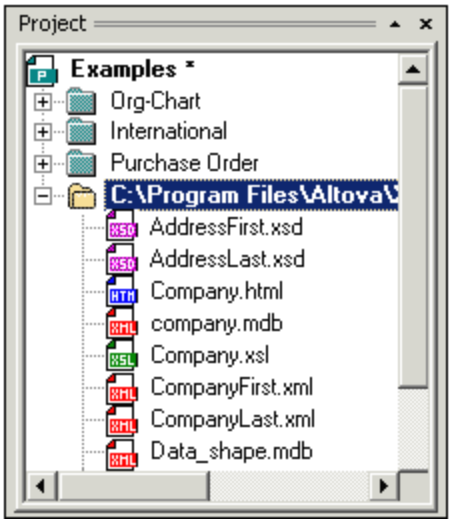
1. Select the menu option **Project | Add External Folder to Project**.
2. Select the folder you want to include from the Browse for Folder dialog box, and click **OK** to confirm.



The selected folder now appears in the project window.

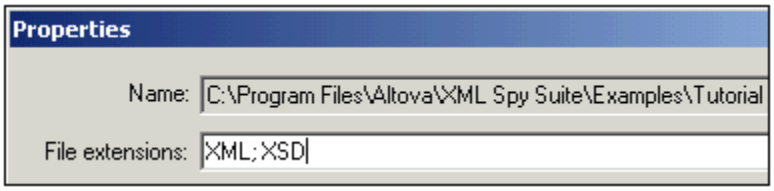


3. Click the plus icon to view the folder contents.

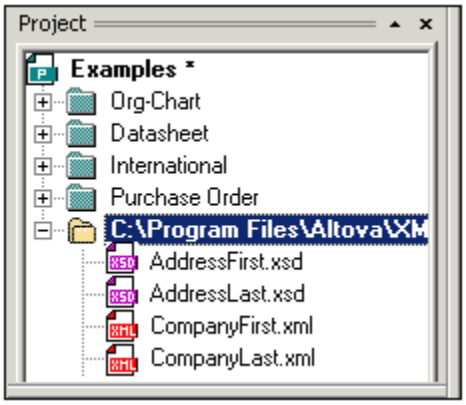


Filtering contents of folders
To filter the contents of the folder:

1. Right-click the local folder, and select the popup menu option **Properties**. This opens the Properties dialog box.



2. Click in the **File extensions** field and enter the file extensions of the file types you want to see. You can separate each file type with a **semicolon** to define multiple types (XML and Schema XSDs in this example).
3. Click **OK** to confirm.



The Project window now only shows the XML and XSD files of the tutorial folder.

Validating external folders
To validate and check an external folder for well-formedness:

1. Select the file types you want to see or check from the external folder,
2. Click the folder and click the **Check well-formedness** or **Validate** icon

(hotkeys **F7** or **F8**). All the files visible under the folder are checked. If a file is malformed or invalid, then this file is opened in the Main Window, allowing you to edit it.

3. Correct the error and run the validation process once more to recheck.

Updating a project folder

You might add or delete files in the local or network directory at any time. To update the folder view, right-click the external folder, and select the popup menu option **Refresh external folder**.

Deleting external folders and files in them

Select an external folder and press the **Delete** key to delete the folder from the Project window. Alternatively, right-click the external folder and select the **Delete** command. Each of these actions only deletes the external folder from the Project window. The external folder is not deleted from the hard disk or network.

To delete a file in an external folder, you have to delete it physically from the hard disk or network. To see the change in the project, refresh the external folder contents (right-click the external folder and select **Refresh**).

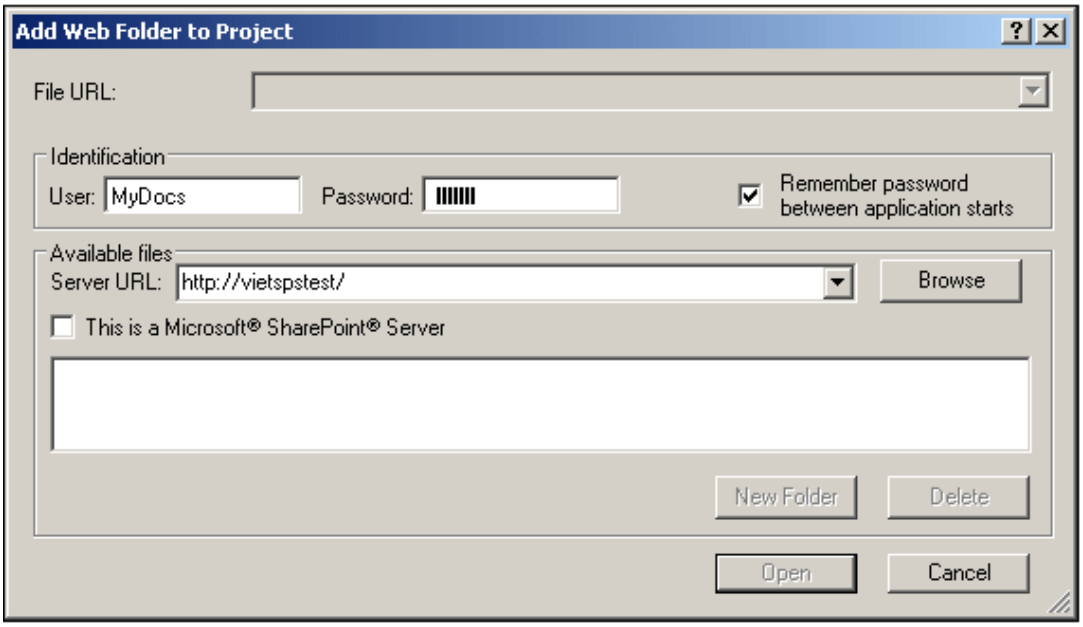
Note: An external folder can be dragged and dropped into a project folder or to any other location in the project (but not into another external folder). Also, an external folder can be dragged from Windows (File) Explorer and dropped into any location in the project window except into another external folder.

11.3.14 Add External Web Folder to Project

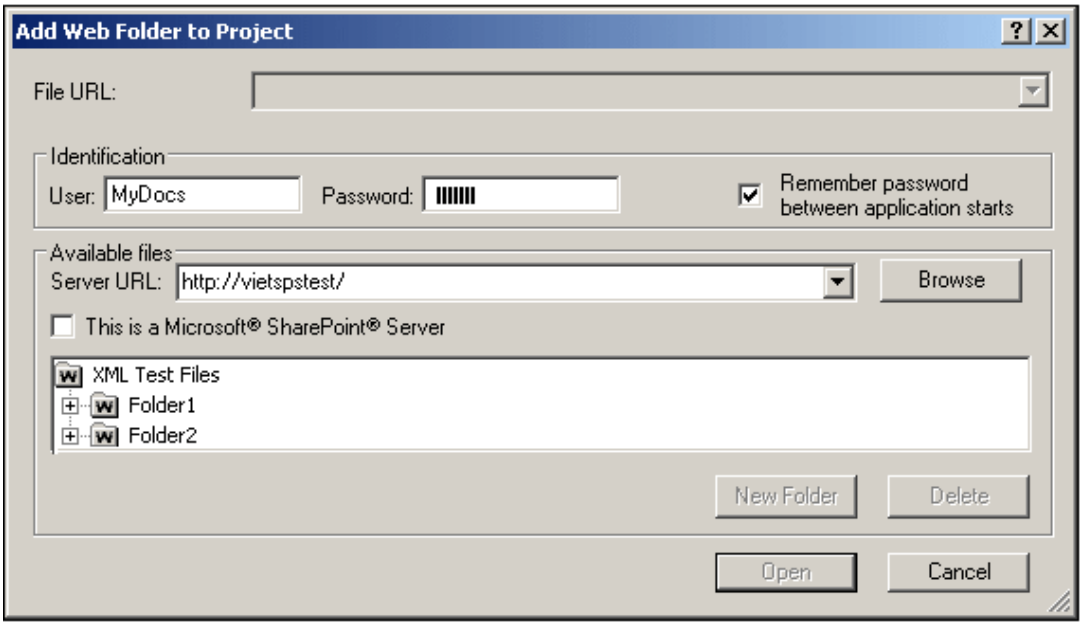
This command adds a new external web folder to the current project. You can also access this command from the context-menu when you right-click a folder in the project window. Note that files contained in external folders cannot be placed under source control.

Adding an external web folder to the project
To add an external web folder to the project, do the following:

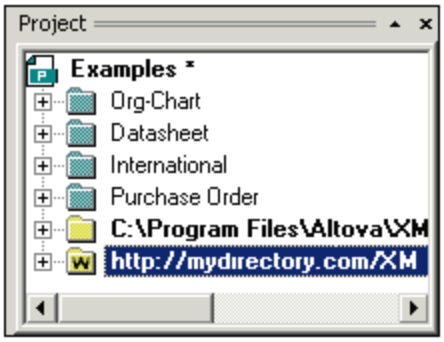
1. Select the menu option **Project | Add External Web Folder to Project**. This opens the Add Web Folder to Project dialog box (*screenshot below*).



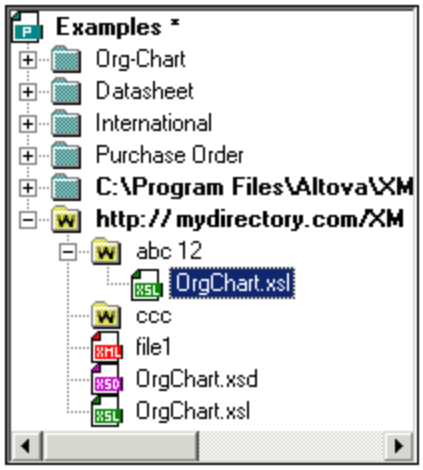
2. Click in the Server URL field and enter the URL of the server URL. If the server is a Microsoft® SharePoint® Server, check this option. See the *Folders on a Microsoft® SharePoint® Server* section below for further information about working with files on this type of server.
3. If the server is password-protected, enter your User ID and password in the *User* and *Password* fields.
4. Click **Browse** to connect to the server and view the available folders.



5. Click the folder you want to add to the project view. The **Open** button only becomes active once you do this. The URL of the folder now appears in the File URL field.
6. Click **Open** to add the folder to the project.



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



Filtering folder contents

To filter the contents of a folder, right-click the folder and select **Properties** from the context menu. In the Properties dialog that pops up, click in the *File Extensions* field and enter the file extensions of the file types you want to see (for example, XML and XSD files). Separate each file type with a semicolon (for example: `xml; xsd; sps`). The Project window will now show that folder only with files having the specified extension.

Validating and checking a folder for well-formedness

To check the files in a folder for well-formedness or to validate them, select the folder and then click the **Check well-formedness**  or **Validate**  icon (hotkeys **F7** or **F8**, respectively). All the files that are visible in the folder are checked. If a file is malformed or invalid, then this file is opened in the main window, allowing you to edit it. Correct the error and restart the process to recheck the rest of the folder. Note that you can select discontinuous files in the folder by holding **Ctrl** and clicking the files singly. Only these files are then checked when you press **F7** or **F8**.

Updating the contents of the project folder

Files may be added or deleted from the web folder at any time. To update the folder view, right-click the external folder and select the context menu option **Refresh**.

Deleting folders and files

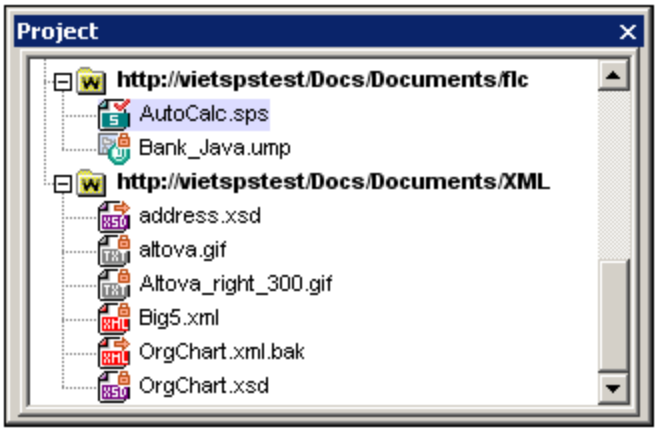
Since it is the Web folder that has been added to the project, it is only the Web folder (and not files within it) that can be deleted from the project. You can delete a Web folder from a project, by either (i) right-clicking the folder and selecting **Delete**, or (ii) selecting the folder and pressing the **Delete** key. This only deletes the folder from the Project view; it does not delete anything on the web server.

Note: Right-clicking a single file and pressing the **Delete** key does not delete a file from the Project window. You have to delete it physically on the server and then refresh the contents of the external folder.

Folders on a Microsoft® SharePoint® Server

When a folder on a Microsoft® SharePoint® Server has been added to a project, files in the folder can be checked out and checked in via commands in the context menu of the file listing in the Project window (*see screenshot below*). To access these commands, right-click the file you wish to work with and select the command you want (**Check Out**, **Check In**, **Undo Check Out**).

The User ID and password can be saved in the properties of individual folders in the project, thereby enabling you to skip the verification process each time the server is accessed.



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

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

The following points should be noted:

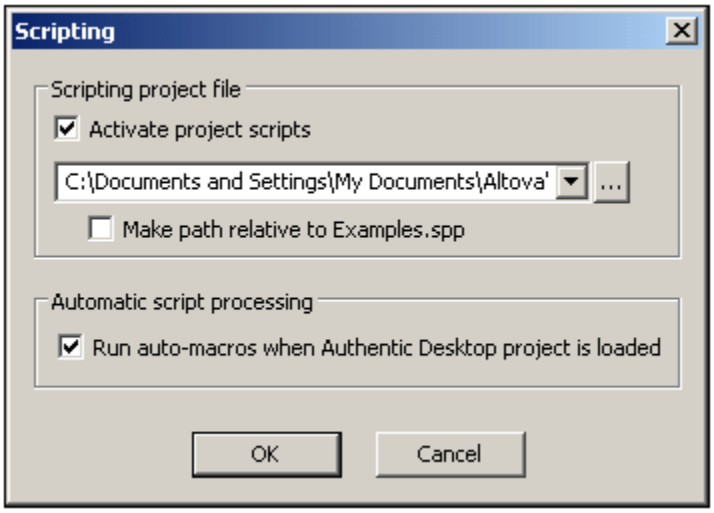
- After you check out a file, you can edit it in your Altova application and save it using **File | Save (Ctrl+S)**.
- You can check-in the edited file via the context menu in the Project window (*see screenshot above*), or via the context menu that pops up when you right-click the file tab in the Main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

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

11.3.15 Script Settings

A scripting project is assigned to an Authentic Desktop project as follows:

1. In the Authentic Desktop GUI, open the required application project.
2. Select the menu command **Project | Script Settings**. The Scripting dialog (*screenshot below*) opens.



3. Check the *Activate Project Scripts* check box and select the required scripting project (`.asprj` file). If you wish to run Auto-Macros when the Authentic Desktop project is loaded, check the *Run Auto-Macros* check box.
4. Click **OK** to finish.

Note: To deactivate (that is, unassign) the scripting project of an Authentic Desktop project, uncheck the *Activate Project Scripts* check box.

11.3.16 Properties

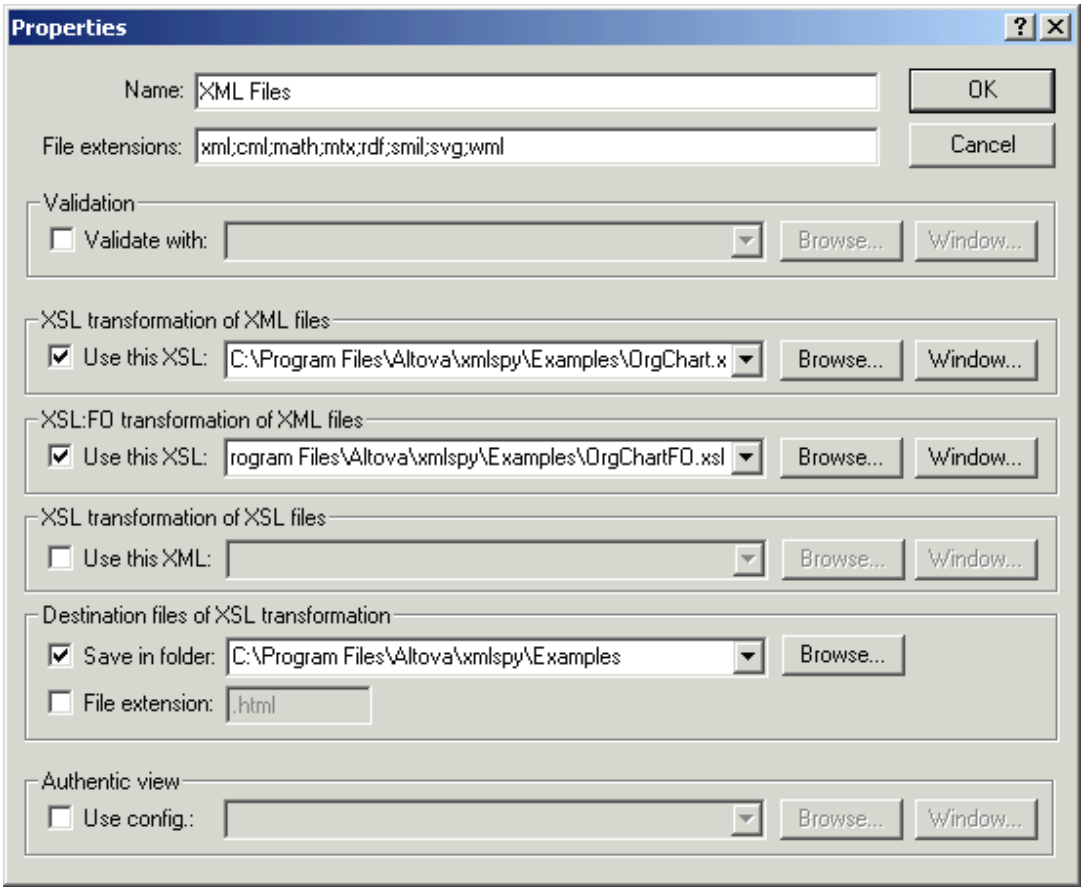
The **Project** | **Project Properties** command lets you define important settings for any of the specific folders in your project.

To define the Project Properties for a folder:

1. Right-click on the folder you want to define the properties for.
2. Select the **Properties...** command from the context menu.

Please note:

If your project file is under source control, a prompt appears asking if you want to check out the project file (*.spp). Click **OK** if you want to edit settings and be able to save them.



The files specified in the **Use this xxx** entry will take precedence over any local assignment directly within the XML file. For example, the `OrgChart.xsl` file (in the **Use this XSL** entry), will always be used when transforming any of the XML files in the **XML Files** folder. Also, such specified files for individual folders take precedence over files specified for ancestor folders.

File extensions

The File extensions help to determine the automatic file-to-folder distribution that occurs when you add new files to the project (as opposed as to one particular folder).

User ID and password for external folders

Among the properties of external folders (including external Web folders) you can save the User ID and password that might be required for accessing the server.

Validate
Define the DTD or Schema document that should be used to validate all files in the current folder (Main Pages in this example).

XSL transformation of XML files
You can define the XSL Stylesheet to be used for XSL Transformation of all files in the folder.

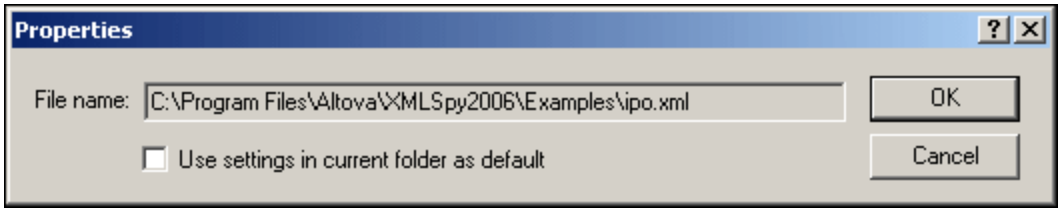
If you are developing XSL Stylesheets yourself, you can also assign an example XML document to be used to preview the XSL Stylesheet in response to an XSL Transformation command issued from the stylesheet document, instead of the XML instance document.

XSL:FO transformation of XML files
You can define the XSL Stylesheet, containing XSL:FO markup, to be used for XSL:FO Transformation of all files in the folder.

Destination files of XSL transformation
For batch XSL Transformations, you can define the destination directory the transformed files should be placed in.

If you have added one file or URL to more than one folder in your project, you can use the Properties dialog to set the default folder whose settings should be used when you choose to validate or transform the file in non-batch mode. To do this, use the **Use settings in current folder as default** check box (*see screenshot*).

To access the Properties dialog and check this check box:
1. Copy an XML file in a project to a different folder.
2. Right-click the copied file in the Project window and select **Properties** from the context menu.



Authentic View
The "Use config." option allows you to select a StyleVision Power Stylesheet (SPS file) when editing XML files using Authentic View, in the current folder. After you have associated the schema, SPS, and XML files with each other, and entered them in a project, changing the location of any of the files could cause errors among the associations.

To avoid such errors, it is best to finalize the locations of your schema, SPS, and XML files before associating them with each other and assigning them to a project.

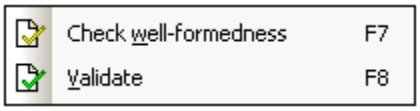
11.3.17 Most Recently Used Projects

This command displays the file name and path for the nine most recently used projects, allowing quick access to these files.

Also note, that Authentic Desktop can automatically open the last project that you used, whenever you start Authentic Desktop. (**Tools | Options | File** tab, Project | Open last project on program start).

11.4 XML Menu

The **XML** menu contains commands commonly used when working with XML documents.

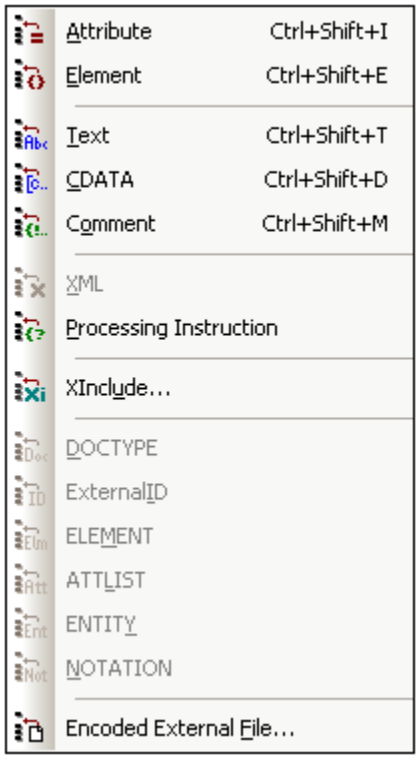


Among the most frequently used XML tasks are checks for the well-formedness of documents and validity of XML documents. Commands for these tasks are in this menu.

11.4.1 Insert

The **XML** | **Insert** command, though enabled in all views, can be used in Grid View only. It has a submenu (*see screenshot*) with which you can insert:

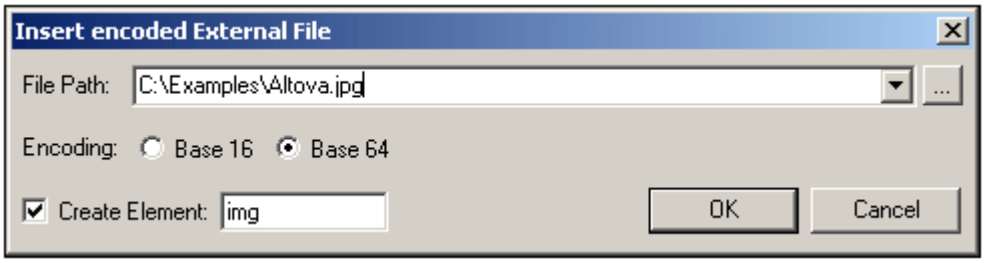
- The XML declaration and node types (Attribute, Element, Text, CDATA, Comment, Processing Instruction) in XML documents;
- DOCTYPE declarations and external DTD declarations in XML documents;
- DTD declarations (ELEMENT, ATTLIST, ENTITY, and NOTATION) in DTD documents and internal DTD declarations of XML documents.



Insert Encoded External File

The **XML** | **Insert | Encoded External File** command is available in Grid View only. It inserts a binary encoded file, such as an image file, as encoded characters. The encoded external file is inserted before the Grid View selection.

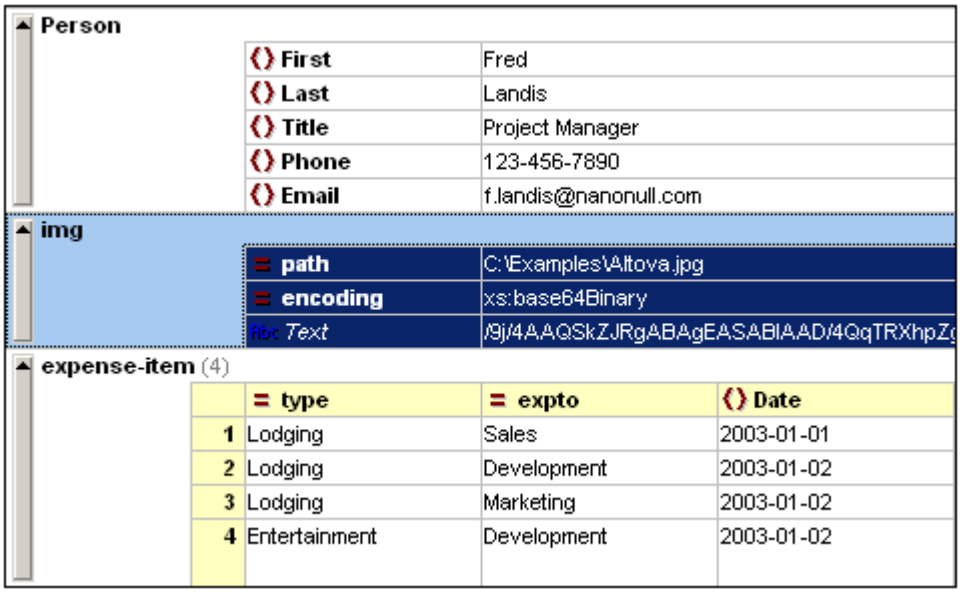
On clicking the command, the Insert Encoded External File dialog (*screenshot below*) pops up. In it you enter the path to the file, select the encoding you want, and specify whether the encoded file is to be inserted in an element or not.



You can browse for or enter the name of the external file to be encoded and embedded. Either a Base-16 or Base-64 encoding must be specified. If you wish to enclose the encoded text in an element, then check the Create Element check box and specify the name of the desired element in the Create Element text box. If the Create Element check box is not checked, then the encoded text will be inserted directly at the cursor location.

On clicking **OK**, the encoded text of the selected file is inserted at the cursor location, with an enclosing element if this has been specified.

The encoded file is inserted in Grid View (*the highlighted element in the screenshot below*).



In Text View, the file will be inserted as below.

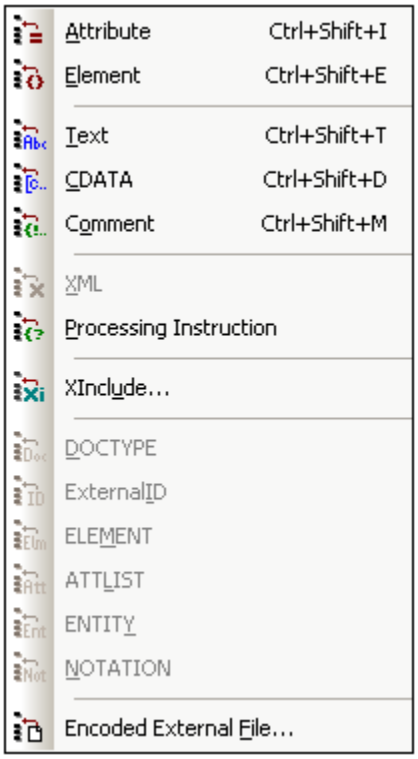
```
<img ext="jpg" encoding="xs:base64Binary">
  iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD/
  //+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4U
  g9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC
</img>
```

The listing above shows the encoded text of a JPG image file. An `img` element was created around the encoded text.

11.4.2 Append

The **XML** | **Append** command, though enabled in all views, can be used in Grid View only. It opens a submenu (*see screenshot*) with which you can append:

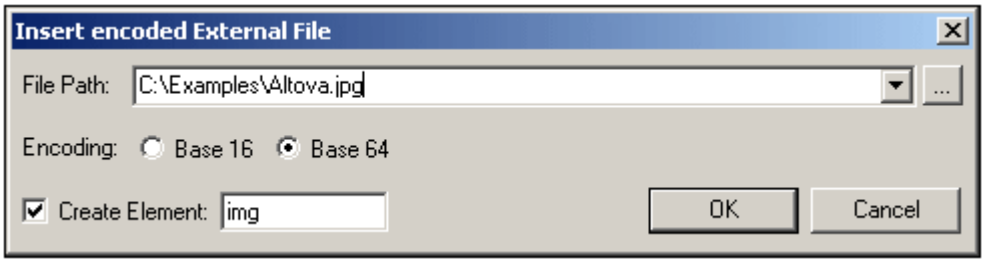
- The XML declaration and node types (Attribute, Element, Text, CDATA, Comment, Processing Instruction) in XML documents;
- DOCTYPE declarations and external DTD declarations in XML documents
- DTD declarations (ELEMENT, ATTLIST, ENTITY, and NOTATION) in DTD documents and internal DTD declarations of XML documents.



Append Encoded External File

The **XML** | **Append | Encoded External File** command is available in Grid View only. It appends a binary encoded file, such as an image file, as encoded characters. The encoded external file is appended after the Grid View selection.

On clicking the command, the Insert Encoded External File dialog (*screenshot below*) pops up. In it you enter the path to the file, select the encoding you want, and specify whether the encoded file is to be inserted in an element or not.



You can browse for or enter the name of the external file to be encoded and embedded. Either a Base-16 or Base-64 encoding must be specified. If you wish to enclose the encoded text in an element, then check the Create Element check box and specify the name of the desired element in the Create Element text box. If the Create Element check box is not checked, then the encoded text will be inserted directly at the cursor location.

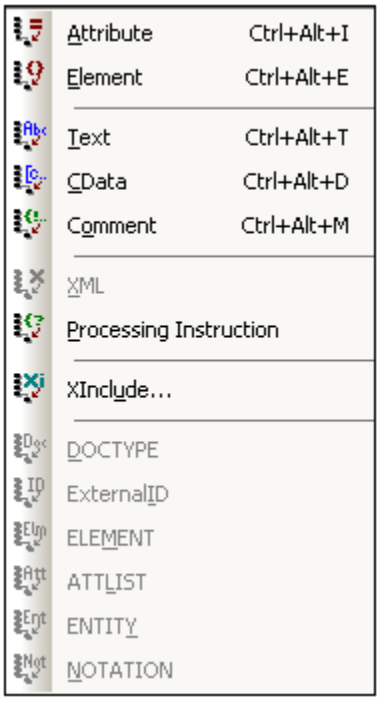
On clicking **OK**, the encoded text of the selected file is inserted at the cursor location, with an enclosing element if this has been specified.

The encoded file is appended in Grid View.

11.4.3 Add Child

The **XML** | **Add Child** command, though enabled in all views, can be used in Grid View only. It opens a submenu (*see screenshot*) with which you can add the following child items to the currently selected element.

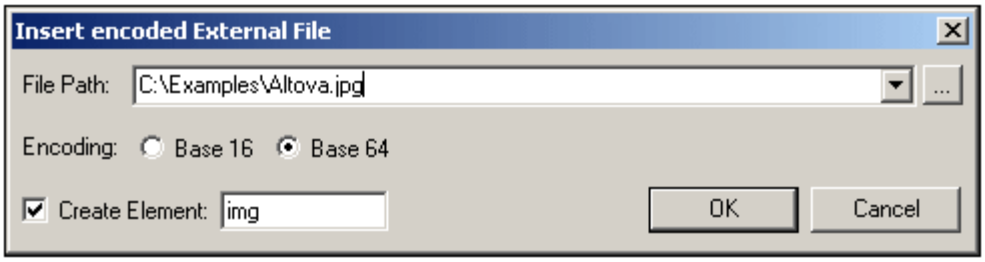
- The XML declaration and node types (Attribute, Element, Text, CDATA, Comment, Processing Instruction) in XML documents;
- DOCTYPE declarations and external DTD declarations in XML documents
- DTD declarations (ELEMENT, ATTLIST, ENTITY, and NOTATION) in DTD documents and internal DTD declarations of XML documents.



Add Child Encoded External File

The **XML** | **Add Child | Encoded External File** command is available in Grid View only. It adds a binary encoded file as a child node. The encoded external file is inserted as a child of the Grid View selection.

On clicking the command, the Insert Encoded External File dialog (*screenshot below*) pops up. In it you enter the path to the file, select the encoding you want, and specify whether the encoded file is to be inserted in an element or not.



On clicking **OK**, the encoded text of the selected file is inserted at the cursor location, with an enclosing element if this has been specified.

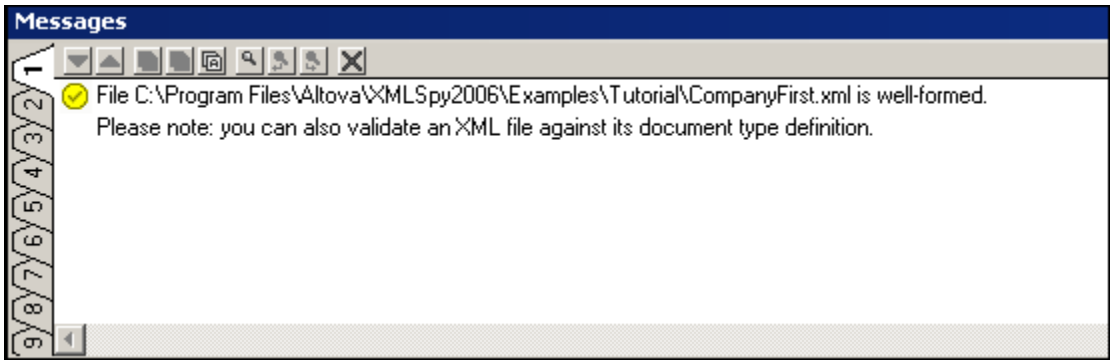
The encoded file is added as a child in Grid View.

11.4.4 Check Well-Formedness

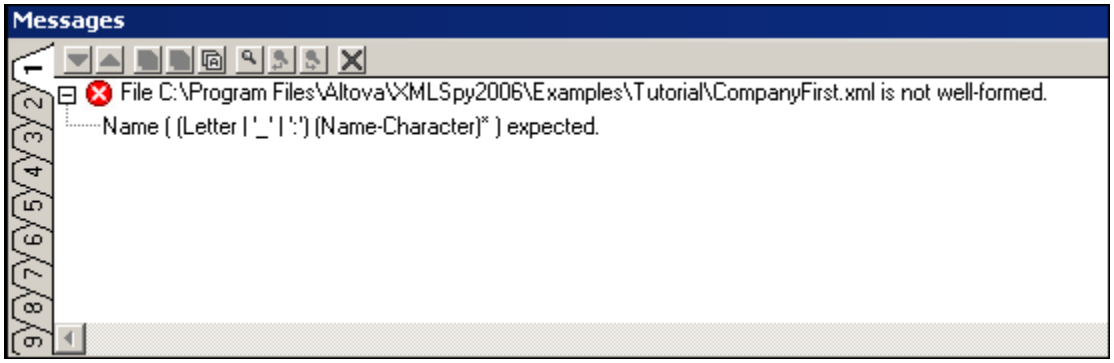
F7

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 or saved. If a file is not well-formed when it is opened, Authentic Desktop opens a Text View window, in which you can edit the file to make it well-formed. The command is enabled in Text View, so the file can be checked for well-formedness.

If the well-formedness check succeeds when you explicitly invoke the check, a message is displayed in the Validation window:



If an error is encountered during the well-formedness check, a corresponding error message is displayed:



Please note: The output of the Messages window has 9 tabs. The validation output is always displayed in the active tab. Therefore, you can check well-formedness in tab1 for one schema file and keep the result by switching to tab 2 before validating the next schema document (otherwise tab 1 is overwritten with the validation result).

It is generally not permitted to save a malformed XML document, but Authentic Desktop gives you a Save Anyway option. This is useful when you want to suspend your work temporarily (in a not well-formed condition) and resume it later.

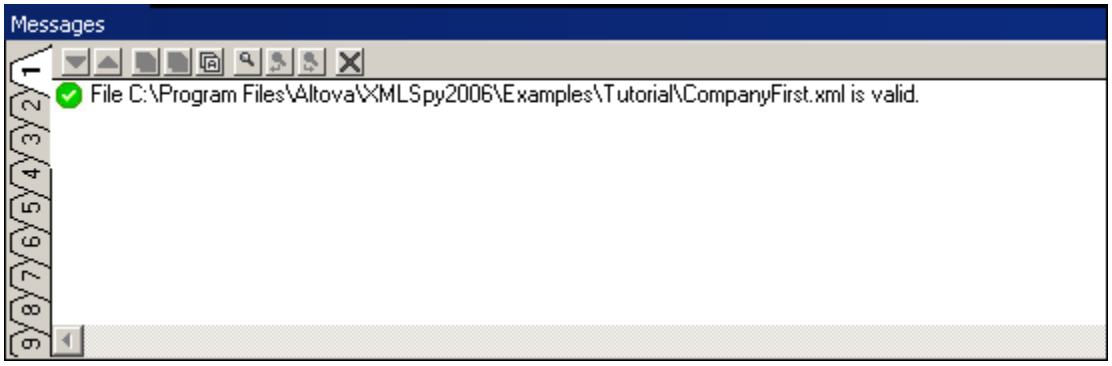
Please note: You can also use the **Check well-formedness** command on any file, folder, or group of files in the active project window. Click on the respective item, and then on the Check Well-Formedness icon.

11.4.5 Validate XML

F8

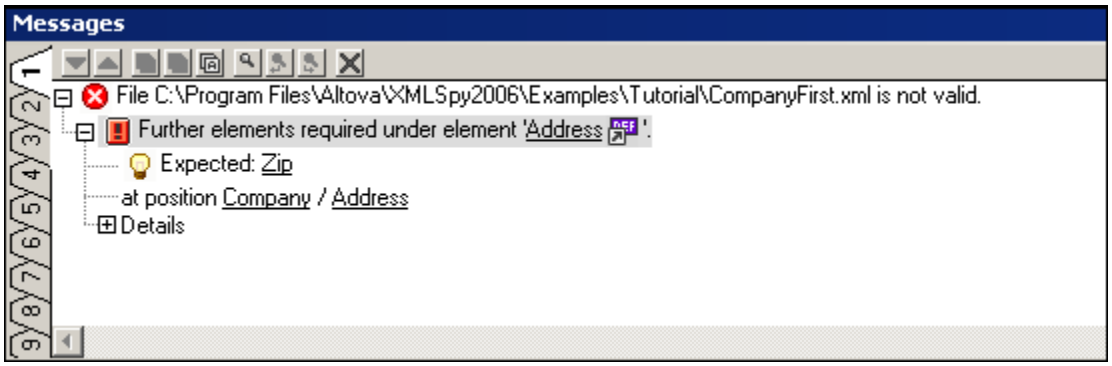
The **XML** | **Validate (F8)** command enables you to validate XML documents against DTDs, XML Schemas, and other schemas. You can specify that a document be automatically validated when a file is opened or saved (**Tools | Options | File**). The **Validate** command also carries out a well-formedness check before checking validity, so there is no need to use the Check Well-Formedness command before using the **Validate** command.

If a document is valid, a successful validation message is displayed in the Validation window:



Otherwise, a message that describes the error is displayed.

Please note: You can click on the links in the error message to jump to the spot in the XML file where the error was found.



Please note: The output of the Validation window has 9 tabs. The validation output is always displayed in the active tab. Therefore, you can check well-formedness in tab1 for one schema file and keep the result by switching to tab 2 before validating the next schema document (otherwise tab 1 is overwritten with the validation result).

The command 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** or **F8**. Invalid files in a project will be opened and made active in the Main Window, and the File Is Invalid error message will be displayed.

Validating XML documents
To validate an XML file, make the XML document active in the Main Window, and click **XML |**

Validate or **F8**. The XML document is validated against the schema referenced in the XML file. If no reference exists, an error message is displayed in the Messages Window. As long as the XML document is open, the schema is kept in memory (see in the DTD/Schema menu).

Automating validation with Altova XML 2012

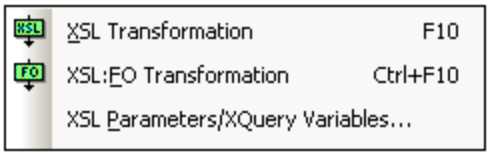
AltovaXML is a free application which contains Altova's XML Validator, XSLT 1.0, XSLT 2.0, and XQuery 1.0 engines. It can be used from the command line, via a COM interface, in Java programs, and in .NET applications to validate XML documents, transform XML documents using XSLT 1.0 and 2.0 stylesheets, and execute XQuery documents.

Validation tasks can therefore be automated with the use of Altova XML. For example, you can create a batch file that calls AltovaXML to perform validation on a set of documents and sends the output to a text file. See the AltovaXML documentation for details.

11.5 XSL/XQuery Menu

The XSL Transformation language lets you specify how an XML document should be converted into other XML documents or text files. One kind of XML document that is generated with an XSLT document is an FO document, which can then be further processed to generate PDF output. Authentic Desktop contains built-in XSLT processors (for XSLT 1.0 and XSLT 2.0) 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 tab of the Options dialog (Tools | Options) in order to be able to use it directly from within the Authentic Desktop interface.

Commands to deal with all the above transformations are accessible in the **XSL/XQuery** menu. In addition, this menu also contains commands to work with the Altova XSLT/XQuery Debugger.



11.5.1 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), 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 tab of the Options dialog (**Tools | Options**).

If your XML document contains a reference to an XSLT stylesheet, then this stylesheet is used for the transformation. (If the XML document is part of a project, an XSLT stylesheet can be specified on a per-folder basis in the Project Properties dialog. Right-click the project folder/s or file/s you wish to transform and select XSL Transformation.) If an XSLT stylesheet has not been assigned to an XML file, you are prompted for the XSLT stylesheet to use. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

Automating XSLT transformations with AltovaXML 2012

AltovaXML is a free application which contains Altova's XML Validator, XSLT 1.0, XSLT 2.0, and XQuery 1.0 engines. It can be used from the command line, via a COM interface, in Java programs, and in .NET applications to validate XML documents, transform XML documents using XSLT 1.0 and 2.0 stylesheets, and execute XQuery documents.

XSLT transformation tasks can therefore be automated with the use of AltovaXML. For example, you can create a batch file that calls AltovaXML to transform a set of documents. See the AltovaXML documentation for details.

Transformations to ZIP files

In order to enforce output to a ZIP file, including Open Office XML (OOXML) files such as `.docx`, one must specify the ZIP protocol in the file path of the output file. For example:

```
filename.zip|zip/filename.xxx
filename.docx|zip/filename.xxx
```

Note: The directory structure might need to be created before running the transformation. If you are generating files for an Open Office XML archive, you would need to zip the archive files in order to create the top-level OOXML file (for example, `.docx`).

11.5.2 XSL-FO Transformation

Ctrl+F10

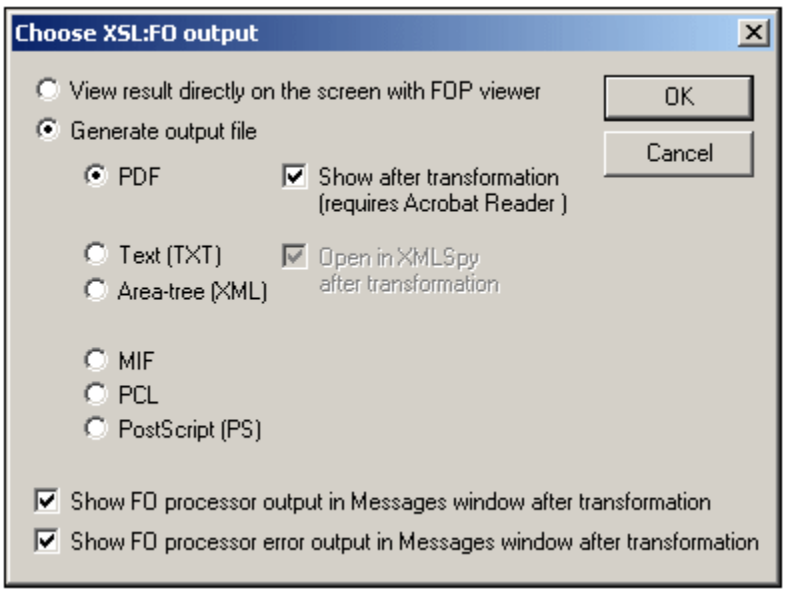
FO is an XML format that describes paged documents. An FO processor, such as the Apache XML Project's FOP, takes an FO file as input and generates PDF as output. So, the production of a PDF document from an XML document is a two-step process.

1. The XML document is transformed to an FO document using an XSLT (aka XSL-FO) 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 (or XSL-FO) stylesheet required to transform to FO is not referenced in the XML document, you are prompted to assign one for the transformation (*screenshot below*). Note that you can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button). The transformation from XML to XSL-FO is carried out by the XSLT processor specified in the XSL tab of the Options dialog (**Tools | Options**). By default the selected XSLT processor is XMLSpy's built-in XSLT processor. The resultant FO document is directly processed with the FO processor specified in the XSL tab 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 tab of the Options dialog (**Tools | Options**).

XSL:FO Transformation output
The **XSL:FO Transformation** command pops up the Choose XSL:FO Output dialog (*screenshot below*). (If the active document is an XML document without an XSLT assignment, you are first prompted for an XSLT file.)



You can view the output of the FO processor directly on screen using FOP viewer or you can

generate an output file in any one of the following formats: PDF, text, an XML area tree, MIF PCL, or PostScript. You can also switch on messages from the FO processor to show (i) the processor's standard output message in the Messages window; and (ii) the processor's error messages in the Messages window. To switch on either these two options, check the appropriate check box at the bottom of the dialog.

Please note:

- The Apache FOP processor can be downloaded free of charge using the link at the Altova Download Center. After downloading and installing FOP, you must set the path to the FOP batch file in the XSL tab of the Options dialog (**Tools | Options**).
- The XSL:FO Transformation command can not only be used on the active file in the Main Window but also on any file or folder you select in the active project. To do this, right-click and select **XSL:FO Transformation**. The XSLT stylesheet assigned to the selected project folder is used.

11.5.3 XSL Parameters / XQuery Variables

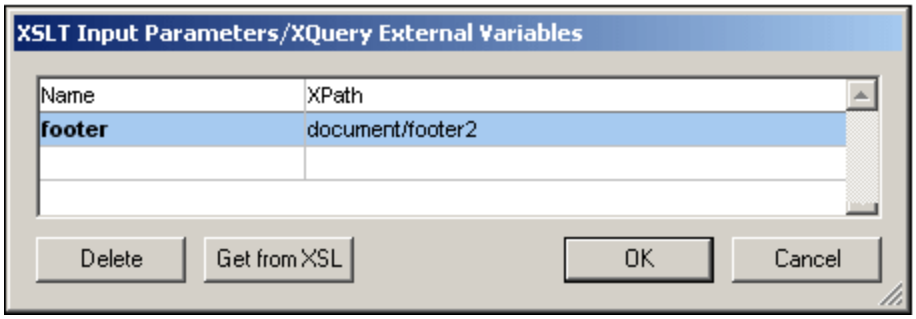
The **XSL/XQuery | XSL Parameters/XQuery Variables** command opens the XSLT Input Parameters/XQuery External Variables dialog (*see screenshot*). You can enter the name of one or more parameters you wish to pass to the XSLT stylesheet, or one or more external XQuery variables you wish to pass to the XQuery document, and their respective values. These parameters are used as follows in XMLSpy:

- 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.

Please 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.

Using XSLT Parameters

The value you enter for the parameter can be an XPath expression without quotes or a text string delimited by quotes. If the active document is an XSLT document, the **Get from XSL** button will be enabled. Clicking this button inserts parameters declared in the XSLT into the dialog together with their default values. This enables you to quickly include declared parameters and then change their default values as required.



Please note: Once a set of parameter-values is entered in the XSLT Input Parameters/XQuery External Variables dialog, it is used for all subsequent transformations until it is explicitly deleted or the application is restarted. Parameters entered in the XSLT Input Parameters/XQuery External Variables dialog are specified at the application-level, and will be passed to the respective XSLT document for every transformation that is carried out via the IDE from that point onward. This means that:

- parameters are not associated with any particular document
- any parameter entered in the XSLT Input Parameters/XQuery External Variables dialog is erased once Authentic Desktop has been closed.

Usage example for XSLT parameters

In the following example, we select the required document footer from among three possibilities in the XML document (`footer1`, `footer2`, `footer3`).

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
   xsi:noNamespaceSchemaLocation="C:\workarea\footers\footers.xsd">
   <footer1>Footer 1</footer1>
   <footer2>Footer 2</footer2>
   <footer3>Footer 3</footer3>
   <title>Document Title</title>
   <para>Paragraph text.</para>
   <para>Paragraph text.</para>
</document>
```

The XSLT file contains a local parameter called `footer` in the template for the root element. This parameter has a default value of `footer1`. The parameter value is instantiated subsequently in the template with a `$footer` value in the definition of the footer block.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format">
   ...
   <xsl:param name="footer" select="document/footer1" />
   ...
   <xsl:template match="/">
      <fo:root>
         <xsl:copy-of select="$fo:layout-master-set" />
         <fo:page-sequence master-reference="default-page"
            initial-page-number="1" format="1">
            <fo:static-content flow-name="xsl-region-after"
               display-align="after">
               ...
               <fo:inline color="#800000" font-size="10pt" font-weight="bold">
                  <xsl:value-of select="$footer"/>
               </fo:inline>
               ...
            </fo:static-content>
         </fo:page-sequence>
      </fo:root>
   </xsl:template>
</xsl:stylesheet>
```

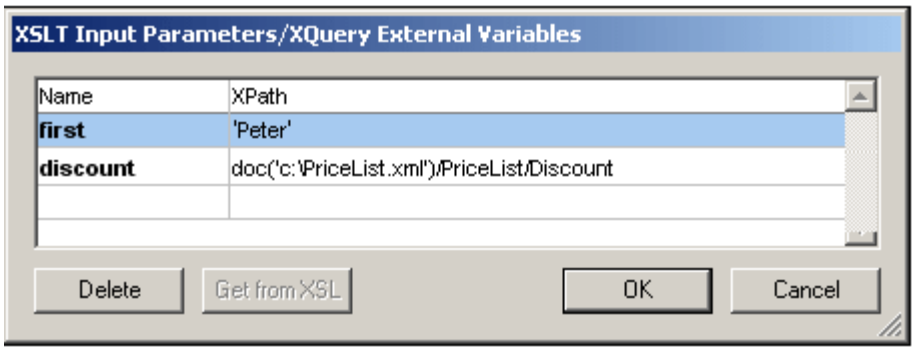
In the XSLT Input Parameters dialog, a new value for the `footer` parameter can be entered, such as the XPath: `document/footer2` (*see screenshot above*) or a text string. During transformation, this value is passed to the `footer` parameter in the template for the root element and is the value used when the footer block is instantiated.

Note:

- If you use the **XSL:FO Transformation** command (**XSL/XQuery | XSL:FO Transformation**), 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.



Note: Once a set of external XQuery variables are entered in the XSLT Input Parameters/XQuery External Variables dialog, they are used for all subsequent executions until they are explicitly deleted or the application is restarted. Variables entered in the XSLT Input Parameters/XQuery External Variables 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 point onward. This means that:

- Variables are not associated with any particular document
- Any variable entered in the XSLT Input Parameters/XQuery External Variables dialog is erased once the application (Authentic Desktop) has been closed down.

Usage example for external XQuery variables
In the following example, a variable `$first` is declared in the XQuery document and is then used in the return clause of the FLWOR expression:

```
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 )
  ```

Note: It is not an error if an external XQuery variable (or XSLT parameter) is defined in the XSLT Input Parameters/XQuery External Variables dialog but is not used in the XQuery

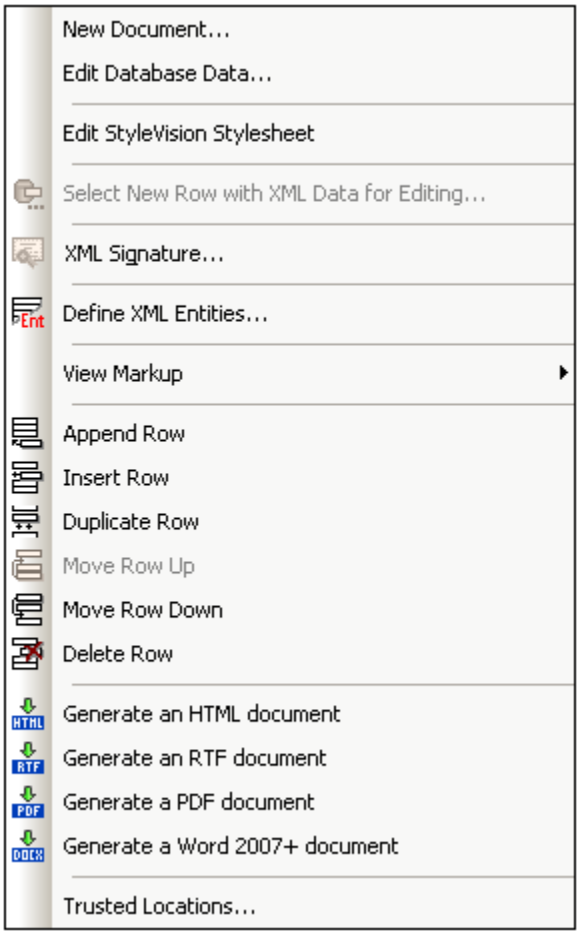
document. Neither is it an error if an XSLT parameter (or external XQuery variable) is defined in the XSLT Input Parameters/XQuery External Variables dialog but is not used in an XSLT transformation.

11.6 Authentic Menu

Authentic View enables you to edit XML documents **based on StyleVision Power Stylesheets (.sps files) created in Altova's StyleVision product!** These stylesheets contain information that enables an XML file to be displayed graphically in Authentic View. In addition to containing display information, StyleVision Power Stylesheets also allow you to write data to the XML file. This data is dynamically processed using all the capability available to XSLT stylesheets and instantly produces the output in Authentic View.

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

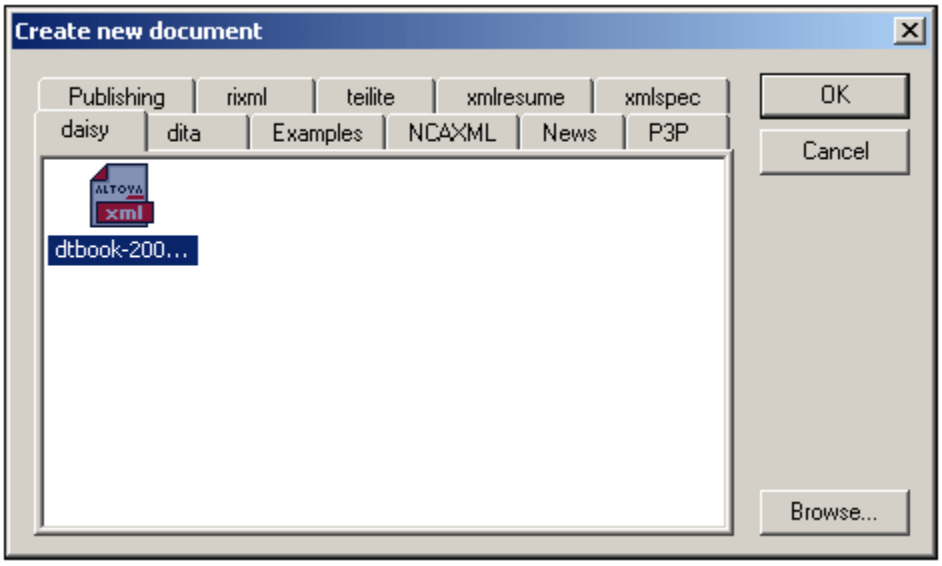
The **Authentic** menu contains commands relevant to editing XML documents in Authentic View . For a tutorial on Authentic View, see the Tutorials section.



11.6.1 New Document

The **Authentic** | **New Document...** 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.

Clicking the **New Document...** command opens the Create New Document dialog.



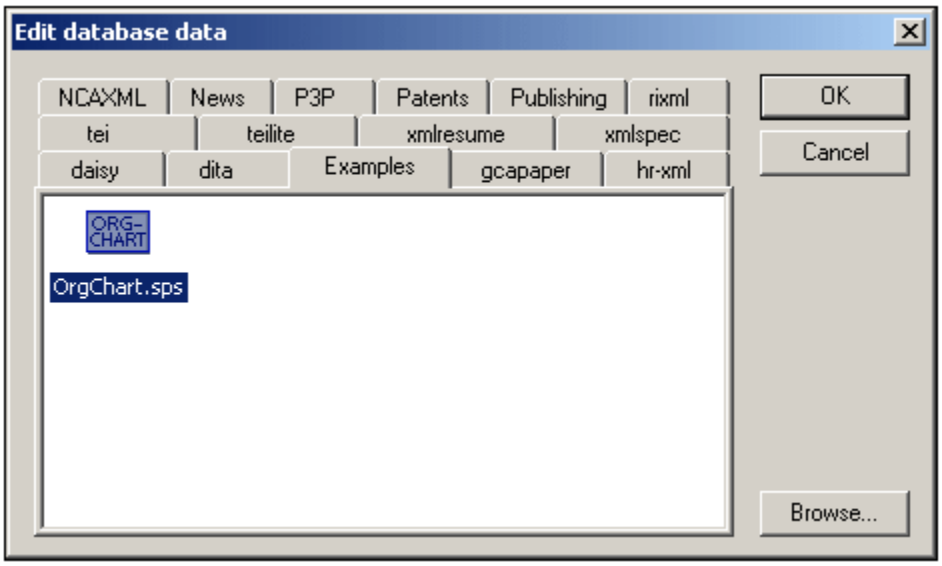
Browse for the required SPS file, and select it. This opens an XML document template in Authentic View.

Note: StyleVision Power Stylesheets are created using Altova StyleVision. The StyleVision Power Stylesheet has a Template XML File assigned to it. The data in this XML file provides the starting data of the new document template that is opened in Authentic View.

11.6.2 Edit Database Data

The **Authentic** | **Edit Database Data...** command enables you to open an editable view of a database (DB) in Authentic View. All the information about connecting to the DB and how to display the DB and accept changes to it in Authentic View is contained in a StyleVision Power Stylesheet. It is such a DB-based StyleVision Power Stylesheet that you open with the **Edit Database Data...** command. This sets up a connection to the DB and displays the DB data (through an XML lens) in Authentic View.

Clicking the **Edit Database Data...** command opens the Edit Database Data dialog.



Browse for the required SPS file, and select it. This connects to the DB and opens an editable view of the DB in Authentic View. The design of the DB view displayed in Authentic View is contained in the StyleVision Power Stylesheet.

Please note: If, with the **Edit Database Data...** command, you attempt to open a StyleVision Power Stylesheet that is not based on a DB or to open a DB-based StyleVision Power Stylesheet that was created in a version of StyleVision prior to the StyleVision 2005 release, you will receive an error.

Please note: StyleVision Power Stylesheets are created using Altova StyleVision.

11.6.3 Assign/Edit a StyleVision Stylesheet

Assign a StyleVision Stylesheet
The **Assign a StyleVision Stylesheet** command assigns a StyleVision Power Stylesheet (SPS) to an **XML document** to enable the viewing and editing of that XML document in Authentic View. The StyleVision Power Stylesheet that is to be assigned to the XML file must be based on the same schema as that on which the XML file is based.

To assign a StyleVision Power Stylesheet to an XML file:

1. Make the XML file the active file and select the **Authentic** | **Assign a StyleVision Stylesheet...** command.
2. The command opens a dialog box in which you specify the StyleVision Power Stylesheet file you wish to assign to the XML.
3. Click **OK** to insert the required SPS statement into your XML document. Note that you can make the path to the assigned file relative by clicking the **Make path relative to ...** check box. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

```
<?xml version="1.0" encoding="UTF-8"?>
<?altova_sps HTML-Orgchart.sps?>
```

In the example above, the StyleVision Power Stylesheet is called `HTML_Orgchart.sps`, and it is located in the same directory as the XML file.

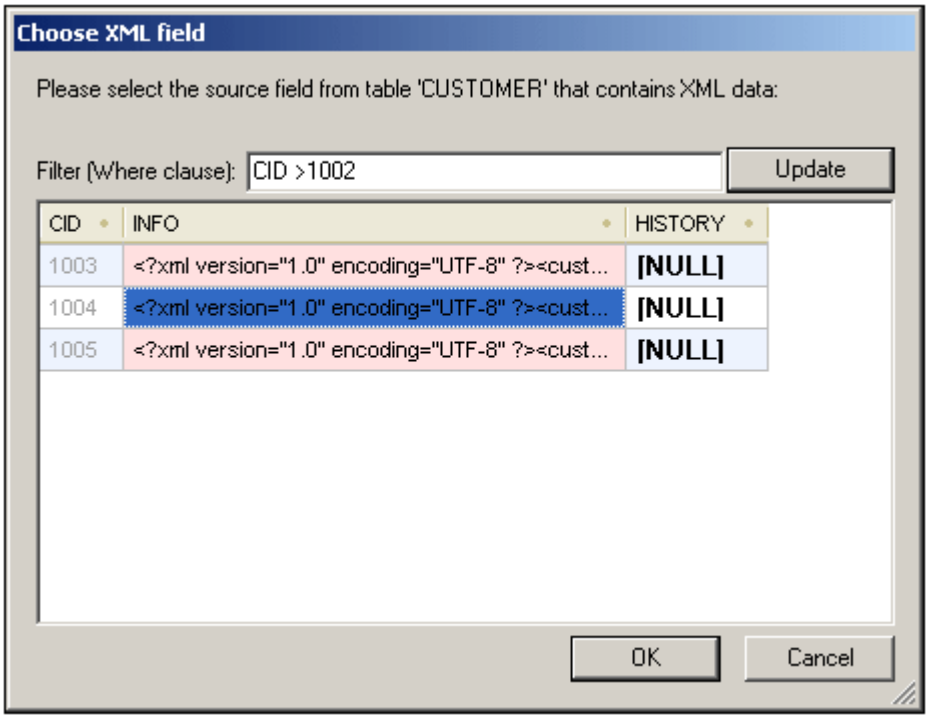
Please note: Previous versions of Altova products used a processing instruction with a target or name of `xmlspysps`, so a processing instruction would look something like `<?xmlspysps HTML-Orgchart.sps?>`. These older processing instructions are still valid with Authentic View in current versions of Altova products.

Edit StyleVision Stylesheet
The **Authentic** | **Edit StyleVision Stylesheet** command starts StyleVision and allows you to edit the StyleVision Power Stylesheet immediately in StyleVision.

11.6.4 Select New Row with XML Data for Editing

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

When an XML DB is used as the XML data source, the XML data that is displayed in Authentic View is the XML document contained in one of the cells of the XML data column. The **Select New Row with XML Data for Editing** command enables you to select an XML document from another cell (or row) of that XML column. Selecting the **Select New Row...** command pops up the Choose XML Field dialog (*screenshot below*), which displays the table containing the XML column.



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

11.6.5 XML Signature

The **XML Signature** command is available in Authentic View when the associated SPS has XML Signatures enabled. The **XML Signature** command is also available as the XML Signature toolbar icon  in the Authentic toolbar.

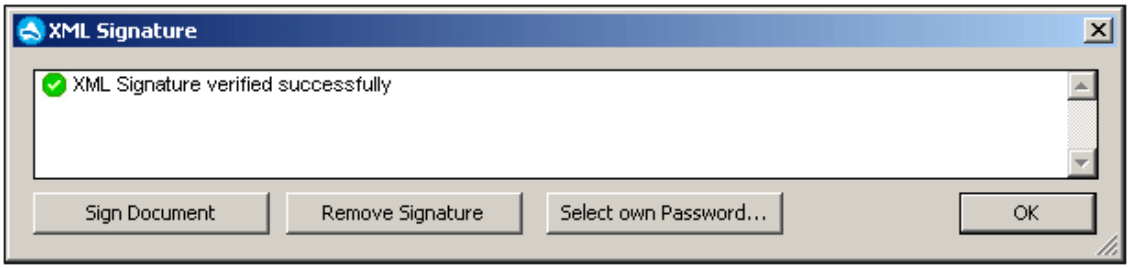
Verification and own certificate/password

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

If the **Select Own Certificate** or **Select Own Password** button is present in this dialog, it means that the Authentic View has been given the option of selecting an own certificate/ password. (Whether a certificate or password is to be chosen has been decided by the SPS designer at the time the signature was configured. The signature will be either certificate-based or password-based.) Clicking either of these buttons, if present in the dialog, enables the Authentic View user to browse for a certificate or to enter a password. The Authentic View user's selection is stored in memory and is valid for the current session only. If, after selecting a certificate or password, the document or application is closed, the certificate/password setting reverts to the setting originally saved with the SPS.

Verification and authentication information

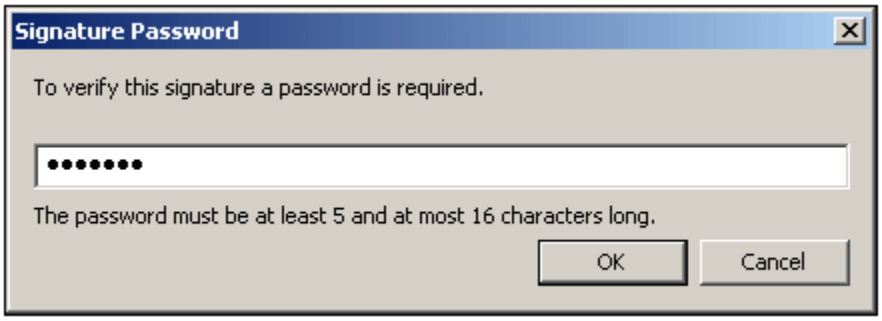
If the verification process is run on a signed document, two general situations are possible. First: If the authentication information is available (in the signature or the SPS), then the verification process is executed directly and the result is displayed (*screenshot below*).



Authentication information is either the signing certificate's key information or the signing password. The SPS designer will have specified whether the certificate's key information is saved in the signature when the XML document is signed, or, in the case of a password-based signature, whether the password is saved in the SPS. In either of these cases, the authentication is available. Consequently the verification process will be run directly, without requiring any input from the Authentic View user.

The second possible general situation occurs when authentication information is not available in the signature (certificate's key information) or SPS file (password). In this situation, the

Authentic View user will be asked to supply the authentication information: a password (*see screenshot below*) or the location of a certificate.



11.6.6 Define XML Entities

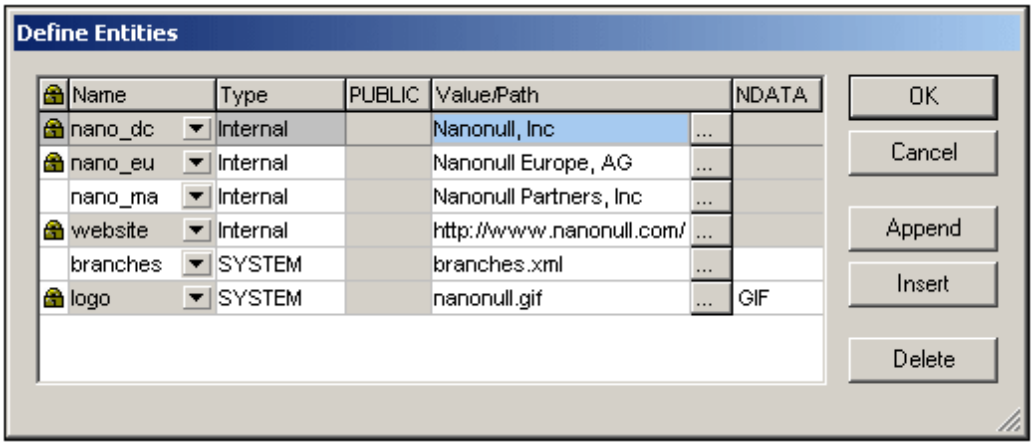
You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the **Insert Entity** submenu of the context menu. When you double-click on an entity in the Entities Entry Helper,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

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

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

To define an entity:

1. Click **Authentic | Define XML Entities...**. This opens the Define Entities dialog.



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

   - If the entity type is Internal, enter the text string you want as the value of your entity. Do not enter quotes to delimit the entry. Any quotes that you enter will be treated as part of the text string.

- If the entity type is SYSTEM, enter the URI of the resource or select a resource on your local network by using the **Browse** button. If the resource contains parsed data, it must be an XML file (i.e. it must have a `.xml` extension). Alternatively, the resource can be a binary file, such as a GIF file.
- If the entity type is PUBLIC, you must additionally enter a system identifier in this field.

6. The NDATA entry tells the processor that this entity is not to be parsed but to be sent to the appropriate processor. The NDATA field should therefore be used with unparsed entities only.

Dialog features
You can append, insert, and delete entities by clicking the appropriate buttons. You can also sort entities on the alphabetical value of any column by clicking the column header; clicking once sorts in ascending order, twice in descending order. You can also resize the dialog box and the width of columns.

Once an entity is used in the XML document, it is locked and cannot be edited in the Define Entities dialog. Locked entities are indicated by a lock symbol in the first column. Locking an entity ensures that the XML document valid with respect to entities. (The document would be invalid if an entity is referenced but not defined.)

Duplicate entities are flagged.

Limitations

- An entity contained within another entity is not resolved, either in the dialog, Authentic View, or XSLT output, and the ampersand character of such an entity is displayed in its escaped form, i.e. `&`.
- External entities are not resolved in Authentic View, except in the case where an entity is an image file and it is entered as the value of an attribute which has been defined in the schema as being of type `ENTITY` or `ENTITIES`. Such entities are resolved when the document is processed with an XSLT generated from the SPS.

11.6.7 View Markup

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

The **Hide Markup** command hides markup symbols in Authentic View.

The **Show Small Markup** command shows small markup symbols in Authentic View.

The **Show Large Markup** command shows large markup symbols in Authentic View.

The **Show Mixed Markup** command shows mixed markup symbols in Authentic View. The person who designs the StyleVision Power Stylesheet can specify either large markup, small markup, or no markup for individual elements/attributes in the document. The Authentic View user sees this customized markup in mixed markup viewing mode.

11.6.8 Append/Insert/Duplicate/Delete Row

The **Append Row** command appends a row to the current table in Authentic View.

The **Insert Row** command inserts a row into the current table in Authentic View.

The **Duplicate Row** command duplicates the current table row in Authentic View.

The **Delete Row** command deletes the current table row in Authentic View.

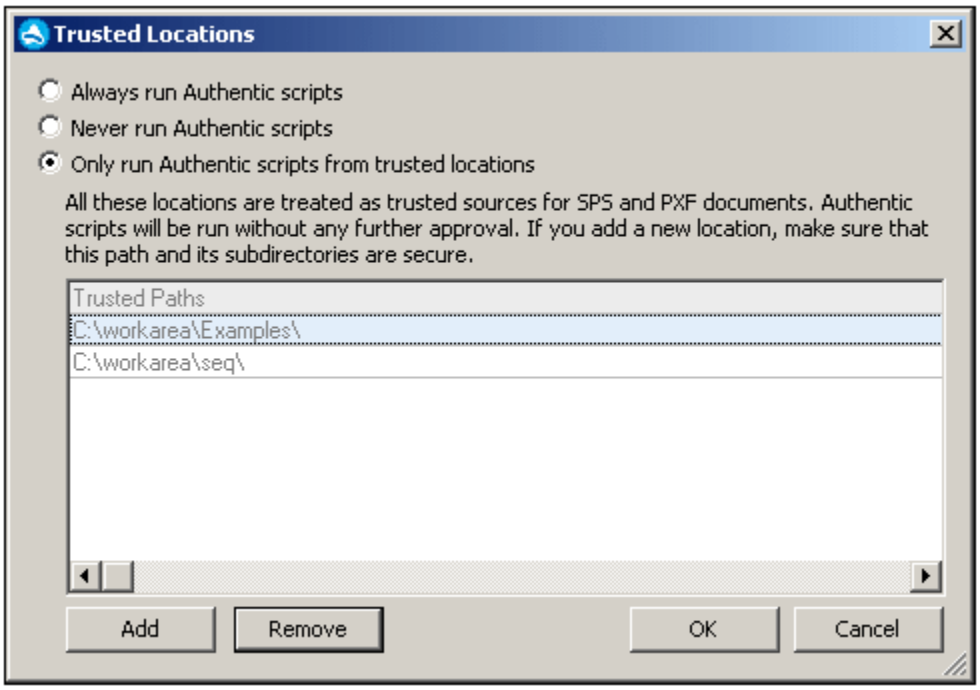
11.6.9 Move Row Up/Down

The **Move Row Up** command moves the current table row up by one row in Authentic View.

The **Move Row Down** command moves the current table row down by one row in Authentic View.

11.6.10 Trusted Locations

The Trusted Locations command pops up the Trusted Locations dialog (*screenshot below*), in which you can specify the security settings for scripts in an SPS. When an XML file based on a script-containing SPS is switched to Authentic View, the script will be allowed to run or not depending on the settings you make in this dialog.



The three available options are:

- Authentic scripts are always run when a file is opened in Authentic View.
- Authentic scripts are never run when a file is opened in Authentic View.
- Only Authentic scripts in trusted locations are run. The list of trusted (folder) locations is shown in the bottom pane. Use the **Add** button to browse for a folder and add it to the list. To remove an entry from the list, select an entry in the Trusted Locations list and click **Remove**.

11.7 View Menu

The **View** menu controls the display of the active Main window and allows you to change the way Authentic Desktop displays your XML documents.

This section provides a complete description of commands in the **View** menu.

11.7.1 Authentic View

This command **switches** the current document into the Authentic View.

Authentic View enables you to edit XML documents **based on StyleVision Power Stylesheet templates created in StyleVision!** The templates in StyleVision are saved as **StyleVision Power Stylesheet**s (***.sps** files), and supply all the necessary information needed by Authentic View.

To **open** a template select the **File | New** command and then click the **Select a StyleVision stylesheet...** button. Please see the Authentic View documentation for further information.

Please note: If, when you try to switch to Authentic View, you receive a message saying that a temporary (temp) file could not be created, contact your system administrator. The system administrator must change the default Security ID for "non-power users" to allow them to create folders and files.

11.7.2 Browser View

This command **switches** the current document into Browser View.

This view uses an XML-enabled browser to render the XML document using information from potential CSS or XSL style-sheets.

When switching to browser view, the document is first checked for validity, if you have selected Validate upon saving in the File tab of the Options dialog. Use the menu command **Tools | Options** to open this dialog.

For further information on this view, please see the detailed description of the various views in the Main Window section.

11.8 Browser Menu

The commands in the **Browser** menu are enabled in Browser View only.

11.8.1 Back

Backspace (Alt + Left Arrow)

In Browser View, the **Back** command displays the previously viewed page. The Backspace key also achieves the same effect. The **Back** command is useful if you click a link in your XML document and want to return to your XML document.

In Schema View, the **Back** command takes you to the previously viewed component. The shortcut key is **Alt + Left Arrow**. Using the **Back** command up to 500 previously viewed positions can be re-viewed.

11.8.2 Forward

 **(Alt + Right Arrow)**

The **Forward** command is only available once you have used the **Back** command. It moves you forward through (i) previously viewed pages in Browser View, and (ii) previous views of schema components in Schema View.

11.8.3 Stop

The **Stop** command instructs the browser to stop loading your document. This is useful if large external files or graphics are being downloaded over a slow Internet connection, and you wish to stop the process.

11.8.4 Refresh

 **F5**

The **Refresh (F5)** command updates the Browser View by reloading the document and related documents, such as CSS and XSL stylesheets, and DTDs.

11.8.5 Fonts

The **Fonts** command allows you to select the default font size for rendering the text of your XML document. It is similar to the Font Size command in most browsers.

11.8.6 Separate Window

The **Separate Window** command opens the Browser View in a separate window, so that side-by-side viewing is possible. If you have separated the Browser View, press **F5** in editing view to automatically refresh the corresponding Browser View. To dock separate windows back into the interface, click the maximize button (at top right) of the active window.

11.9 Tools Menu

The tools menu allows you to:

- Check the spelling of your XML documents
- Access the scripting environment of Authentic Desktop. You can create, manage and store your own forms, macros and event handlers
- View the currently assigned macros
- Compare any two files to check for differences
- Compare any two folders to check for differences
- Define global resources
- Change the active configuration for global resources in XMLSpy
- Customize your version of Authentic Desktop: define your own toolbars, keyboard shortcuts, menus, and macros
- Define global Authentic Desktop settings

11.9.1 Spelling

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

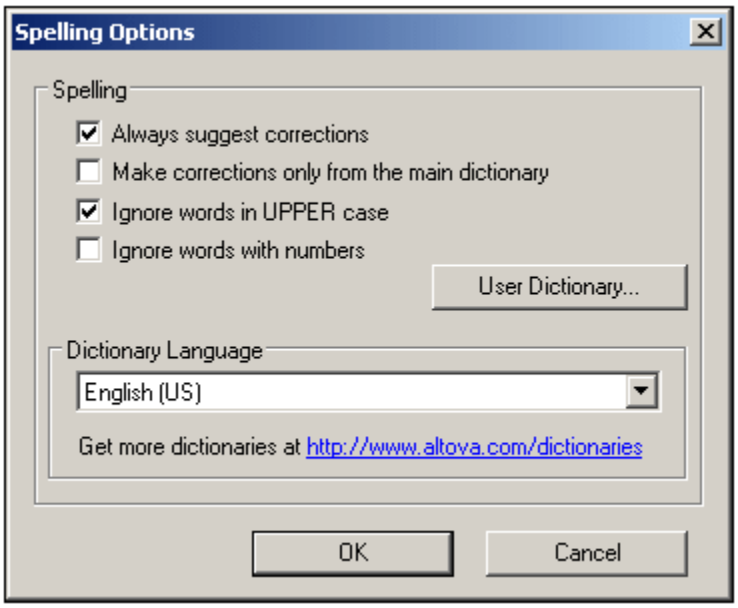
Note: The selection of built-in dictionaries that ship with Altova software does not constitute any language preferences by Altova, but is largely based on the availability of dictionaries that permit redistribution with commercial software, such as the MPL, LGPL, or BSD licenses. Many other open-source dictionaries exist, but are distributed under more restrictive licenses, such as the GPL license. Many of these dictionaries are available as part of a separate installer located at http://www.altova.com/dictionaries. It is your choice as to whether you can agree to the terms of the license applicable to the dictionary and whether the dictionary is appropriate for your use with the software on your computer.

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

- Selecting the spellchecker language
- Defining the scope of the check
- Running the spelling check

Selecting the spellchecker language
The spellchecker language can be set as follows:

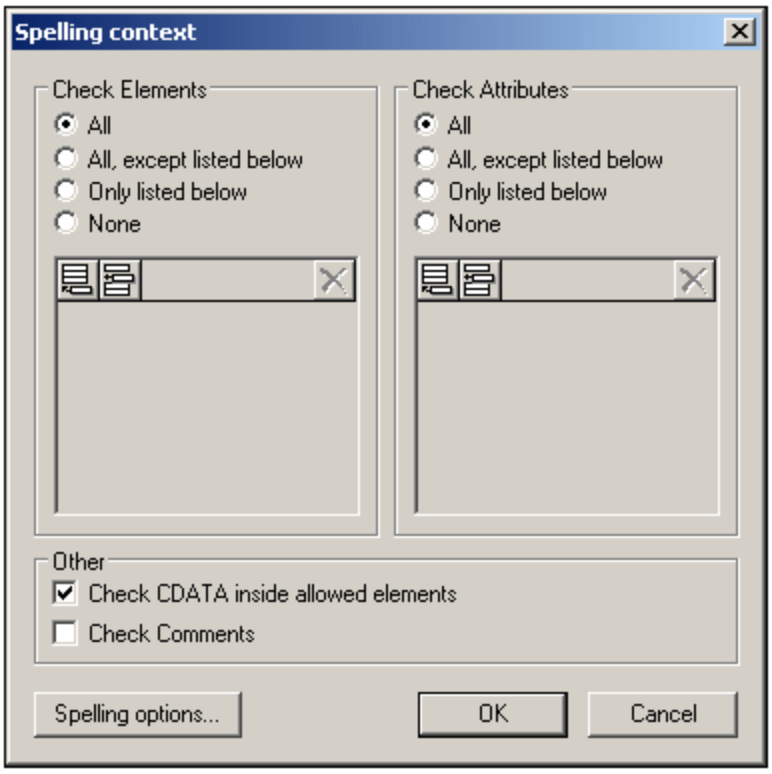
1. Click the **Tools | Spelling Options** menu command.
2. In the Spelling Context dialog that pops up, click the **Spelling Options** button.
3. In the Spelling Options dialog (*screenshot below*), select one of the installed dictionaries from the dropdown list of the Dictionary Language combo box.



4. Click **OK** to finish.

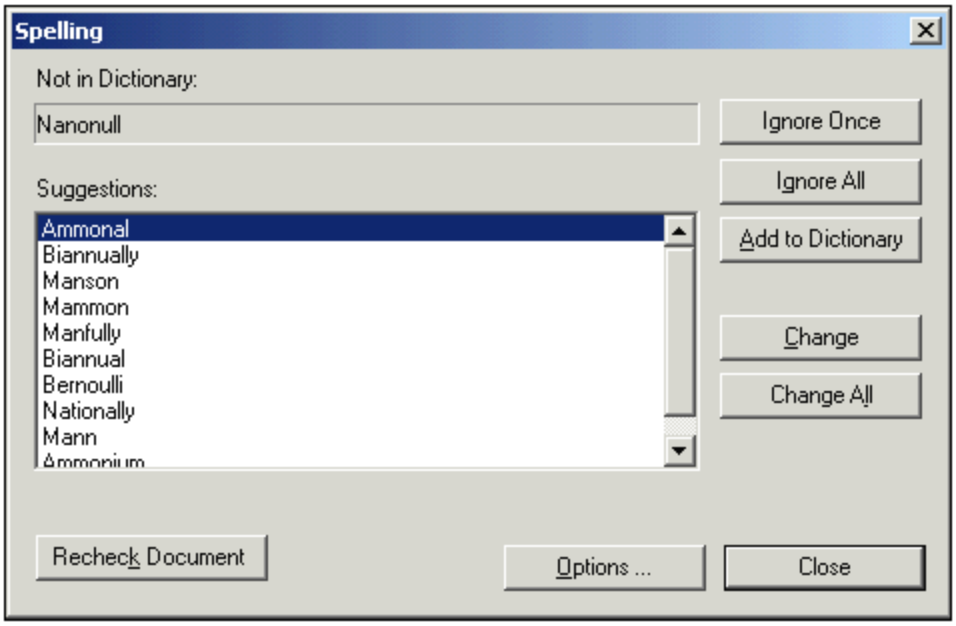
The dictionary language you selected will be used by the spellchecker for spelling checks. If the language you want is not already installed, you can download additional language dictionaries. How to do this is described in the section, Adding dictionaries for the spellchecker.

Defining the scope of the check
When the spellchecker is run in Text View or Grid View, the scope of the check can be defined immediately before starting the check. Do this by selecting the command **Tools | Spelling Options** and by defining the required scope in the Spelling Context dialog that pops up (*see screenshot below*).



You can select which elements and attributes are to be checked and whether CDATA sections and comments should be checked. Also see the description of the Spelling Options command for related information.

Running the spellchecker
The **Tools | Spelling (Shift+F7)** command automatically starts checking the currently active XML document according to the defined scope. 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 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
This command opens a dialog box depending on the current view.

- If the current view is Authentic View, the Spelling Options dialog box is opened.

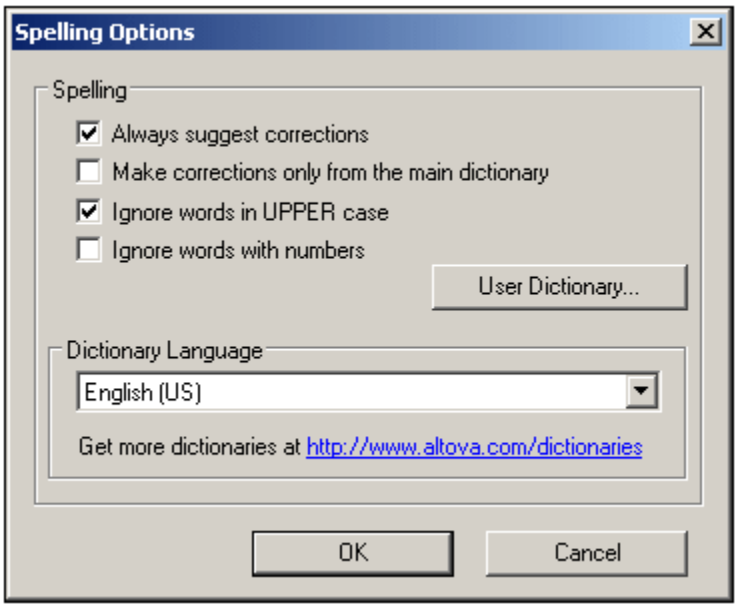
For more information about these dialog boxes, see the section Spelling Options.

Close
This command closes the Spelling dialog box.

11.9.2 Spelling Options

The **Tools | Spelling Options** command opens the **Spelling Options**.

Spelling options
The Spelling Options dialog is used to define global spellchecker options.



Always suggest corrections:
Activating this option causes suggestions (from both the language dictionary and the user dictionary) to be displayed in the Suggestions list box. Disabling this option causes no suggestions to be shown.

Make corrections only from main dictionary:
Activating this option causes only the language dictionary (main dictionary) to be used. The user dictionary is not scanned for suggestions. It also disables the **User Dictionary** button, preventing any editing of the user dictionary.

Ignore words in UPPER case:
Activating this option causes all upper case words to be ignored.

Ignore words with numbers:
Activating this option causes all words containing numbers to be ignored.

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.

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:

On Windows 7: `C:\ProgramData\Altova\SpellChecker\Lexicons`
On Windows XP: `C:\Documents and Settings\All Users\Application Data\Altova\SharedBetweenVersions\SpellChecker\Lexicons`

Within the `Lexicons` folder, different language dictionaries are each stored in different folder: `<language name>\<dictionary files>`. For example, on a Windows 7 machine, files for the two English-language dictionaries (`English (British)` and `English (US)`) will be stored as below:

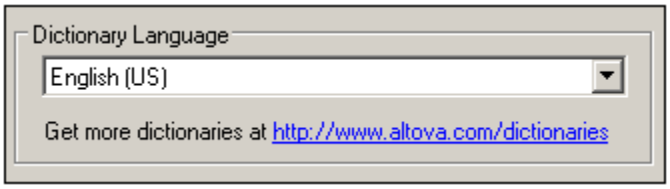
```
C:\ProgramData\Altova\SpellChecker\Lexicons\English (British)\en_GB.aff
C:\ProgramData\Altova\SpellChecker\Lexicons\English (British)\en_GB.dic
C:\ProgramData\Altova\SpellChecker\Lexicons\English (US)\en_US.dic
C:\ProgramData\Altova\SpellChecker\Lexicons\English (US)\en_US.dic
```

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).

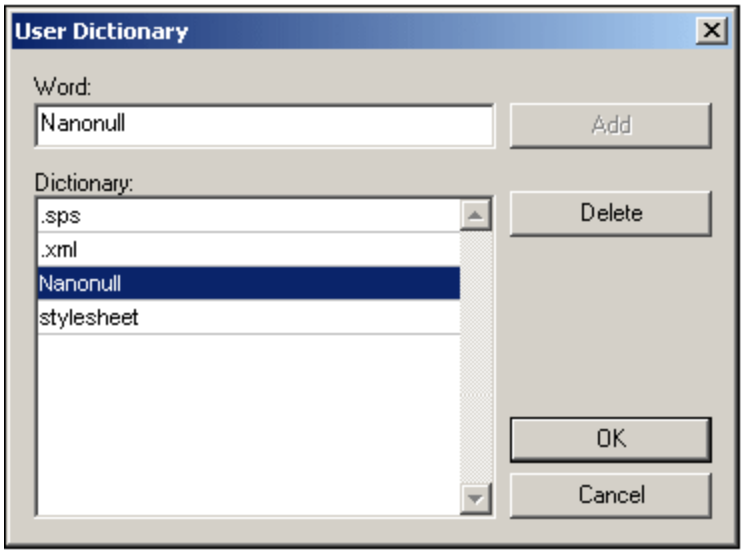
You can add dictionaries for the spellchecker in two ways, neither of which require that the files be registered with the system:

- By adding Hunspell dictionaries into a new subfolder of the `Lexicons` folder. Hunspell dictionaries can be downloaded, for example, from http://wiki.services.openoffice.org/wiki/Dictionaries 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 by clicking the link in the Dictionary language pane of the Spelling Options dialog (*see screenshot below*).



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 screenshot at top of section*).



To add a word to the user dictionary, enter the word in the Word text box and click **Add**. The word will be added to the alphabetical list in the Dictionary pane. To delete a word from the dictionary, select the word in the Dictionary pane and click **Delete**. The word will be deleted from the Dictionary pane. When you have finished editing the User Dictionary dialog, click **OK** for the changes to be saved to the user dictionary.

Words may also be added to the User Dictionary during a spelling check. If an unknown word is encountered during a spelling check, then the Spelling dialog pops up prompting you for the action you wish to take. If you click the **Add to Dictionary** button, then the unknown word is added to the user dictionary.

The user dictionary is located at:

On Windows 7: `C:\Users\<user>\Documents\Altova\SpellChecker\Lexicons\user.dic`
On Windows XP: `C:\Documents and Settings\<user>\My Documents\Altova\SpellChecker\Lexicons\user.dic`

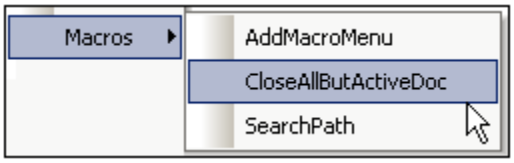
11.9.3 Scripting Editor

The Scripting Editor command opens the Scripting Editor window. How to work with the Scripting Editor is described in the Scripting section of this documentation.

Note: The .NET Framework version 2.0 or higher will have to be installed on your machine in order for the Scripting Editor to run.

11.9.4 Macros

Mousing over the **Macros** command rolls out a submenu containing the macros defined in the Scripting Project that is currently active in Authentic Desktop (*screenshot below*). The active Scripting Project is specified in the Scripting tab of the Options dialog.

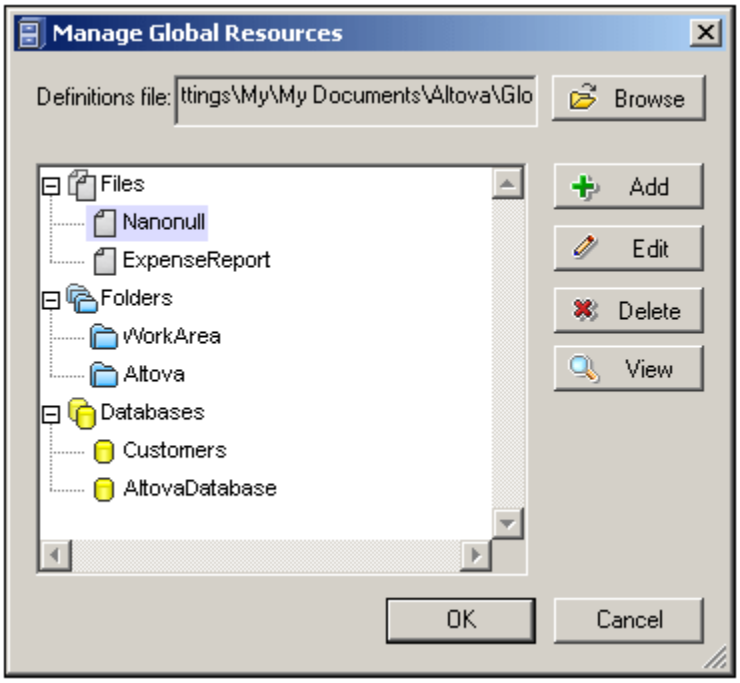


Clicking a macro in the submenu (*see screenshot above*) runs the macro.

11.9.5 Global Resources

The **Global Resources** command pops up the Global Resources dialog (*screenshot below*), in which you can:

- Specify the Global Resources XML File to use for global resources.
- Add file, folder, and database global resources (or aliases)
- Specify various configurations for each global resource (alias). Each configuration maps to a specific resource.

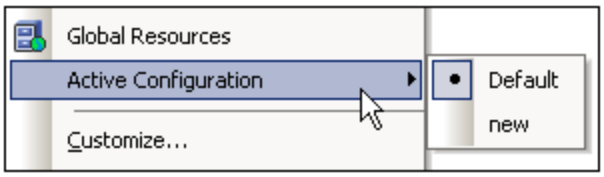


How to define global resources is described in detail in the section, Defining Global Resources.

Note: The Altova Global Resources dialog can also be accessed via the Global Resources toolbar (**Tools | Customize | Toolbars | Global Resources**).

11.9.6 Active Configuration

Mousing over the **Active Configuration** menu item rolls out a submenu containing all the configurations defined in the currently active Global Resources XML File (*screenshot below*).



The currently active configuration is indicated with a bullet. In the screenshot above the currently active configuration is `Default`. To change the active configuration, select the configuration you wish to make active.

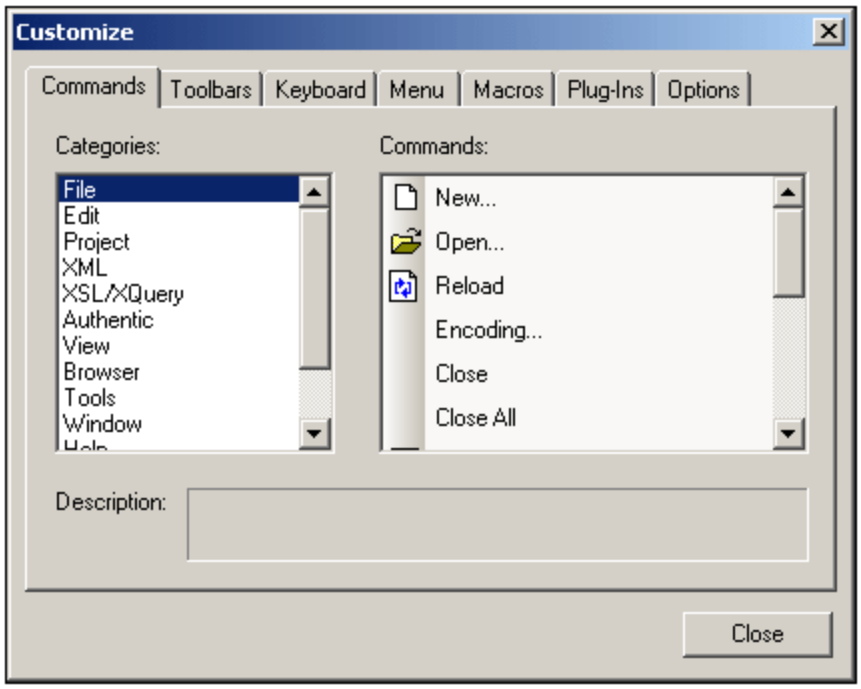
Note: The active configuration can also be selected via the Global Resources toolbar (**Tools | Customize | Toolbars | Global Resources**).

11.9.7 Customize

The **Customize** command lets you customize Authentic Desktop to suit your personal needs.

Commands

The **Commands** tab allows you customize your menus or toolbars.



To add a command to a toolbar or menu:

1. Select the menu item **Tools | Customize**. The Customize dialog appears.
2. Select the **All Commands** category in the Categories list box. The available commands appear in the Commands list box.
3. Click on a command in the Commands list box and drag it to an to an existing menu or toolbar. An **I**-beam appears when you place the cursor over a valid position to drop the command.
4. Release the mouse button at the position you want to insert the command.

- A small button appears at the tip of mouse pointer when you drag a command. The "x" below the pointer means that the command cannot be dropped at the current cursor position.
- The "x" disappears whenever you can drop the command (over a tool bar or menu).
- Placing the cursor over a menu when dragging opens it, allowing you to insert the command anywhere in the menu.
- Commands can be placed in menus or tool bars. If you created you own toolbar you can populate it with your own commands/icons.

Please note:

You can also edit the commands in the **context menus** (right-click anywhere to open the context menu), using the same method. Click the **Menu** tab and then select the specific context menu available in the Context Menus combo box.

To delete a command or menu:

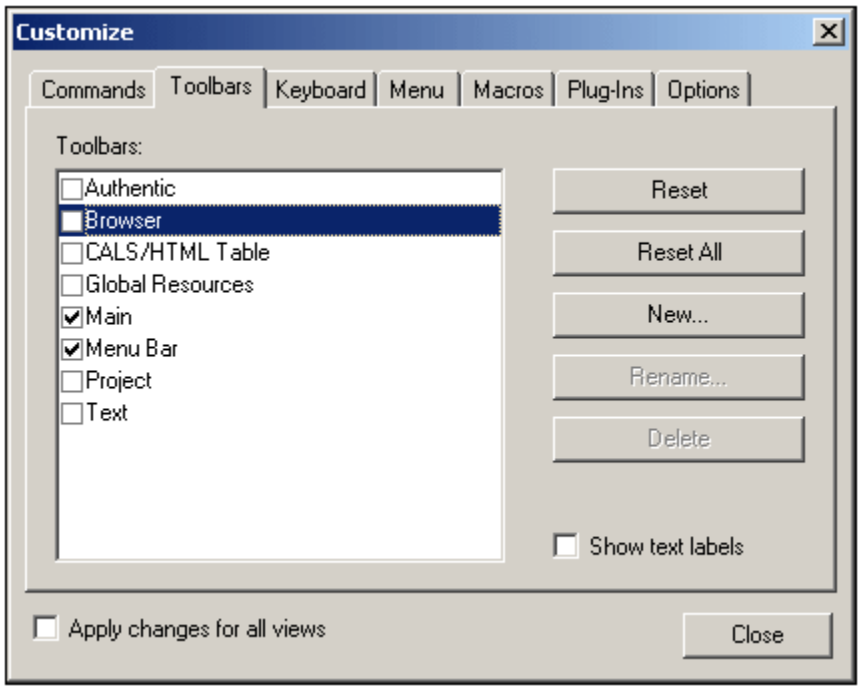
1. Select the menu item **Tools | Customize**. The Customize dialog appears.
2. Click on the menu entry or icon you want to delete, and drag with the mouse.
3. Release the mouse button whenever the "x" icon appears below the mouse pointer. The command, or menu item, is deleted from the menu or tool bar.

Toolbars

The **Toolbars** tab allows you to activate or deactivate specific toolbars, as well as create your own specialized ones.

Authentic Desktop toolbars contain symbols for the most frequently used menu commands. For each symbol you get a brief "tool tip" explanation when the mouse cursor is directly over the item and the status bar shows a more detailed description of the command.

You can drag the toolbars from their standard position to any location on the screen, where they appear as a floating window. Alternatively you can also dock them to the left or right edge of the main window.



To activate or deactivate a toolbar

- Click the check box to activate (or deactivate) the specific toolbar.

Apply changes for all views

- Check this check box at the bottom of the dialog to apply changes to all views (and not only to the current view). On clicking **Close**, all changes that were made **after** checking the check box will be applied to all views.

To create a new toolbar

1. Click the **New...** button, and give the toolbar a name in the Toolbar name dialog box.

2. Drag commands to the toolbar in the **Commands** tab of the Customize dialog box.

To reset the Menu Bar

1. Click the Menu Bar entry.
2. Click the **Reset** button, to reset the menu commands to the state they were in when Authentic Desktop was installed.

To reset all toolbar and menu commands

1. Click the **Reset All** button, to reset all the toolbar commands to the state they were when the program was installed. A prompt appears stating that all toolbars and menus will be reset.
2. Click **Yes** to confirm the reset.

To change a toolbar name

- Click the **Rename**... button to edit the name of the toolbar.

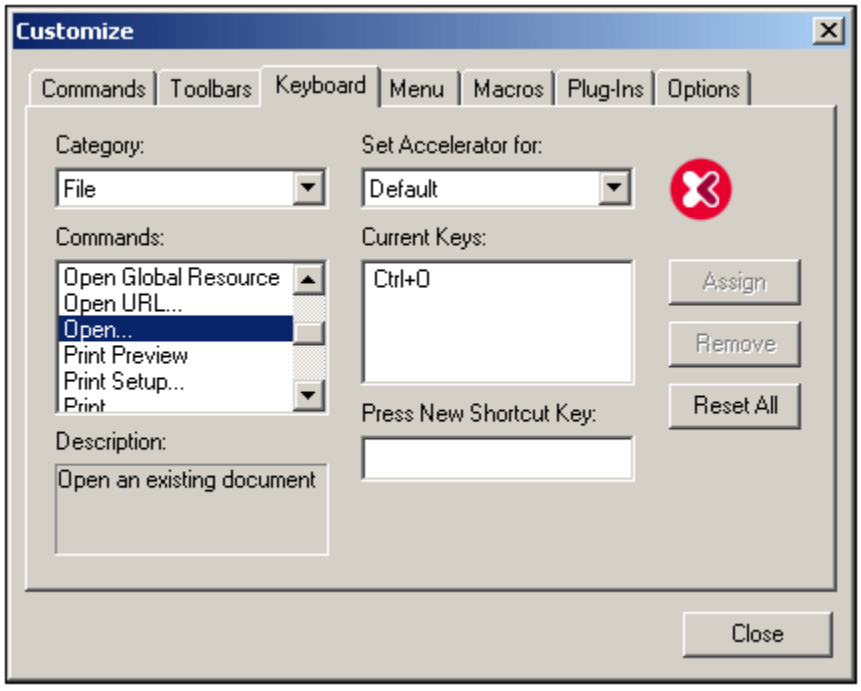
To delete a toolbar

1. Select the toolbar you want to delete in the Toolbars list box.
2. Click the **Delete** button.
3. A prompt appears, asking if you really want to delete the toolbar. Click **Yes** to confirm the deletion.

Show text labels
This option displays explanatory text below toolbar icons when activated.

Keyboard

The **Keyboard** tab allows you to define (or change) keyboard shortcuts for any Authentic Desktop command.



To assign a new Shortcut to a command:

1. Select the All Commands category using the **Category** combo box. Note that if a macro has been selected as an Associated Command, then macros can also be selected via the category combo box and a shortcut for the macro can be set.
2. Select the **command** you want to assign a new shortcut to, in the Commands list box
3. Click in the **Press New Shortcut Key**: text box, and press the shortcut keys that are to activate the command.
   The shortcuts appear immediately in the text box. If the shortcut was assigned previously, then that function is displayed below the text box.
4. Click the **Assign** button to assign the shortcut.
   The shortcut now appears in the Current Keys list box.
   (To **clear** this text box, press any of the control keys, **CTRL**, **ALT** or **SHIFT**).

To de-assign or delete a shortcut:

1. Click the shortcut you want to delete in the Current Keys list box.
2. Click the **Remove** button.
3. Click the **Close** button to confirm.

Set accelerator for:

Currently no function.

Currently assigned keyboard shortcuts:

Hotkeys by key	
F1	Help Menu
F3	Find Next
F5	Refresh
F7	Check well-formedness
F8	Validate
F10	XSL Transformation
CTRL+F10	XSL:FO Transformation
F11	Step into
CTRL+F11	Step Over
Shift + F11	Step Out
Num +	Expand
Num -	Collapse
Num *	Expand fully
CTRL+Num-	Collapse unselected
CTRL + G	Goto line/char
CTRL+TAB	Switches between open documents
CTRL+F6	Cycle through open windows
Arrow keys (up / down)	Move selection bar
Esc.	Abandon edits/close dialog box
Return/Space bar	confirms a selection
Alt + F4	Closes Authentic Desktop
CTRL + F4	Closes active window
Alt + F, 1	Open last file
CTRL + N	File New
CTRL + O	File Open
CTRL + S	File Save
CTRL + P	File Print
CTRL + A	Select All
Shift + Del	Cut (or CTRL + X)
CTRL + C	Copy
CTRL + V	Paste
CTRL + Z	Undo
CTRL + Y	Redo
Del	Delete
CTRL + F	Find
F3	Find Next
CTRL + H	Replace

CTRL + I	Append Attribute
CTRL + E	In Grid View, Append Element. in Text View, Jump to Start/End Tag when cursor is in other member of the pair
CTRL + T	Append Text
CTRL + D	Append CDATA
CTRL + M	Append Comment
CTRL + SHIFT + I	Insert Attribute
CTRL + SHIFT + E	Insert Element
CTRL + SHIFT + T	Insert Text content
CTRL + SHIFT + D	Insert CDATA
CTRL + SHIFT + M	Insert Comment
CTRL + ALT + I	Add Child Attribute
CTRL + ALT + E	Add Child Element
CTRL + ALT + T	Add Child Text
CTRL + ALT + D	Add Child CDATA
CTRL + ALT + M	Add Child Comment

Hotkeys for Text View

CTRL + "+"	Zoom In
CTRL + "-"	Zoom Out
CTRL + 0	Reset Zoom
CTRL + mouse wheel forward	Zoom In
CTRL + mouse wheel back	Zoom Out

Currently assigned keyboard shortcuts:

Hotkeys by function

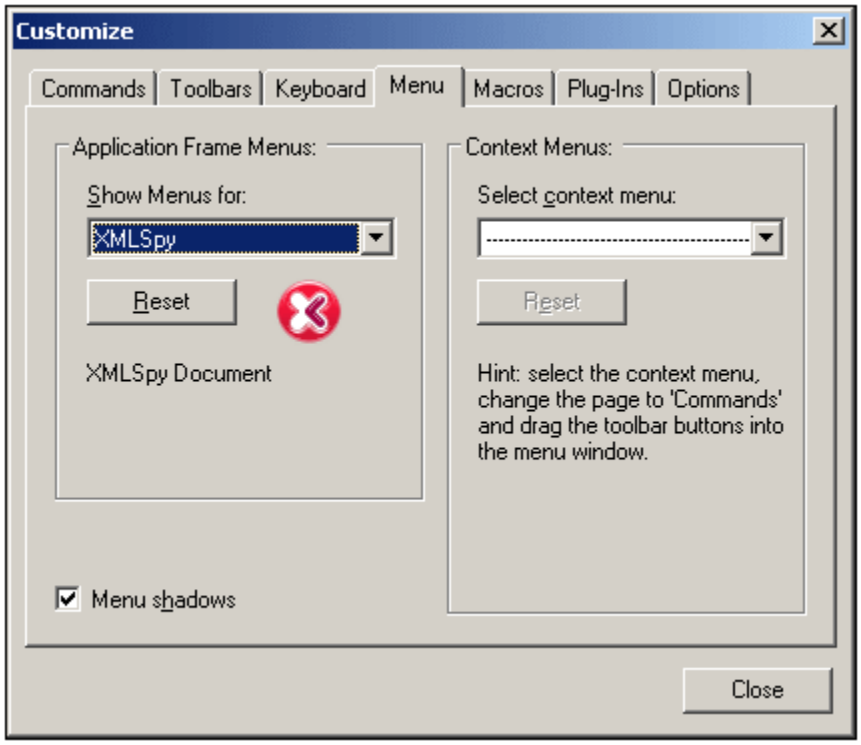
Abandon edits	Esc.
Add Child Attribute	CTRL + ALT + I
Add Child CDATA	CTRL + ALT + D
Add Child Comment	CTRL + ALT + M
Add Child Element	CTRL + ALT + E
Add Child Text	CTRL + ALT + T
Append Attribute	CTRL + I
Append CDATA	CTRL + D
Append Comment	CTRL + M
Append Element	CTRL + E (Grid View)
Append Text	CTRL + T
Check well-formedness	F7
Closes active window	CTRL + F4
Close Authentic Desktop	Alt + F4
Collapse	Num -
Collapse unselected	CTRL + Num-
Confirms a selection	Return / Space bar
Copy	CTRL + C
Cut	SHIFT + Del (or CTRL + X)
Cycle through windows	CTRL + TAB and CTRL + F6
Delete item	Del
Expand	Num +
Expand fully	Num *
File New	CTRL + N
File Open	CTRL + O
File Print	CTRL + P
File Save	CTRL + S
Find	CTRL + F
Find Next	F3
Goto line/char	CTRL + G
Help Menu	F1
Highlight other tag in pair when cursor is inside a start or end element tag	CTRL + E (Text View)
Insert Attribute	CTRL + SHIFT + I
Insert CDATA	CTRL + SHIFT + D
Insert Comment	CTRL + SHIFT + M
Insert Element	CTRL + SHIFT + E
Insert Text content	CTRL + SHIFT + T
Move selection bar	Arrow keys (up / down)
Open last file	Alt + F, 1
Paste	CTRL + V
Redo	CTRL + Y
Refresh	F5
Replace	CTRL + H
Select All	CTRL + A
Start Debugger/Go	Alt + F11

Step Into	F11
Step Out	Shift + F11
Step Over	CTRL + F11
To view an element definition	CTRL + Double click on an element.
Undo	CTRL + Z
Validate	F8
XSL Transformation	F10
XSL:FO Transformation	CTRL + F10

In the application, you can see a list of commands, together with their shortcuts and descriptions, in the Keyboard Map dialog (**Help | Keyboard Map**).

Menu

The **Menu** tab allows you to customize the main menu bars as well as the (popup - right click) context menus.



You can customize both the Default and Authentic Desktop menu bars.

The **Default** menu (*screenshot below*) is the one visible when no XML documents of any type are open in Authentic Desktop.

The Authentic Desktop menu is the menu bar visible when at least one XML document has been opened.

To customize a menu:

1. Select the menu bar you want to customize from the **Show Menus for:** combo box

2. Click the **Commands** tab, and drag the commands to the menu bar of your choice.

To delete commands from a menu:

1. Click right on the command, or icon representing the command.
2. Select the **Delete** option from the popup menu,

or,

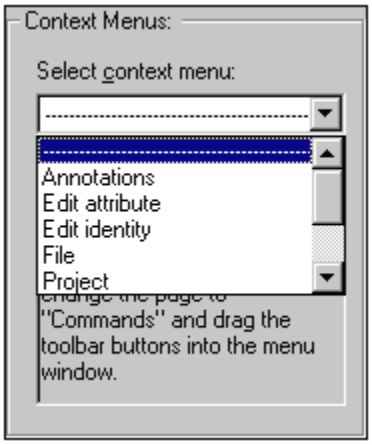
1. Select **Tools | Customize** to open the Customize dialog box.
2. Drag the command away from the menu, and drop it as soon as the check mark icon appears below the mouse pointer.

To reset either of the menu bars:

1. Select either the Default or Authentic Desktop entry in the **Show Menus for** combo box.
2. Click the **Reset** button just below the menu name.
   A prompt appears asking if you are sure you want to reset the menu bar.

To customize any of the Context menus (right-click menus):

1. Select the context menu from the **Select context menu** combo box. The context menu you selected appears.
2. Click the **Commands** tab, and drag the commands to the context menu.



To delete commands from a context menu:

1. Click right on the command, or icon representing the command.
2. Select the **Delete** option from the popup menu

or,

1. Select **Tools | Customize** to open the Customize dialog box.
2. Drag the command away from the context menu, and drop it as soon as the check mark icon appears below the mouse pointer.

To reset any of the context menus:

1. Select the context menu from the combo box, and
2. Click the **Reset** button just below the context menu name.
   A prompt appears asking if you are sure you want to reset the context menu.

To close a context menu window:

- Click on the **Close icon** at the top right of the title bar

or

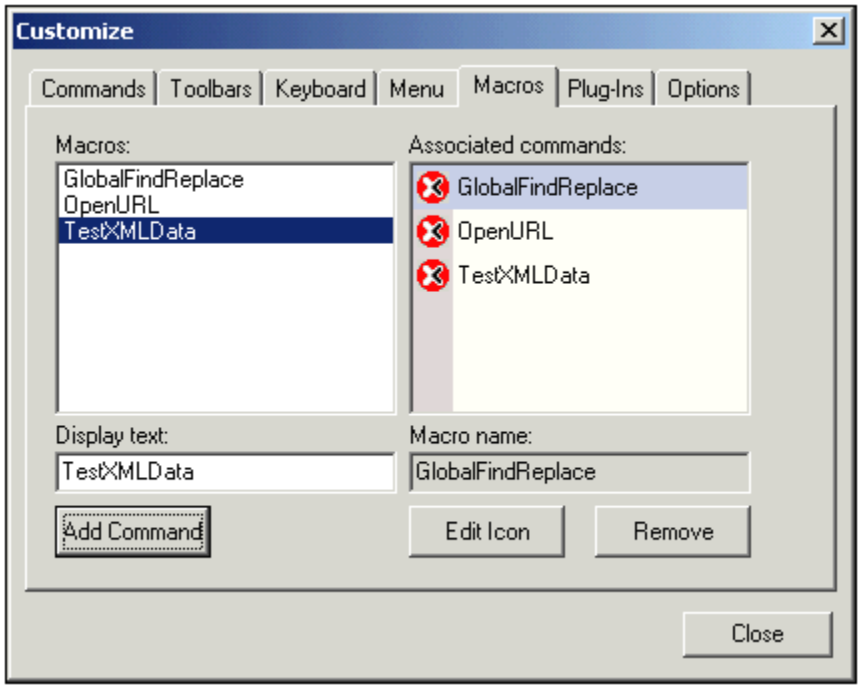
- Click the Close button of the Customize dialog box.

Menu shadows

- Click the **Menu shadows** check box, if you want all your menus to have shadows.

Macros

The **Macros** tab allows you to place macros (created using the XML scripting environment) in a toolbar or menu.



To place a macro (icon) into a toolbar or menu:

1. Start the scripting environment using **Tools | Switch to Scripting environment**.
2. Double-click the XMLSpyMacros entry in the Modules folder of the **XMLSpyGlobalScripts** project.
   Previously defined macros will then be visible in the right hand window.
3. Switch back to Authentic Desktop, and select **Tools | Customize** and click on the **Macros** tab.
   The macros defined in the scripting environment are now visible in the Macros list box at left.
4. Click the macro name and then the **Add Command** button. This places the macro name in the Associated Commands list box.
5. Click the macro name in the Associated Commands list box, and drag it to any tool bar or menu.

To edit a macro icon:

- Click the **Edit Icon** button.

To delete a macro from the Associated commands list box:

- Click the **Remove** button.

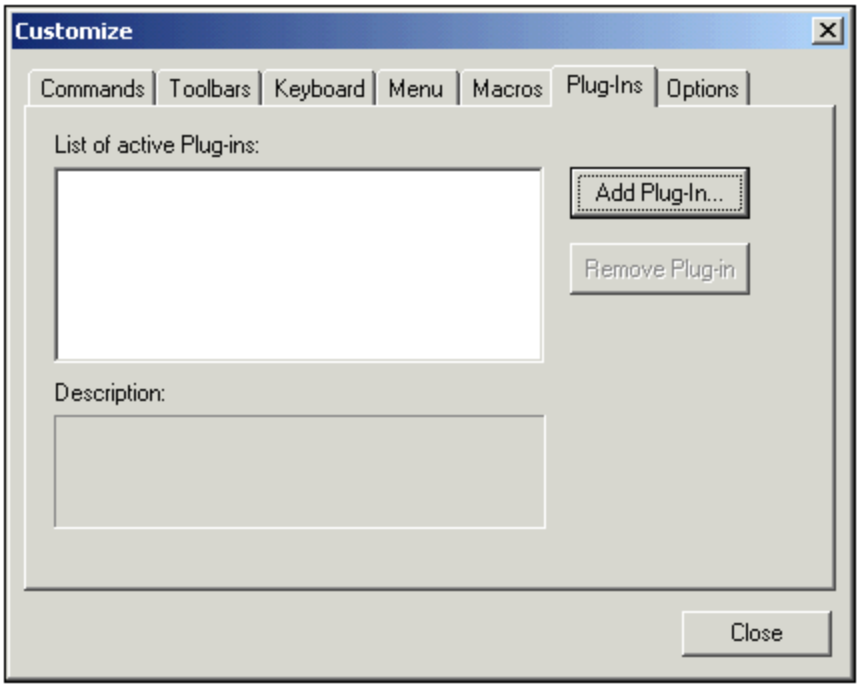
Note: If a macro has been set as an Associated Command, you can set a keyboard shortcut for it. In the Keyboard tab of the Customize dialog, select Macros in the Category combo box, then select the required macro, and set the shortcut. You should set a macro as an associated command in order for Macros to be available for keyboard shortcuts.

Plug-Ins

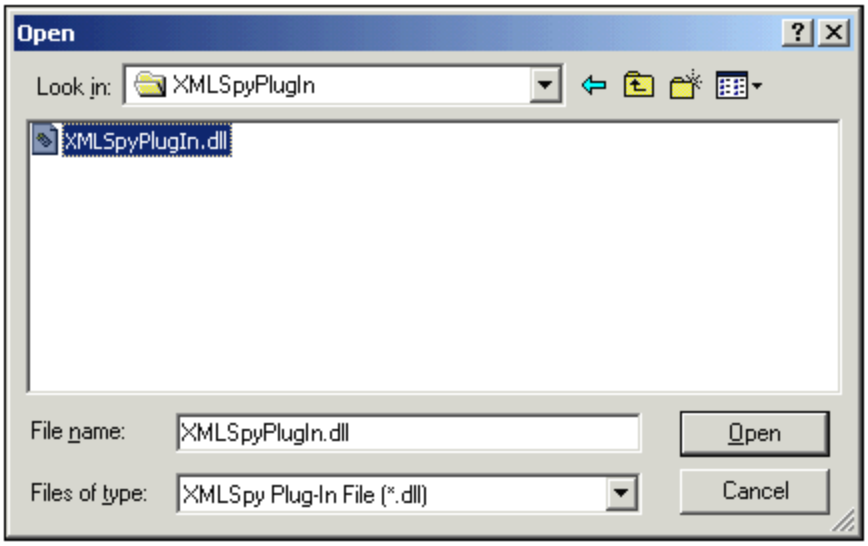
The Plug-Ins tab allows you to place plug-ins in a toolbar or menu.

To place a plug-in icon into a toolbar or menu:

1. Click the **Add Plug-In...** button.



2. Select the folder and then click the plug-in file to mark it (`XMLSpyPlugIn.dll` in this case).



3. Click the **Open** button to install the plug-in.
   The plug-in name appears in the "list of active plug-ins" list, and the plug-in icon(s) appears in a new toolbar.

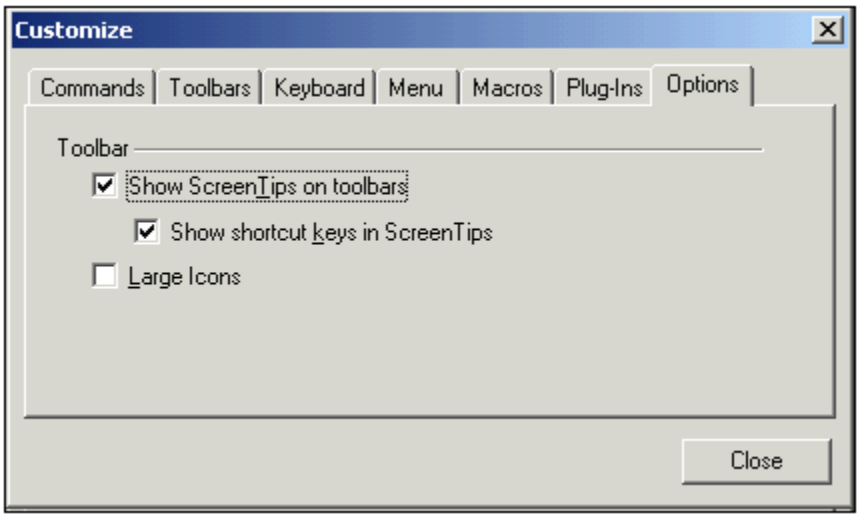
To remove a plug-in:

- Click the plug-in name in the "List of active plug-ins" list and click the "Remove Plug-in" button

.

Options

The Options tab allows you to set general environment settings.



Toolbar
When active, the **Show ScreenTips on toolbars** check box displays a popup when the mouse pointer is placed over an icon in any of the icon bars. The popup contains a short description of the icon function, as well as the associated keyboard shortcut, if one has been assigned.

The **Show shortcut keys in ScreenTips** check box, allows you to decide if you want to have the shortcut displayed in the tooltip.

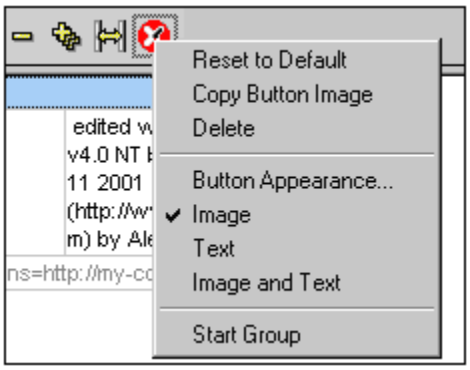
When active, the **Large icons** check box switches between the standard size icons, and larger versions of the icons.

Customize Context Menu

The Customize context menu allows you to further customize icons and menu items.

To open the customize context menu:

1. First open the Customize dialog by selecting **Tools | Customize**.
2. Then place the mouse pointer over an icon (or menu) and **click right** to open the context menu.



Reset to default
Currently no function.

Copy Button image
This option copies the icon you right-click to the clipboard.

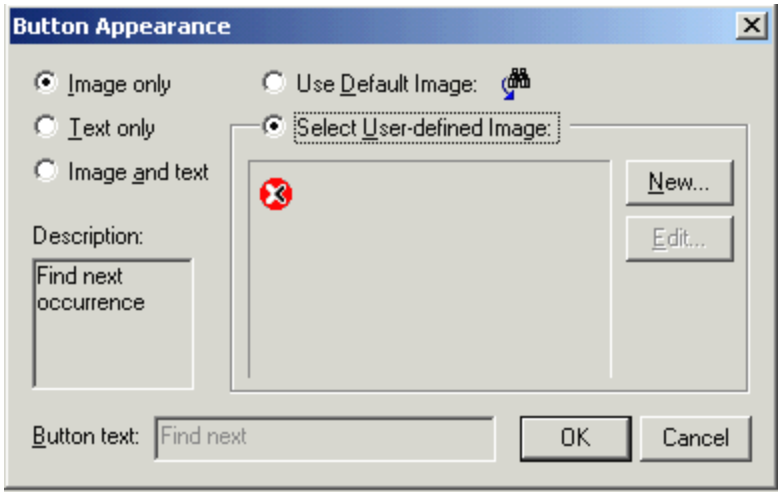
Delete
This option deletes the icon or menu you right click. Deleting a menu deletes all menu options contained in it!

To restore a deleted menu:

1. Select the **Menu** tab in the Customize dialog.
2. Select the menu you want to restore (Authentic Desktop or Default).
3. Click the **Reset** button below the menu selection combo box.

Button Appearance...
This option allows you to edit the button image as well as the button text. Currently only the macro icons can be edited (default Authentic Desktop icon).

The Image only, Text only and Image and text radio buttons, let you define what you want to edit. Click the Select User-defined Image radio button and click one of the icons.



Click the **Edit...** button, to open the Edit button image dialog box. The Button text can be edited if you select either **Text only** or **Image and text** options.

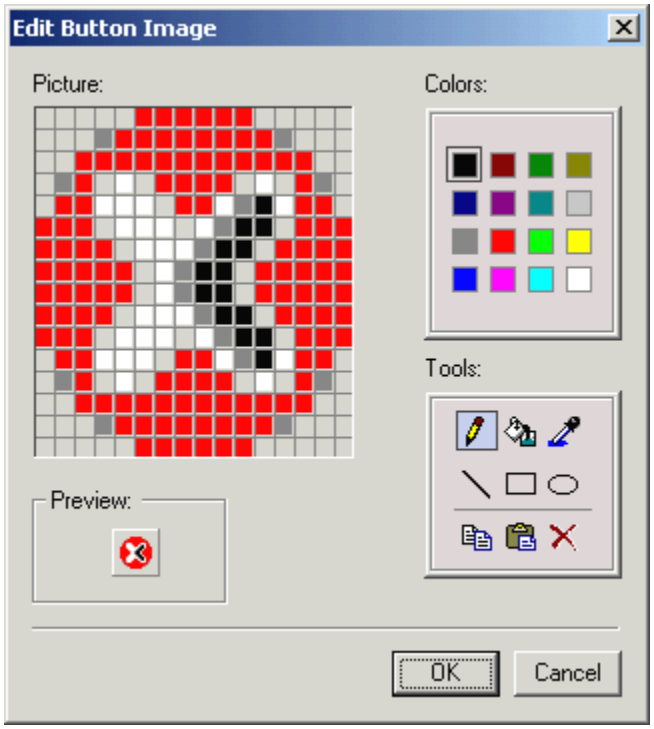


Image
This option changes the textual description of a function to the graphical image (icon) representing that function.

Text
This option changes the graphical display of an icon to the text representing it.

Image and Text
This option enables the simultaneous display of an icon and its text.

Start group
This option inserts a vertical divider to the left of the icon you right clicked.

11.9.8 Options

The **Tools** | **Options** command enables you to define global application settings. These settings are specified in a tabbed dialog box and saved in the registry. They apply to all current and future document windows. The **Apply** button in the Options dialog displays the changes in the currently open documents and fixes the current settings. The changes are seen immediately in the background windows.

Each tab of the Options dialog is described in detail in this section.

File

The **File** tab defines the way Authentic Desktop opens and saves documents. Related settings are in the Encoding tab.

Open/New file in Grid view
You can choose to open an existing file or create a new file either in Grid View or in Text View. If you select Grid View, you can also choose to automatically expand all lines.

Automatic reload of changed files
If you are working in a multi-user environment, or if you are working on files that are dynamically generated on a server, you can watch for changes to files that are currently open in the interface. Each time Authentic Desktop detects a change in an open document, it will prompt you about whether you want to reload the changed file.

Validation
If you are using DTDs or schemas to define the structure of your XML documents, you can automatically check the document for validity whenever it is opened or saved. During Open and Save operations, you have the option of validating files only if the file-size is less than a size you specify in MB. If the document is not valid, an error message will be displayed. If it is valid, no message will be displayed and the operation will proceed without any notification. Authentic Desktop can also cache these files in memory to save any unnecessary reloading (e.g. when the schema being referred to is accessed through a URL). If your schema location declaration uses an URL, disable the "cache DTD/Schema files in memory" option to have changes made to the schema appear immediately, and not use the cached version of the schema.

Project
When you start Authentic Desktop, you can open the last-used project automatically.

Save File
When saving an XML document, Authentic Desktop includes a short comment `<!-- Edited with Authentic Desktop http://www.altova.com -->` near the top of the file. This option can only be deactivated by licensed users, and takes effect when editing or saving files in the Enhanced Grid or Schema Design View.

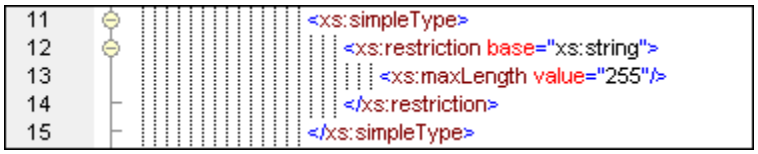
If a StyleVision Power Stylesheet is associated with an XML file, the 'Authentic: save link to design file' option will cause the link to the StyleVision Power Stylesheet to be saved with the XML file.

Line breaks
When you open a file, the character coding for line breaks in it are preserved if **Preserve old** is selected. Alternatively, you can choose to code line breaks in any of three codings: **CR&LF** (for PC), **CR** (for MacOS), or **LF** (for Unix).

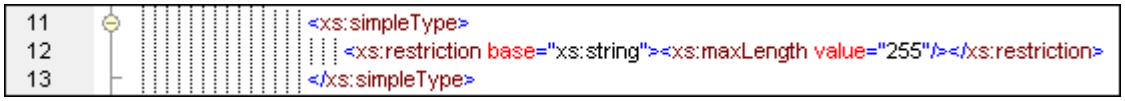
No output formatting for
In Text View, the indentation of an element can be made to reflect its position in the element

hierarchy (*see* **Save File**). You can, however, override this indentation for individual elements. To do this, enter the element name in the **No output formatting for** field. All elements entered in this field will be formatted such that their descendant elements have no whitespace between them (*see screenshots*).

Hierarchical indentation for all elements:



No output formatting has been specified for element `xs:restriction`:



After making the settings, click **OK** to finish and close the Options dialog.

File Types

The **File types** tab 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 to customize the functions for that particular file type:

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 editing and other features for various file types. The features for a file type are set by specifying the conformance 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. The screenshot above shows the Filetypes tab of the Enterprise edition. If your edition is not the Enterprise edition, it will have fewer views than shown in the screenshot.

Text View
This check box lets you set syntax-coloring for particular file types.

Disable automatic validation
This option enables you to disable automatic validation per file type. Automatic validation typically takes place when a file is opened or saved, or when a view is changed.

Save empty elements in short <E/> format
Some applications that use XML documents or output generated from XML documents may have problems understanding the short `<Element/>` form for empty elements defined in the XML 1.0 Specification. You can instruct Authentic Desktop to save elements in the longer (but

also valid) <Element></Element> form.

Add new file extension
Adds a new file type to the File types list. You must then define the settings for this new file type using the other options in this tab.

Delete selected file extension
Deletes the currently selected file type and all its associated settings.

After making the settings, click **OK** to finish and close the Options dialog.

View

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

Pretty-print
When you select **Edit | Pretty-Print XML Text** in Text View or switch from another view to Text View, the XML document will be "pretty-printed". The pretty-printing will be with or without indentation according to whether the *Use Indentation* option in this dialog is checked or not. The amount of indentation can be specified in the Tabs pane of the Text View Settings dialog.

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.

After modifying the options settings, click **OK** to finish and close the Options dialog.

Encoding

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

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** command.

Open XML files with unknown encoding as
If the encoding of an XML file cannot be determined or if the XML document has no encoding specification, the file will be opened with the encoding you select in this combo box.

Open non-XML files in

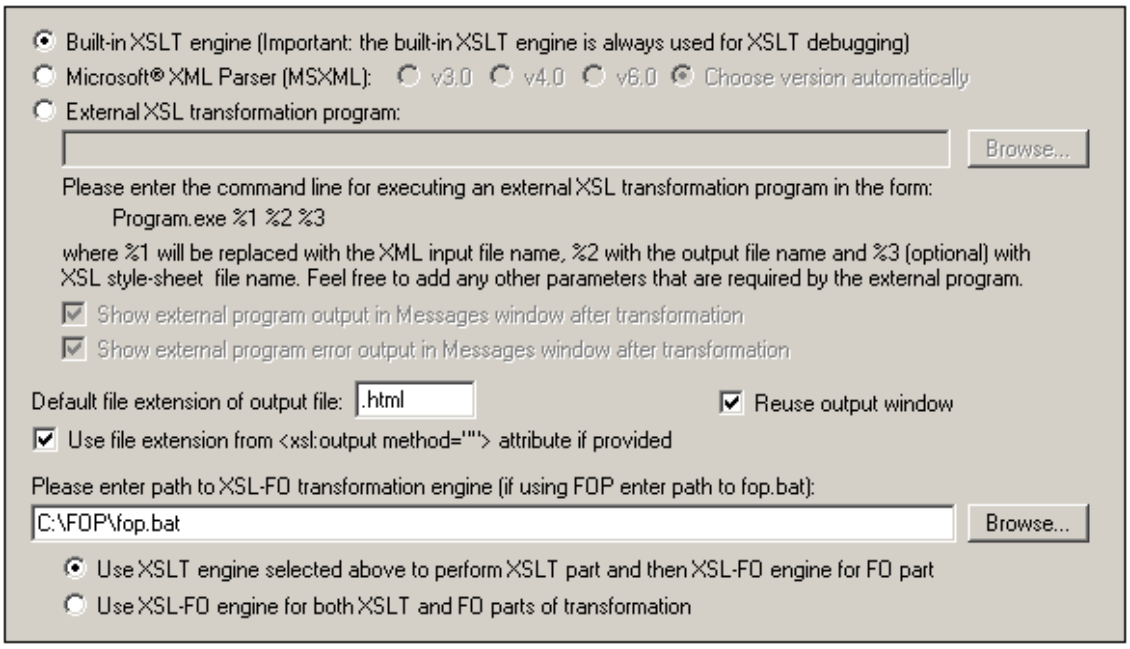
Existing and new non-XML files are opened with the encoding you select in this combo box. You can change the encoding of the document by using the **File | Encoding** command.

BOM (Byte Order Mark)
When a document with two-byte or four-byte character encoding is saved, the document can be saved either with (i) little-endian byte-ordering and a little-endian BOM (*Always create BOM if not UTF-8*); or (ii) the detected byte-ordering and the detected BOM (*Preserve detected BOM on saving*).

After making the settings, click **OK** to finish and close the Options dialog.

XSL

The **XSL** tab (*screenshot below*) enables you to define options for XSLT transformations and XSL-FO transformations carried out from within the application.



XSLT transformations
Authentic Desktop contains the Altova XSLT 1.0 Engine and Altova XSLT 2.0 Engine, which you can use for XSLT transformations. The appropriate XSLT engine (1.0 or 2.0) is used (according to the value of the `version` attribute of the `xsl:stylesheet` or `xsl:transform` element).

For transforming XML documents using XSLT, you could use one of the following:

- The built-in Altova XSLT Engine (comprising the Altova XSLT 1.0 Engine and the Altova XSLT 2.0 Engine).
- The MSXML 3.0, 4.0, or 6.0 parser (which is pre-installed). If you know which version of the MSXML parser is running on your machine, you could select it; otherwise, you should let the application select the version automatically. (The *Choose version automatically* option is active by default.) In this case, the application tries to select the most recent available version.
- An external XSLT processor of your choice. You must specify the command line string for the external XSLT processor. The following variables are available for building the

command line string:

`%1` = XML document to process
`%2` = Output file to generate
`%3` = XSLT stylesheet to use (if the XML document does not contain a reference to a stylesheet)

For example, the command to run a simple transformation with the Saxon (XSLT 1.0) processor is:

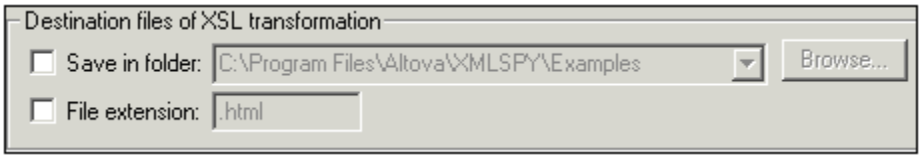
```
saxon.exe -o output.xml input.xml stylesheet.xslt
parameter-name=parameter-value
```

To run this command from the application, select the External XSL Transformation Program radio button, and enter the following line in the text box:

```
c:\saxon\saxon.exe -o %2 %1 %3 parameter-name=parameter-value
```

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

The *Reuse output window* option causes subsequent transformations to display the result document in the same output window. If the XML file belongs to a project and *Reuse output window* option is disabled, the setting only takes effect if the *Save in folder* output file path (*screenshot below*) in the relevant project properties is **also** disabled.

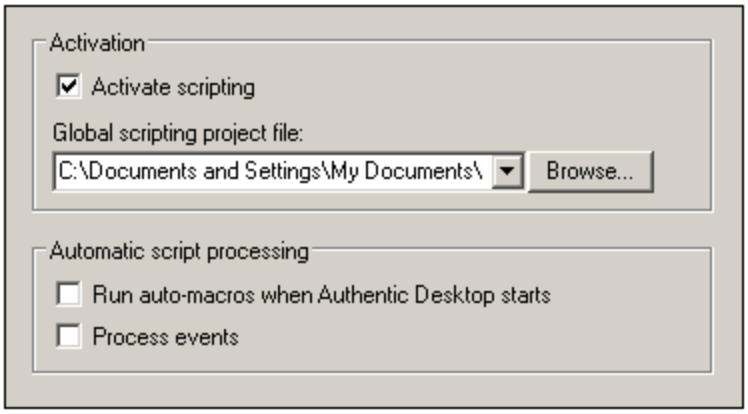


XSL-FO transformations
FO documents are processed using an FO processor, and the path to the executable of the FO processor must be specified in the text box for the XSL-FO transformation engine. The transformation is carried out using the **XSL/XQuery | XSL-FO Transformation** menu command. If the source file (the active document when the command is executed in the IDE) is an XSL-FO document, the FO processor is invoked for the transformation. If the source document is an XML document, an XSLT transformation is required to first convert the XML document to an XSL-FO document. This XSLT transformation can be carried out either by the XSLT engine you have specified as the default engine for the application (*see above*), or by the XSLT engine that might be built into the FO processor you have specified as the default FO processor for the application. To select between these two options, click the appropriate radio button.

After making the settings, click **OK** to finish and close the Options dialog.

Scripting

The **Scripting** tab (*screenshot below*) allows you to enable the Scripting Environment on application startup. Check the *Activate Scripting* check box to do this. You can then specify the Global Scripting Project file (*see screenshot below*).



To set a global scripting project for Authentic Desktop, check the *Activate Scripting* check box and then browse for the Altova Scripting Project (`.asprj`) file you want. You can also specify: (i) whether Auto-Macros in the scripting project should be automatically executed when Authentic Desktop starts, and (ii) whether application event handler scripts in the project should be automatically executed or not; check or uncheck the respective check boxes accordingly.

Click **OK** when done. Macros in the Global Scripting Project will then be displayed in the submenu of the **Macros** command.

Source Control

The **Source Control** tab (*screenshot below*) enables you to specify the source control provider, and the settings and default logon ID for each source control provider.

Source Control Plugin
The current source control plugin can be selected from among the currently installed source control systems. These systems are listed in the dropdown list of the combo box. After selecting the required source control, specify the login ID for it in the next text box. The **Advanced** button pops up a dialog specific to the selected source control plugin, in which you can define settings for that source control plugin. These settings are different for different source control plugins.

User preferences
A range of user preferences is available, including the following:

- Status updates can be performed in the background after a user-defined interval of time, or they can be switched off entirely. Very large source control databases could consume considerable CPU and network resources. The system can be speeded up, however, by disabling background status updates or increasing the interval between them..
- When opening and closing projects, files can be automatically checked out and checked in, respectively.
- The display of the Check Out and Check In dialogs can be suppressed.
- The **Reset** button is enabled if you have checked/activated the *Don't show this again* option in one of the dialog boxes. On clicking the **Reset** button, the *Don't show this again* prompt is re-enabled.

After making the settings, click **OK** to finish and close the Options dialog.

11.10 Window Menu

To organize the individual document windows in an Authentic Desktop session, the **Window** menu contains standard commands common to most Windows applications.

You can cascade the open document windows, tile them, or arrange document icons once you have minimized them. You can also switch the various Entry Helper windows on or off, or switch to an open document window directly from the menu.

11.10.1 Cascade

This command rearranges all open document windows so that they are all cascaded (i.e. staggered) on top of each other.

11.10.2 Tile Horizontally

This command rearranges all open document windows as **horizontal tiles**, making them all visible at the same time.

11.10.3 Tile Vertically

This command rearranges all open document windows as **vertical tiles**, making them all visible at the same time.

11.10.4 Project Window

This command lets you switch the Project Window on or off.

This is a dockable window. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar, to allow docking or hide the window.

11.10.5 Info Window

This command lets you switch the Info Window on or off.

This is a dockable window. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar, to allow docking or hide the window.

11.10.6 Entry Helpers

This command lets you switch all three Entry-Helper Windows on or off.

All three Entry helpers are dockable windows. Dragging on a title bar detaches it from its current position and makes it a floating window. Click right on the title bar to allow docking or hide the window.

11.10.7 Output Windows

The Output Windows are a set of tabbed output windows, such as the Messages window (which displays messages like validation results), the Find in Files window, and the XPath window (which shows XPath evaluation results). The initial setting is for them to open at below the Main Window. The Output Windows command lets you switch the Output Windows on or off.

The Output Windows window is dockable. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar to allow docking or to hide the window.

For a complete description of Output Windows see Output Windows in the section, Text View.

11.10.8 Project and Entry Helpers

This command toggles on and off the display of the Project Window and the Entry Helpers together.

11.10.9 All On/Off

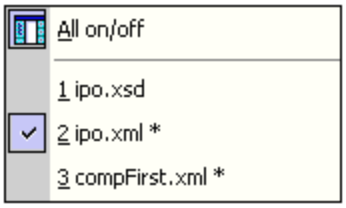
This command lets you switch all dockable windows on, or off:

- the Project Window
- the Info Window
- the three Entry-Helper Windows
- the Output Windows

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

11.10.10 Currently Open Window List

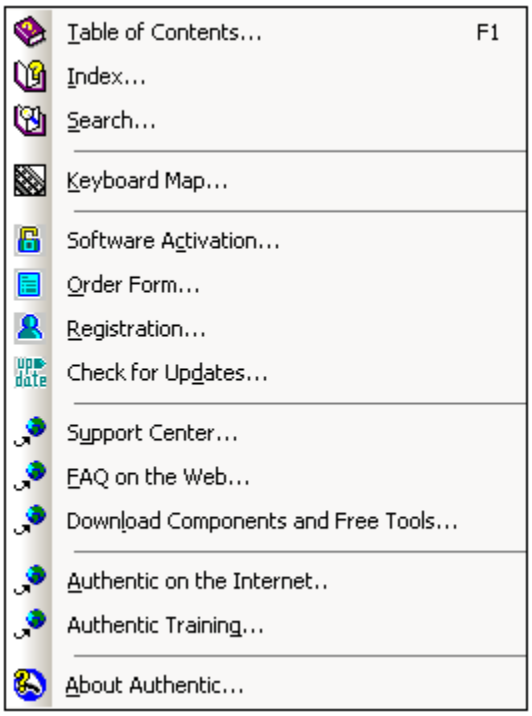
This list shows all currently open windows, and lets you quickly switch between them.



You can also use the Ctrl-TAB or CTRL F6 keyboard shortcuts to cycle through the open windows.

11.11 Help Menu

The **Help** menu contains all commands required to get help or more information on Authentic Desktop, as well as links to information and support pages on our web server.



The **Help** menu also contains the Registration dialog, which lets you enter your license key-code once you have purchased the product.

11.11.1 Table of Contents

The **Help | Table of contents** command displays a **hierarchical representation** of all chapters and topics contained in the online help system. Use this command to jump to the table of contents directly from within Authentic Desktop.

Once the help window is open, use the three tabs to toggle between the table of contents, index , and search panes. The Favorites tab lets you bookmark certain pages within the help system.

11.11.2 Index

The **Help | Index** command accesses the **keyword index** of the Online Help. You can also use the Index tab in the left pane of the online help system.

The index lists all relevant keywords and lets you navigate to a topic by double-clicking the respective keyword. If more than one topic matches the selected keyword, you are presented a list of available topics to choose from.

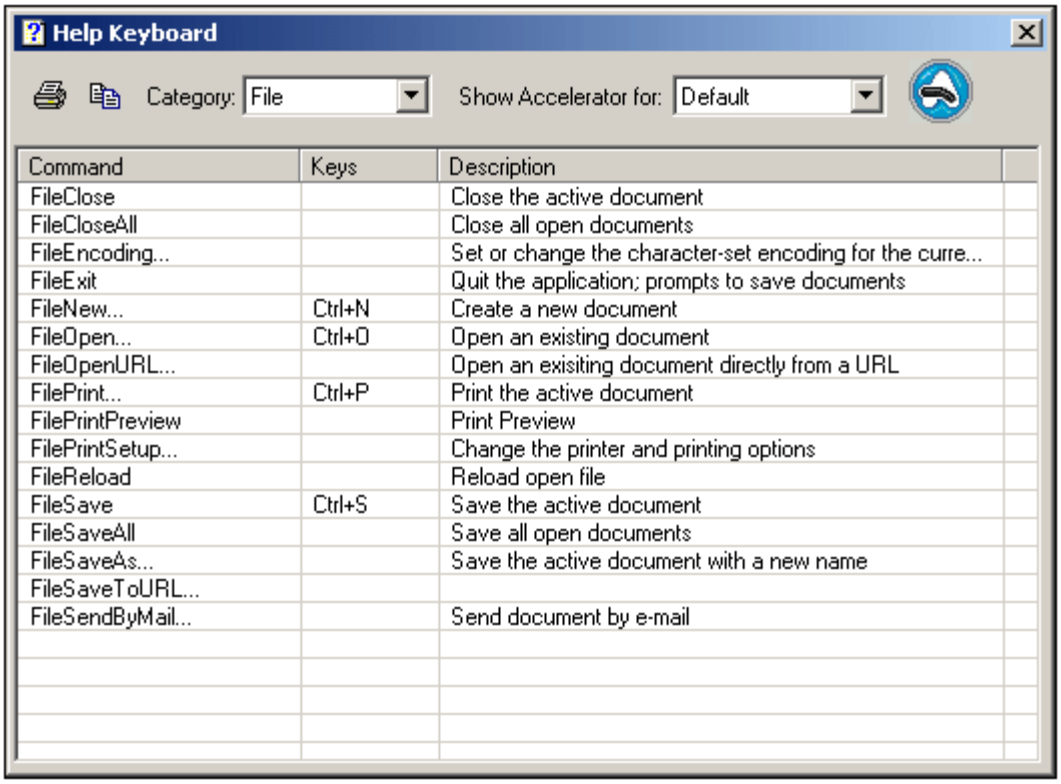
11.11.3 Search

The **Help | Search** command performs a **full-text search** on the entire online help system.

1. Enter your search term in the query field and press **Enter**.
   The online help system displays a list of available topics that contain the search term you've entered.
2. Double-click on any item in the list to display the corresponding topic.

11.11.4 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.



To view commands in a particular menu, select the menu name in the Category combo box. You can print the command by clicking the printer icon.

11.11.5 Activation, Order Form, Registration, Updates

Software Activation
After you download your Altova product software, you can activate it using either a free evaluation key or a purchased permanent license key.

- ***Free evaluation key.*** When you first start the software after downloading and installing it, the Software Activation dialog will pop up. In it is a button to request a free evaluation key-code. Enter your name, company, and e-mail address in the dialog that appears, and click Request Now! The evaluation key is sent to the e-mail address you entered and should reach you in a few minutes. Now enter the key in the key-code field of the Software Activation dialog box and click **OK** to start working with your Altova product. The software will be unlocked for a period of 30 days.
- ***Permanent license key.*** The Software Activation dialog contains a button to purchase a permanent license key. Clicking this button takes you to Altova's online shop, where you can purchase a permanent license key for your product. There are two types of permanent license: single-user and multi-user. Both will be sent to you by e-mail. A *single-user license* contains your license-data and includes your name, company, e-mail, and key-code.A *multi-user license* contains your license-data and includes your company name and key-code. Note that your license agreement does not allow you to install more than the licensed number of copies of your Altova software on the computers in your organization (per-seat license). Please make sure that you enter the data required in the registration dialog exactly as given in your license e-mail.

Note: When you enter your license information in the Software Activation dialog, ensure that you enter the data exactly as given in your license e-mail. For multi-user licenses, each user should enter his or her own name in the Name field.

The Software Activation dialog can be accessed at any time by clicking the **Help | Software Activation** command.

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

Registration
The first time you start your Altova software after having activated it, a dialog appears asking whether you would like to register your product. There are three buttons in this dialog:

- **OK:** Takes you to the Registration Form
- **Remind Me Later:** Pops up a dialog in which you can select when you wish to be next reminded.
- **Cancel:** Closes the dialog and suppresses it in future. If you wish to register at a later time, you can use the **Help | Registration** command.

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

11.11.6 Support Center, FAQ, Downloads

Support Center
If you have any questions regarding our product, please feel free to use this command to send a query to the Altova Support Center at any time. This is the place where you'll find links to the FAQ, support form, and e-mail addresses for contacting our support staff directly.

FAQ
To help you in getting the best support possible, we are providing a list of Frequently Asked Questions (FAQ) on the Internet, that is constantly updated as our support staff encounters new issues that are raised by our customers.

Please make sure to check the FAQ before contacting our technical support team. This will allow you to get help more quickly.

We regret that we are not able to offer technical support by phone at this time, but our support staff will typically answer your e-mail requests within one business day.

If you would like to make a feature suggestion for a future version of Authentic Desktop or if you wish to send us any other general feedback, please use the questionnaire form.

Download components and free tools
This command is a link to the Components Download page at the Altova website, from where you can download components, free tools, and third-party add-ins that you can use in your XML development environment. Included among these are the Altova Validator, and XSLT and XQuery engines.

11.11.7 On the Internet

On the Internet
This command takes you directly to the Altova web-server http://www.altova.com where you can find out about news, product updates and additional offers from the Altova team.

Training
The **Training** command takes you to the Online Training page on the Altova website. Here you can enroll for online courses to help you develop expertise in using Altova products.

11.11.8 About

This command shows the Authentic Desktop splash screen and copyright information dialog box, which includes the Authentic Desktop logo. If you are using the 64-bit version of Authentic Desktop (available in Enterprise edition only), this is indicated with the suffix (x64) after the application name. There is no suffix for the 32-bit version.

Please note:
This dialog box shows the version number - to find the number of the actual build you are using, please look at the status bar, which always includes the full version and build number.

11.12 Command Line

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

Open a file
Command: `authentic.exe file.xml`
Action: Opens the file, `file.xml`, in Authentic Desktop

Open multiple files
Command: `authentic.exe file1.xml file2.xml`
Action: Opens the files, `file1.xml` and `file2.xml`, in Authentic Desktop

Assign an SPS file to an XML file for Authentic View editing
Command: `authentic.exe myxml.xml /sps mysps.sps`
Action: Opens the file, `myxml.xml` in Authentic View with `mysps.sps` as its SPS file. The `/sps` flag specifies that the SPS file that follows is to be used with the XML file that precedes the `/sps` flag (for Authentic View editing).

Open a new XML template file via an SPS file
Command: `authentic.exe mysps.sps`
Action: Opens a new XML file in Authentic View. The display will be based on the SPS and the new XML file will have a skeletal structure based on the SPS schema. The name of the newly created XML file must be assigned when saving the XML file.

Chapter 3

Programmers' Reference

Programmers' Reference

Authentic Desktop is an Automation Server. It exposes programmable objects to other applications called Automation Clients. As a result, an Automation Client can directly access the objects and functionality that the Automation Server makes available. An Automation Client of XMLSpyAuthentic Desktople, can use the XML validation functionality of Authentic Desktop. Developers can thus enhance their applications with the ready-made functionality of Authentic Desktop.

The programmable objects of Authentic Desktop are made available to Automation Clients via the Application API of Authentic Desktop, which is a COM API. The object model of the API and a complete description of all available objects are provided in this documentation (see the section Application API).

The API can be accessed from within the following environments:

- Scripting Editor
- IDE Plug-ins
- External programs
- ActiveX Integration

Each of these environments is described briefly below.

Scripting Editor: Customizing and modifying Authentic Desktop functionality
You can customize your installation of Authentic Desktop by modifying and adding functionality to it. You can also create Forms for user input and modify the user interface so that it contains new menu commands and toolbar shortcuts. All these features are achieved by writing scripts that interact with objects of the Application API. To aid you in carrying out these tasks efficiently, Authentic Desktop offers you an in-built Scripting Editor. A complete description of the functionality available in the Scripting Editor and how it is to be used is given in the Scripting Editor section of this documentation. The supported programming languages are **JScript** and **VBScript**.

IDE Plug-ins: Creating plug-ins for Authentic Desktop
Authentic Desktop enables you to create your own plug-ins and integrate them into Authentic Desktop. You can do this using Authentic Desktop's special interface for plug-ins. A description of how to create plug-ins is given in the section Authentic Desktop IDE Plug-ins.

An application object gets passed to most methods that must be implemented by an IDE plug-in and gets called by the application. Typical languages used to implement an IDE plug-in are **C#** and **C++**. For more information, see the section Authentic Desktop IDE Plugins.

External programs
Additionally, you can manipulate Authentic Desktop with external scripts. For example, you could write a script to open Authentic Desktop at a given time, then open an XML file in Authentic Desktop, validate the file, and print it out. External scripts would again make use of the Application API to carry out these tasks. For a description of the Application API, see the section Application API.

Using the Application API from outside Authentic Desktop requires an instance of Authentic Desktop to be started first. How this is done depends on the programming language used. See

the section, Programming Languages, for information about individual languages.

Essentially, Authentic Desktop will be started via its COM registration. Then the `Application` object associated with the Authentic Desktop instance is returned. Depending on the COM settings, an object associated with an already running Authentic Desktop can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript and VBScript script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C# is a a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#..
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for Applications, Perl, and Python.

ActiveX Integration

A special case of accessing the Application API is via the Authentic Desktop ActiveX control. This feature is only available if the Authentic Desktop integration package is installed. Every ActiveX Control has a property that returns a corresponding COM object for its underlying functionality. The manager control provides an `Application` object, the document control a `Document` object, and the placeholder object, in cases where it contains the project tree, returns the `Project` object. The methods supported by these objects are exactly as described in the Interfaces section of the Application API. Care must be taken not to use methods that do not make sense in the context of ActiveX control integration. For details see ActiveX Integration.

About Programmers' Reference

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

- Scripting Editor: a user reference for the Scripting Environment available in Authentic Desktop
- IDE Plug-ins: a description of how to create plug-ins for Authentic Desktop
- Application API: a reference for the Application API
- ActiveX Integration: a guide and reference for how to integrate the Authentic Desktop GUI and Authentic Desktop functionality using an ActiveX control

1 Scripting Editor

The Scripting Editor of Authentic Desktop uses the Form Editor components of the Microsoft .NET Framework, and thus provides access to the Microsoft .NET Framework. This means that JScripts and VBScripts not only work with the Authentic Desktop API—which is a COM API and the API of Authentic Desktop—but can also access and use classes of the Microsoft .NET framework.

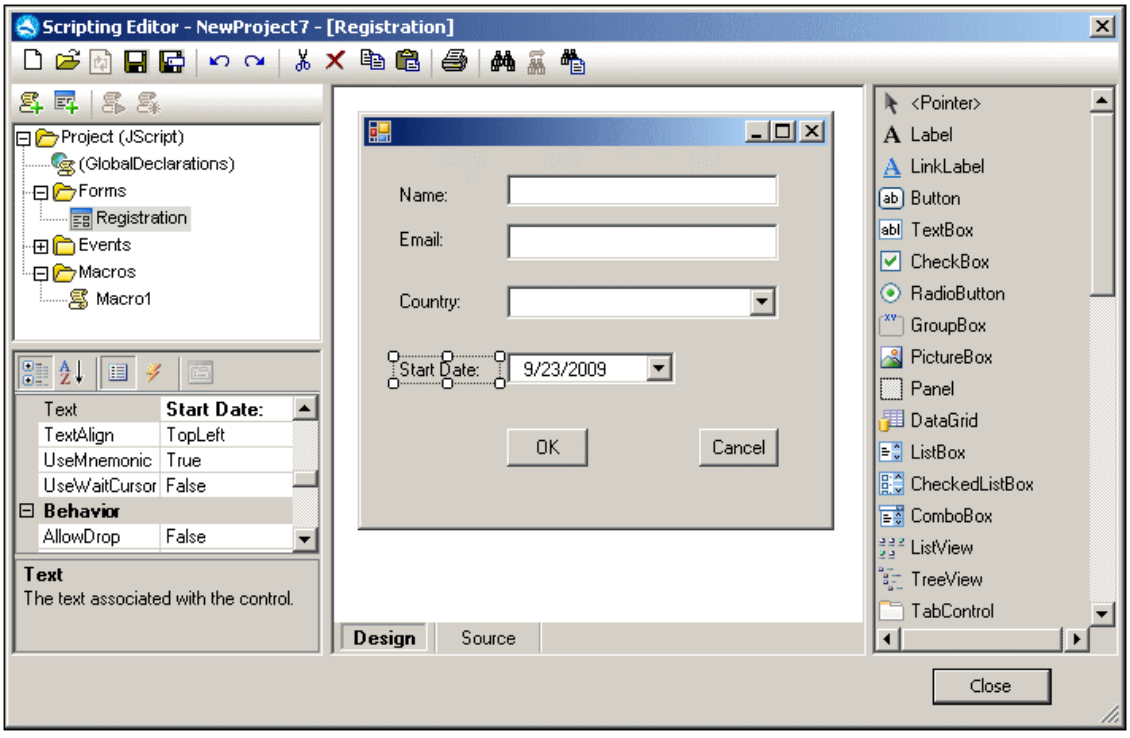
You can therefore create and use your own macros and forms within Authentic Desktop, and thus add to and modify the functionality of your installation of Authentic Desktop.

Note: Visual Basic is **not supported** as a language in the scripting environment. Only VBScript and JavaScript are. Ensure that you use VBScript syntax and not Visual Basic syntax in the scripting environment.

Note: Microsoft's **.NET Framework 2.0 or higher** is a system prerequisite for Scripting Editor, and it must be installed before Authentic Desktop is installed.

The Scripting Editor

The Scripting Editor (*screenshot below*) opens in a separate window and is accessed via the **Tools | Scripting Editor** menu command in the Authentic Desktop GUI. The programming languages that can be used in the Scripting Environment are **JScript** and **VBScript**. The scripting language can be changed by right-clicking the Project item in the Project window, selecting **Scripting Language**, and selecting the language you want.



What you can do with the Scripting Editor

In the Scripting Editor, you can create Forms, Event Handlers, and Macros to build up a

Scripting Project. A Scripting Project can then be set as the Global Scripting Project for Authentic Desktop, thus enabling scripts in the Scripting Project to be used in the application. Additionally, different Scripting Projects can be assigned to different Authentic Desktop projects, thus allowing different scripts to be used for different Authentic Desktop projects.

Every script project can define the .NET runtime version it wants to use. An application can handle multiple scripting projects with different .NET runtime versions simultaneously, but the appropriate .NET version must be installed. For example, script projects with .NET 4.0 will only run on computers having .NET 4.0 installed.

Documentation about the Scripting Editor

The documentation describing the Scripting Environment (this section) is organized into the following parts:

- An overview, which provides a high level description of the Scripting Editor and Scripting Projects.
- A list of steps required to create a Scripting Project.
- An explanation of Global Declarations, together with an example.
- A description of how to create Forms.
- A discussion of Authentic Desktop-specific event handlers.
- An explanation of how to use macros in the Scripting Editor and in Authentic Desktop.

1.1 Overview

The Scripting Editor provides an interface in which you can: (i) graphically design Forms while assigning scripts for components in the Form; (ii) create Event Handlers, and (iii) create Macros.

These Forms, Event Handlers, and Macros are organized into scripting projects, which are then assigned to Authentic Desktop application projects and can be used in the application.

Variables and functions can be defined in a Global Declarations script, which is always executed before Macro or Event Handler scripts.

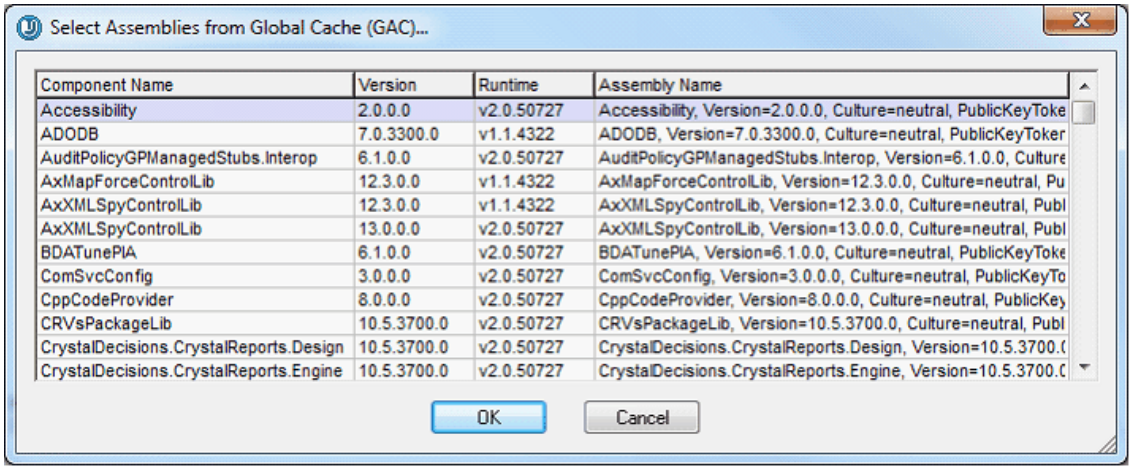
This section gives an overview of the Scripting Editor and Scripting Projects. It is organized into the following sections:

- Scripting Projects in Authentic Desktop, which describes how the scripting projects you create with the Scripting Editor will be used in Authentic Desktop.
- The Scripting Editor GUI, which provides a detailed look at the different parts of the Scripting Editor GUI and how they are to be used.
- Components of a Scripting Project, which explains the different components that go to make up a scripting project.

The details about the creation of the various components (Global Declarations, Forms, Event Handlers, and Macros) are described in their respective sections.

.NET assemblies
Every scripting project can have references to .NET assemblies—in addition to the default references. .NET assemblies can be added for the whole scripting project or for individual macros (by using the new `CLR.LoadAssembly` command in the source code; see Built-in Commands). Assemblies can be added, for example, from the Global Assembly Cache.

To add an assembly, right-click the project or macro, and, from the context menu that pops up, select **Add .NET Assembly | Assembly from Global Cache (GAC)**.



This works in the same way as with Visual Studio and allows access not only to the complete Microsoft .NET Framework but also to any user-defined assembly.

1.1.1 Scripting Projects in Authentic Desktop

All scripts and scripting information created in the Scripting Editor are stored in **Altova Scripting Projects** (**. asprj files**).

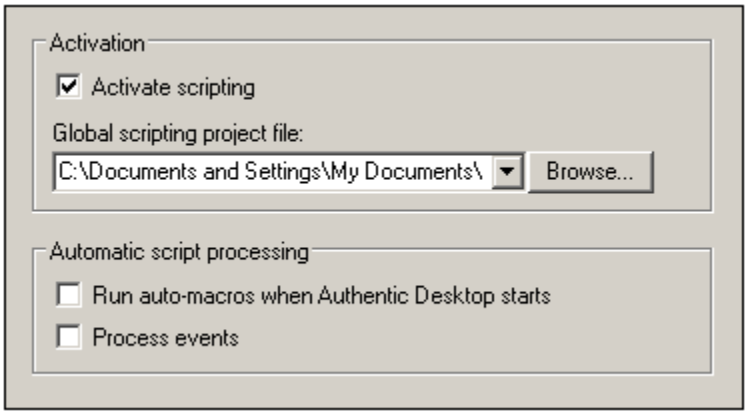
You can create any number of Altova Scripting Projects. After a scripting project has been created, it can be used in the following ways:

- It can be set as the global scripting project for Authentic Desktop. Scripts in the global scripting project can then be called from within the application, and macros of the Global Scripting Project can be used for all Authentic Desktop projects.
- It can be assigned to an Authentic Desktop project (as an application project). When an Authentic Desktop project is open in Authentic Desktop, scripts in the associated scripting project can be called.

Your Authentic Desktop package contains a sample scripting project called `SampleScripts.asprj`. This file is located in the folder: `C:\Documents and Settings\<username>\My Documents\Altova\Authentic Desktop2012\Examples\` and contains global declarations for a few standard tasks.

Setting the global scripting project of an application

The global scripting project of an application is set in the Scripting tab of the Options dialog of Authentic Desktop (*screenshot below*, **Tools | Options**).



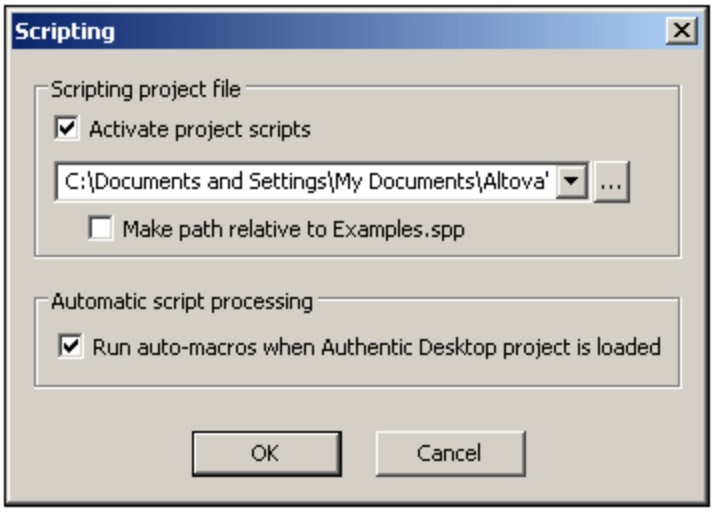
To set a global scripting project for Authentic Desktop, check the *Activate Scripting* check box and then browse for the Altova Scripting Project (`.asprj`) file you want. You can also specify: (i) whether Auto-Macros in the scripting project should be automatically executed when Authentic Desktop starts, and (ii) whether application event handler scripts in the project should be automatically executed or not; check or uncheck the respective check boxes accordingly.

Note: Nested script execution is possible, i.e. Macros can call other macros, and events are received during macro, or event, execution.

Assigning a scripting project to an Authentic Desktop project

A scripting project is assigned to an Authentic Desktop project as follows:

1. In the Authentic Desktop GUI, open the required application project.
2. Select the menu command **Project | Script Settings**. The Scripting dialog (*screenshot below*) opens.



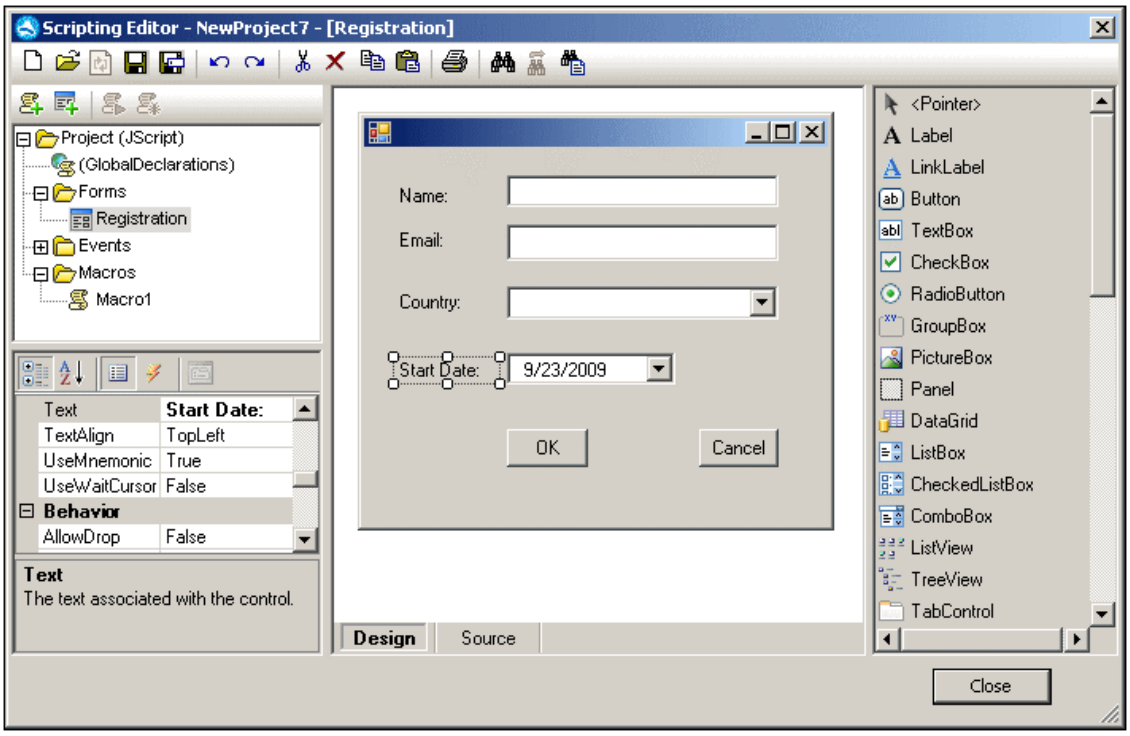
3. Check the *Activate Project Scripts* check box and select the required scripting project (`.asprj` file). If you wish to run Auto-Macros when the Authentic Desktop project is loaded, check the *Run Auto-Macros* check box.
4. Click **OK** to finish.

Note: To deactivate (that is, unassign) the scripting project of an Authentic Desktop project, uncheck the *Activate Project Scripts* check box.

1.1.2 The Scripting Editor GUI

The Scripting Editor GUI is shown below. It has the following parts:

- A toolbar
- A Scripting Project Tree pane (top left-hand side)
- A Properties and Events pane (bottom left)
- A Main Window with Design and Source tabs
- A Form Object Palette (right-hand side)



Scripting Editor toolbar
The Scripting Editor toolbar contains icons for:

- Standard file commands such as **New**, **Open**, **Save**, and **Print**. These commands are used to create new scripting projects, open existing scripting projects, and save and print scripting projects.
- Standard editing commands such as **Copy**, **Paste**, **Undo**, **Redo**, **Find**, and **Replace**. Note that the **Find** and **Replace** commands are applied to code in the Source tab of the Scripting Editor.

Scripting Project Tree
The Scripting Project Tree (*screenshot below*) shows the various components of the scripting project, structured along four main branches: (i) Global Declarations, (ii) Forms, (iii) Events, and (iv) Macros.

The Scripting Project Tree provides access to each component of the scripting project. For example, in order to display and edit a particular Form, expand the Forms folder in the tree (*see screenshot above*), right-click the Form you wish to display or edit, and click **Open** from the context menu that pops up.

A quicker way to open a Form, Event, macro, or the Global Declarations script, is to double-click the respective icon, or text. To delete a Form or Macro from the scripting project, right-click the component and select the **Delete** command from the context menu.

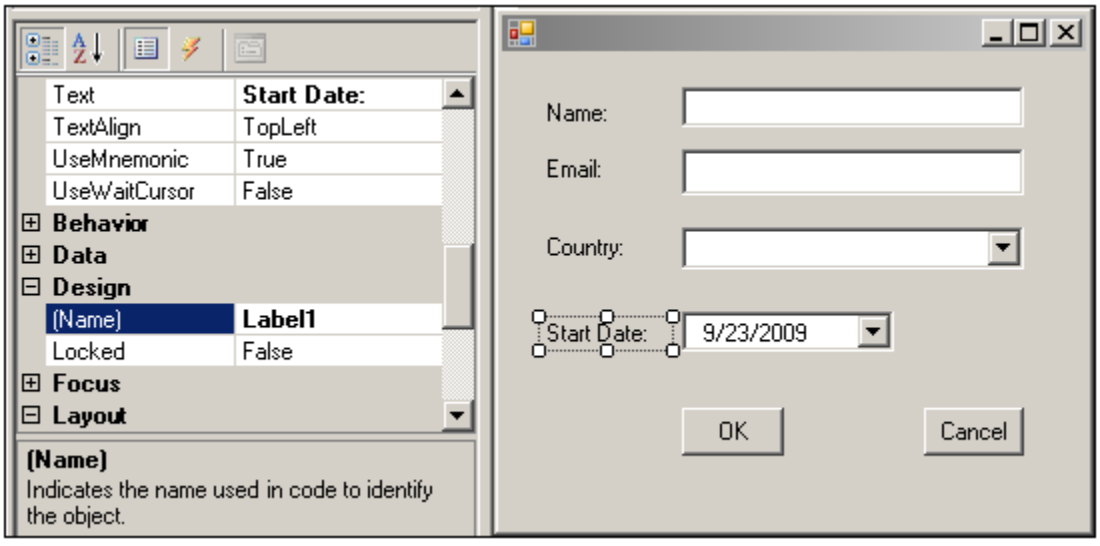
The Scripting Project Tree pane contains a toolbar with icons (*screenshot below*).

The icons, from left to right, are for: (i) creating a new macro, (ii) creating a new form, (iii) running a macro, and (iv) debugging a macro. These commands are also available in the context menu that appears when you right-click any component in the Scripting Project Tree.

Properties and Events

The Properties and Events pane (*screenshot below*) displays the following:

- Form properties, when the Form is selected
- Object properties, when an object in a Form is selected. (The screenshot below shows, at left, the properties of the object selected in the Form at right.)
- Form events, when a Form is selected
- Object events, when an object in a Form is selected



To switch between the properties and events of the selected component, click, respectively, the **Properties** icon (third from left in the Properties and Events toolbar, *see screenshot above*) and the **Events** icon (fourth from left).

The first and second icons from left in the toolbar are, respectively, the **Categorized** and **Alphabetical** icons. These display the properties or events either organized by category or organized in ascending alphabetical order.

When a property or event is selected, a short description of it is displayed at the bottom of the Properties and Events pane.

Main Window
The Main Window displays one component at a time and has one or two tabs depending on what is being displayed. If a Global Declarations script, an Event, or a Macro is being displayed, then a single tab, the Source tab, displays the source code of the selected component.

The Source tab supports:

- syntax coloring
- source code folding
- setting/deleting bookmarks using **CTRL+F2**
- autocompletion entry helper with parameter info
- Goto Brace, Goto Brace Extend
- Zoom In / Zoom Out
- full method/property signature shown next to the autocompletion entry helper
- brace highlighting during code entry

```
if ( x == y.GetName( a, b, c() ) )
```

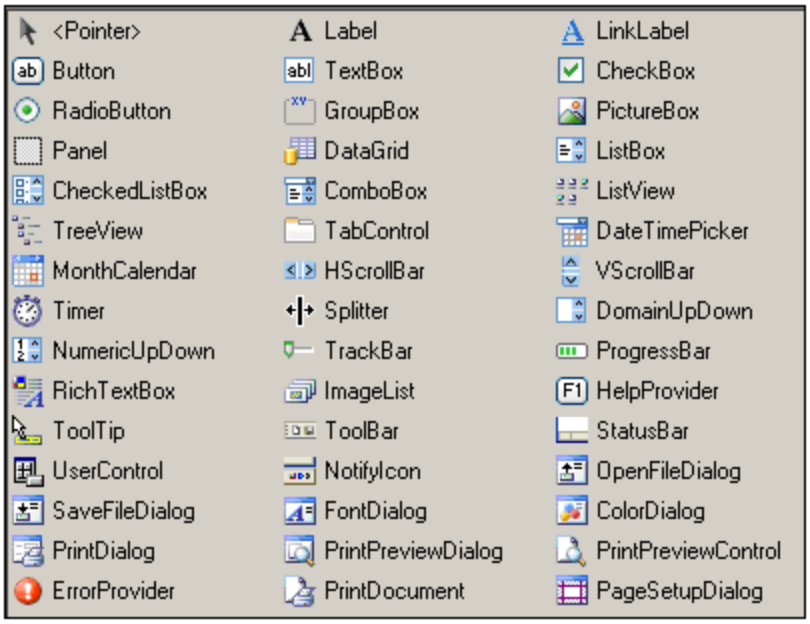
- mouse over popups; placing the mouse over a known method or property, displays its signature (and documentation if available)

If a **Form** is being displayed, then the Main Window has two tabs: a Design tab showing and enabling the layout of the Form, and a Source tab containing the source code for the Form. Content in both the Design tab and Source tab can be edited.

Note: Since JScript and VB Script are untyped languages, entry helpers and auto-completion is supported only in cases of "fully qualified constructs" and "predefined" names.

If names start with `objDocument`, `objProject`, `objXMLData`, or `objAuthenticRange`, members of the corresponding interface will be shown. Auto-completion entry helper and parameter info are shown during editing, but can also be obtained on demand by pressing **Ctrl+Space**.

Form Object Palette
The Form Object Palette contains all the objects that are available for designing Forms and looks something like the screenshot below. Registered ActiveX controls can be added to the Form Object Palette by right-clicking the pane and selecting the **Add ActiveX Control** command



To insert an object from the Form Object Palette click the object you want in the palette, then click at the location in the Form where you wish to insert the object. The object will be placed at this location. In many cases you will need to supply some properties of the object via the Properties and Events pane. You can drag the object to other locations as well as resize it. Further, a number of editing commands, such as centering and stacking objects, can be accessed via the context menu of the selected Form object.

Some Form objects, such as `Timer`, are not added to the Form but are created as Tray Components in a tray at the bottom of the Main Window. You can select the object in the tray and set properties and event handlers for the object via the Properties and Events pane. For an example of how Tray Components are handled, see Form usage and commands.

1.1.3 Components of a Scripting Project

An Altova Scripting Project consists of the following four major components:

- *Global Declarations*, a component which contains definitions of variables and functions that are available to, and can be used by, all Forms, Macros, and Event Handler scripts in the scripting project.
- *Forms*, a component which contains all the Forms defined in the scripting project.
- *Events*, a component which contains Event Handler scripts for all application-based—as opposed to Form-based—events.
- *Macros*, a component which contains all the Macros defined in the scripting project.

These components are displayed in and accessed via the Scripting Project Tree of the Scripting Editor (*screenshot below*).

Given below is a brief description of each of these components.

Global Declarations
The Global Declarations component is a script that contains variables and functions that can be used by Forms, Event Handlers, and Macros. The functions make use of the XMLSpy API to access Authentic Desktop functionality. Creating a variable or function in the Global Declarations module enables it to be accessed from all the Forms, Event Handlers and Macros in the scripting project.

To add a variable or function, open the Global Declarations component (by right-clicking it in the Scripting Project Tree and selecting **Open**) and edit the Global Declarations script in the Main Window. In this script, add the required variable or function.

Forms
In the Scripting Editor, you can build a Form graphically using a palette of Form objects such as text input fields and buttons. For example, you can create a Form to accept the input of an element name and to then remove all occurrences of that element from the active XML document.

For such a Form, a function script can be associated with a text box so as to take an input variable, and an Event Handler can be associated with a button to start execution of the delete functionality, which is available in the XMLSpy API. A Form is invoked by a call to it either within a function (in the Global Declarations script) or directly in a Macro. For details of how to create and edit Forms, see the Forms section.

Event handling
Event Handler scripts can be associated with a variety of available events. You can control events that occur both within Forms (Form events) and within the general application interface (application events). The script associated with an event is executed immediately upon the triggering of that event.

Most events have parameters which provide detailed information about the event. The return value from the script typically instructs the application about how to continue its processing (for example, the application may not allow editing).

An Event Handler runs when the relevant event occurs in the Form or in Authentic Desktop. For details about how to create event handlers, see Event Handlers.

Macros

Macros are used to implement complex or repetitive tasks. Macros do not use either parameters or return values.

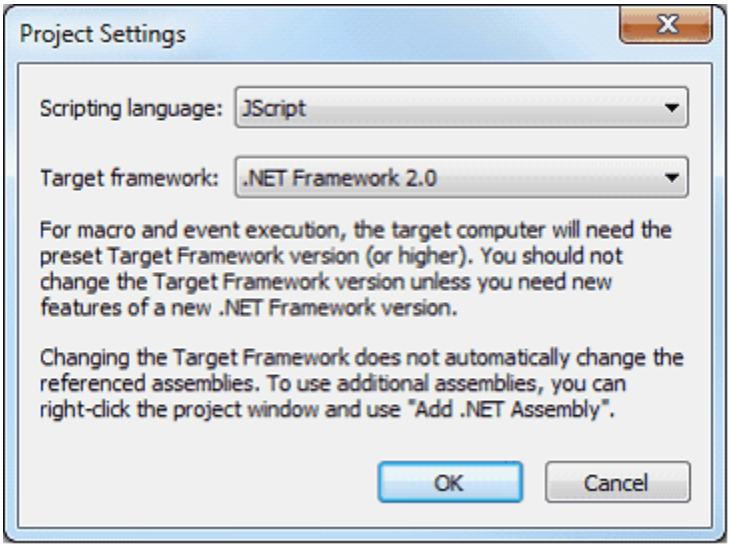
In a Macro, it is possible to access all variables and functions declared in the Global Declarations and to display Forms for user input.

For a simple example of creating a Macro, see Writing a Macro. Also see Running Macros for a description of the ways in which a Macro can be called. A Macro is run from within the Authentic Desktop interface by clicking **Tools | Macros | *[MacroName]***

1.2 Creating a Scripting Project

The broad steps for creating a Scripting Project are as follows:

1. Open the Scripting Editor by clicking the command **Tools | Scripting Editor**.
2. In the Scripting Editor, open a new scripting project by clicking the **New** icon in the Scripting Editor toolbar. The Project Settings dialog (*screenshot below*) pops up.



   Select either JScript or VBScript in the first combo box and the .NET Framework in teh second combo box, and click **OK**. The new Scripting Project is created.
3. Click the **Save** icon in the Scripting Editor toolbar to save the Scripting Project as a `.asprj` file.
4. A Scripting Project can be considered to be made up of several components that work together. These components will typically be a combination of: Global Declarations, Forms, Events, and Macros. They can be created in any order, but you should clearly understand how they work together. The way each type of component is called and executed is described below. How to create each type of component is described in the respective sections about the component type.
5. After you have finished creating all the required components, save the Scripting Project (by clicking the **Save** icon in the Scripting Editor toolbar).
6. Close the Scripting Editor.

Please note:

Right clicking the Project folder and selecting "Scripting Language..." lets you change the scripting language at any time.

How Forms, Event Handlers, and Macros are called and executed

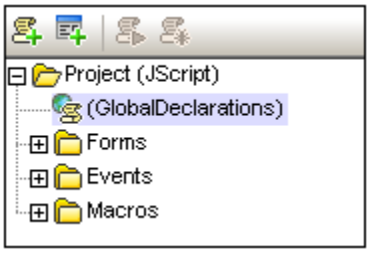
Forms, Event Handlers, and Macros are all created in the Scripting Editor. However, the way they are called and executed is different for each and has a bearing on how you create your scripting projects.

- A Form is invoked by a call to it either within a function in the Global Declarations script or directly in a Macro.
- An Event Handler runs when the relevant event occurs in Authentic Desktop. If an Event Handler for a single event is defined in both the Global Scripting Project and the Authentic Desktop-project-specific Scripting Project, then the event handler for the project-specific Scripting Project is executed first and that for the Global Scripting Project immediately afterwards.

- A Macro is executed from within the Authentic Desktop interface by clicking **Tools | Macros | *[MacroName]***. In a Macro, it is possible to access all variables and functions declared in the Global Declarations and to display Forms for user input.

1.3 Global Declarations

The Global Declarations component is present by default in every Scripting Project (*see screenshot below*), and therefore does not have to be created. In order to add variables and functions to the Global Declarations script of a Scripting Project, you need to open the Global Declarations script and add the code fragment to the Global Declarations script. See Components of a Scripting Project and Creating a Scripting Project for more information.



To open the Global Declarations script of a Scripting Project, right-click the *Global Declarations* item in the Scripting Project Tree (*screenshot above*), and select **Open**. The Global Declarations script opens in the Main Window.

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

Given below is an example function. Remember that creating a variable or function in the Global Declarations script makes this variable or function accessible to all Forms, Event Handlers, and Macros.

Example function
A function called `RemoveAllNamespaces` would have code like this:

```
function RemoveAllNamespaces( objXMLData)
{
       if( objXMLData == null)
    return;

  if( objXMLData. HasChildren)      {
    var objChild;

    // spyXMLDataElement := 4
    objChild = objXMLData. GetFirstChild( 4);

    while( objChild) {
    RemoveAllNamespaces( objChild);

          try{
              var nPos, txtName;
              txtName = objChild. Name;

              if(( nPos = txtName. indexOf( ":")) >= 0)  {
                  objChild. Name = txtName. substring( nPos+1);
              }

    objChild = objXMLData. GetNextChild();
          }
          catch( Err)              {
              objChild = null;
          }
      }
```

```
    }
  }
```

Note:

- It is possible to define local variables and helper functions within macros and event handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application.ShowForm("MsgBox");
}
function On_BeforeStartEditing(objXMLData)
{
  txtLocal = "On_BeforeStartEditing()";
  Helper();
}
```

- Recursive functions are supported.

1.4 Forms

Creating and editing Forms in the Scripting Editor consists of the following steps:

1. Creating a New Form. The new Form is created and named, and has properties defined for it.
2. Designing the Form. A Form is designed by adding Form Objects to it and assigning values for the different Form Objects.
3. Scripting Form Events. Scripts are assigned to Form-related events.

1.4.1 Creating a New Form

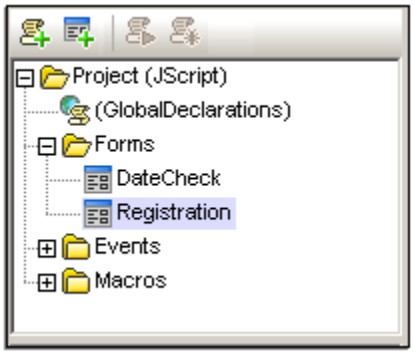
Creating a new Form in the Scripting Editor involves the following steps:

1. Creating a new Form and naming it
2. Specifying the properties of the Form

Creating a new Form and naming it

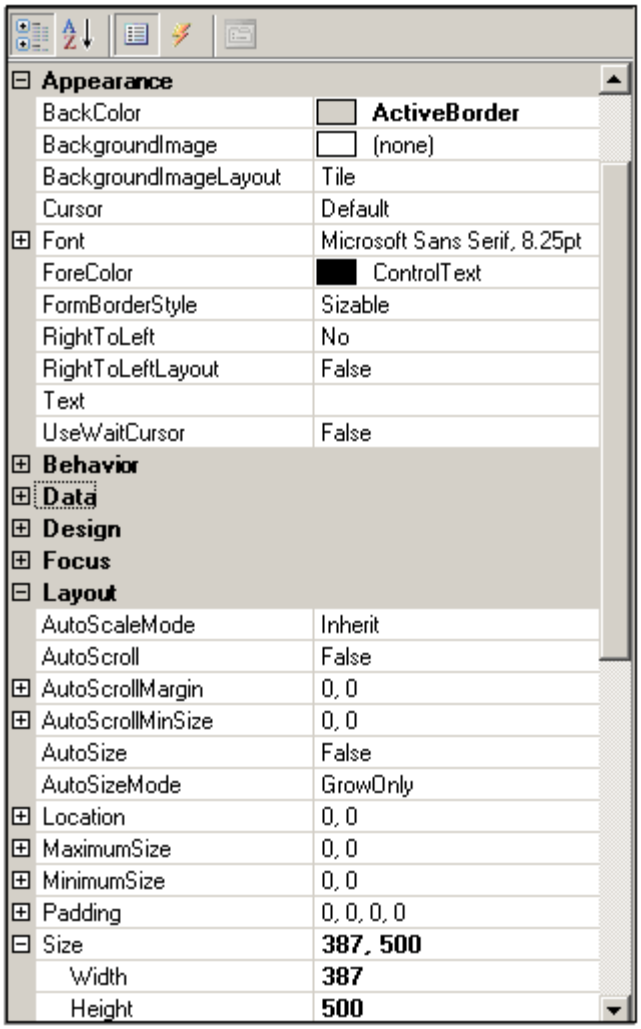
To add a new Form to a scripting project, click the **Add Form** icon (*highlighted in screenshot below*) in the toolbar of the Project Overview pane. Enter the name of the new Form.

A new Form is added to the project. It appears in the Main Window and an entry for it is created in the Scripting Project Tree pane, under the Forms heading. Press the F2 function key to rename the form, or right click the form name and select Rename from the context menu. In the screenshot below, we have named the new Form *Registration*.



Form properties

The properties of the Form, such as its size, background color, and font properties, can be set in the Properties pane. The screenshot below shows the size and background-color property values in bold, in the *Layout* and *Appearance* categories, respectively.



Testing a Form

You can test a form in the Scripting Editor by right-clicking it in the Project Overview pane and selecting the **Test Form** Command.

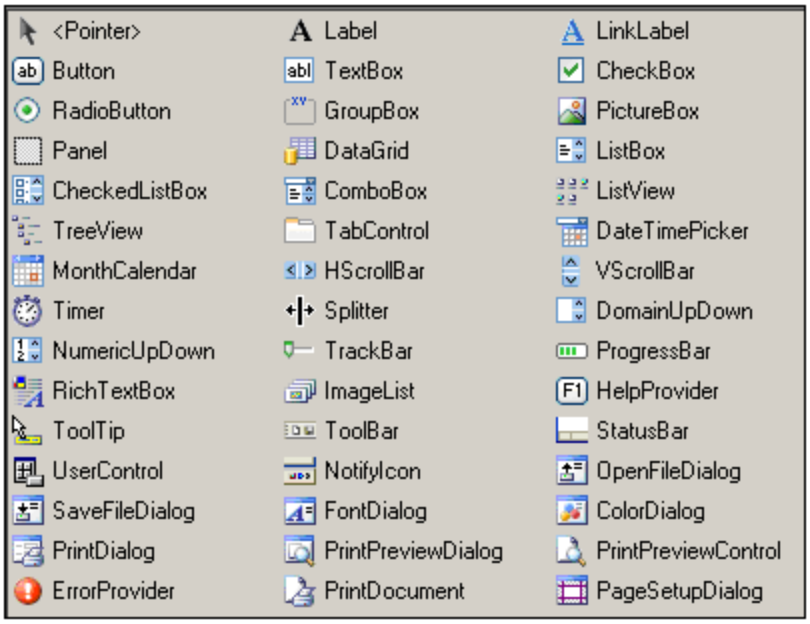
1.4.2 Form Design and Form Objects

Designing a Form consists of the following steps:

- Placing an object from the Form Object Palette in the Form design.
- Assigning values for the properties of individual Form Objects.
- Assigning scripts for Form-based events.

The Form Object Palette

The Form Object Palette contains all the objects that are available for designing Forms and looks something like the screenshot below. Registered ActiveX controls can be added to the Form Object Palette by right-clicking the pane and selecting the **Add ActiveX Control** command



To insert an object from the Form Object Palette click the object you want in the palette, then click at the location in the Form where you wish to insert the object. The object will be placed at this location. In many cases you will need to supply some properties of the object via the Properties and Events pane. You can drag the object to other locations as well as resize it. Further, a number of editing commands, such as centering and stacking objects, can be accessed via the context menu of the selected Form object.

Some Form objects, such as `Timer`, are not added to the Form but are created as Tray Components in a tray at the bottom of the Main Window. You can select the object in the tray and set properties and event handlers for the object via the Properties and Events pane. For an example of how Tray Components are handled, see Form usage and commands.

Some of the most commonly used objects are described below:

Label: Adds text fields such as captions or field descriptions.

Button: Adds a button. It is possible to assign bitmaps as background images for these buttons.

Check Box: Adds a check box, which enables `Yes/No` type selections.

Combo Box: Adds a combo box, which allows the user to select an option from a drop-down menu.

List Box: Adds a list box, which displays a list of items for selection.

TextBox: Enables the user to enter a single line of text.

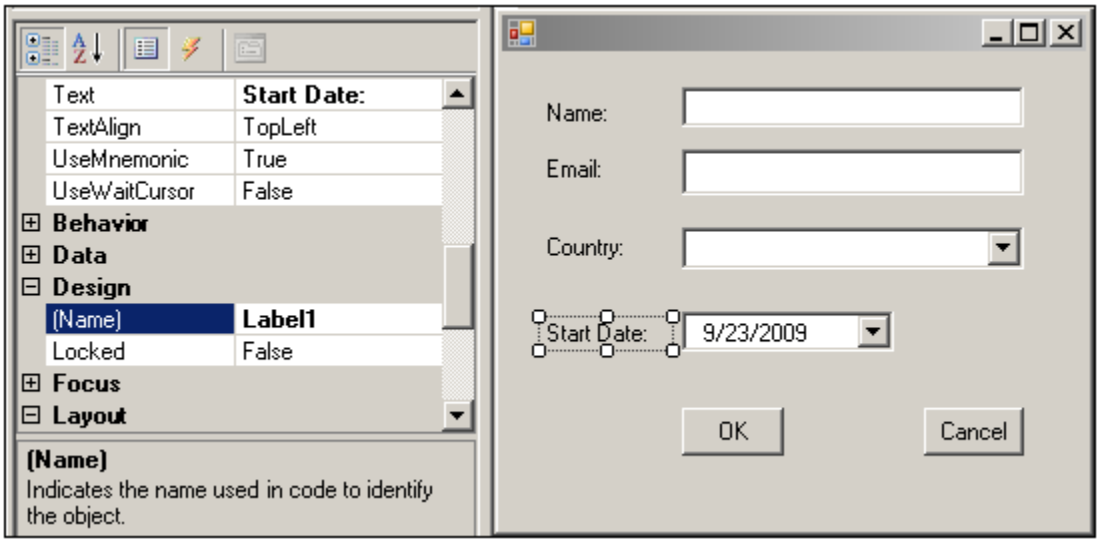
Rich TextBox: Enables the user to enter multiple lines of text.

Creating objects and setting their properties
To create an object in the Form, first select the required object in the Form Object Palette and then click the location in the Form where you want to insert it. After the object has been inserted, you can resize it as well as drag it to another location in the Form.

When an object is selected in the design, you can specify its properties in the Properties and Events pane. In the toolbar of the Properties and Events pane, click the Properties icon to display a list of the object's properties.

For example, in the screenshot below, the Label object with the text *Start Date* has been selected in the design. In the Properties and Events pane, the name of the object (which is the name that is to be used to identify the object in code, `Label1` in the screenshot below) is given in the *Design* category of properties; in this case, the name of the object is `Label1`.

The text of the label (which is what appears in the Form) must be entered as the value of the *Text* property in the *Appearance* category of properties.

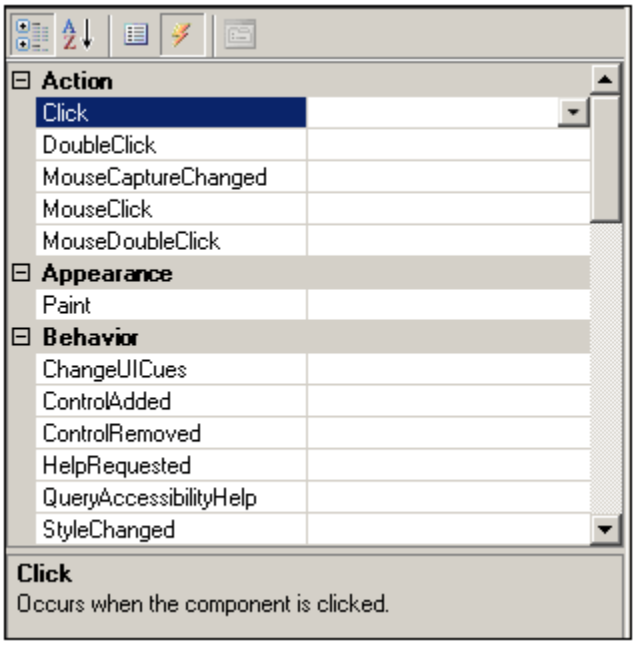


To assign other object properties, enter values for them in the Properties and Events pane.

Testing a Form
You can test a form in the Scripting Editor by right-clicking it in the Project Overview pane and selecting the **Test Form** Command.

1.4.3 Form Events

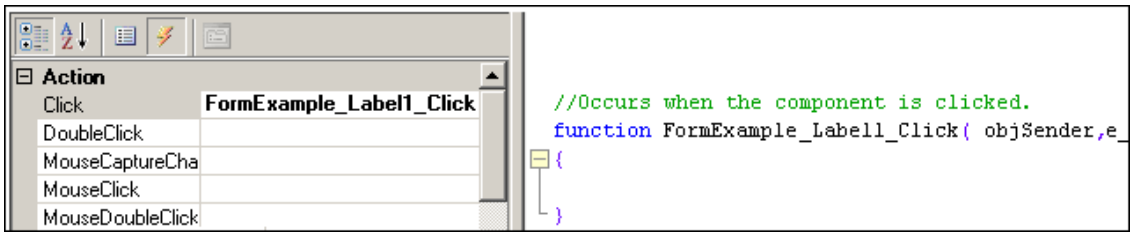
When an object is selected in the design, clicking on the Events icon in the toolbar of the Properties and Events pane (*fourth icon from left*), displays all the events available for that object (*see screenshot below*). These can be displayed either by category (*screenshot below*) or alphabetically.



For each event, you can enter the name of an existing event handler or function. Alternatively:

- you can double click on an event to create: (i) an empty function script in the *Source* tab of the Main Window, and (ii) an association of the newly created function with the selected event.
- double click a button in the design tab, to directly generate the handler stub in the code window.

The screenshot below was taken after the *Click* event was double-clicked. Notice that an empty event handler function called `FormExample_Label1_Click` has been created in the Main Window and that, in the Properties and Events pane, this function has been associated with the *Click* event.



Enter the required scripting code and save the project.

Writing the required scripts

After the visual design of the form is complete, form objects will typically be associated with suitable scripts. The example below is a script that adds colors when a button is clicked. The script is inserted as an event handler for the `Click` event of the button `Button1` (the event is

available in the Properties and Events pane when the button is selected in the design):

```
function FormExample_Button1_Click( objSender, e_EventArgs )
{
	// Sets the ForeColor (red) of the button.
	objSender.ForeColor = CLR.Static( "System.Drawing.Color" ).Red;
	// Sets the BackColor (blue) of the button.
	objSender.BackColor = CLR.Static( "System.Drawing.Color" ).Blue;
	// Sets the form BackColor (green).
	objSender.FindForm().BackColor = CLR.Static( "System.Drawing.Color"
).Green;
}
```

1.5 Events

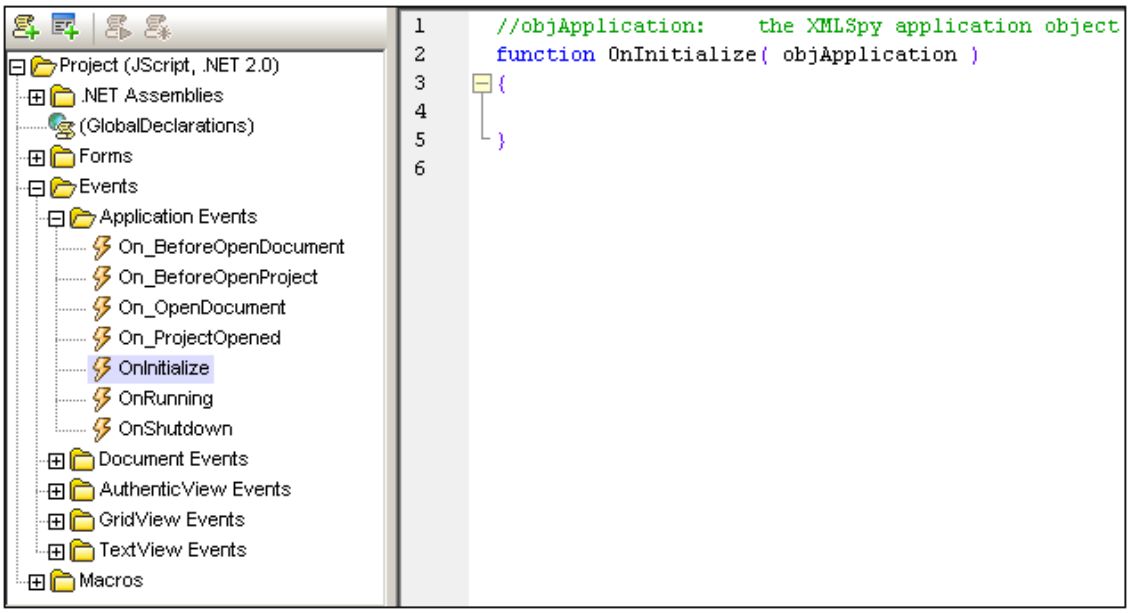
The Events folder of the scripting project (*see screenshot below*) contains folders for the following type of events:

- Application Events
- Document Events
- Authentic View Events
- Grid View Events
- Text View Events

Note that these events are Authentic Desktop-specific, as opposed to Form-based events. Each of the folders listed above contains a set of events for which Event Handler scripts can be written.

Application Events, for example, are shown in the screenshot below.

To access the event handler script of any of these events, right-click the event and select **Open** from the context menu. The script will be displayed in the Main Window (*see screenshot below*) and can be edited there. After you have finished editing the script, save changes by clicking the **Save** command in the toolbar of the Scripting Editor.



Note the following points:

- Event Handlers need function headers with the correct spelling of the event name. Otherwise the Event Handler will not be called.
- It is possible to define local variables and helper functions within Macros and Event Handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application.ShowForm("MsgBox");
}
```

```
function On_BeforeStartEditing(objXMLData)
{
  txtLocal = "On_BeforeStartEditing()";
  Helper();
}
```

- In order for events to be processed, the Process Events options must be toggled on in the Scriptings options of Authentic Desktop. See Scripting Projects in Authentic Desktop for details.
- Also see Programming Points.

Application Events

OnInitialize
The `OnInitialize` event is raised after the main window becomes visible but before any project is loaded. This event is not raised if the application can't be loaded at all.

OnRunning
If the application is completely loaded and after the `OnInitialize` event occurs, the `OnRunning` event is raised.

OnShutdown
The event is raised after any open project and all documents have been closed on shutdown of the application. The main window is no longer visible.

Example

The following script is an Event Handler for the `On_BeforeOpenProject` event. It allows you to add a script that will be executed each time before Authentic Desktop opens a project. The example script below sequentially opens all XML files located in the XML folder of the project and validates them. If the validation fails, the script shows the validation error and stops. If a file passes the validity test, it will be closed and the next file will be opened.

Enter the following script for the `On_BeforeOpenProject()` event, and then save the scripting project.

```
function On_BeforeOpenProject()
{
 var bOK;
 var nIndex, nCount;
 var objItems, objXMLFolder = null;

 objItems = Application.CurrentProject.RootItems;
 nCount = objItems.Count;

 // search for XML folder
 for(nIndex = 1; nIndex <= nCount; nIndex++) {
  var txtExtensions;
  txtExtensions = objItems.Item(nIndex).FileExtensions;

 if(txtExtensions.indexOf("xml") >= 0)     {
  objXMLFolder = objItems.Item(nIndex);
 break;
 }
}

// does XML folder exist?
if(objXMLFolder) {
 var objChild, objDoc;

 nCount = objXMLFolder.ChildItems.Count;
```

```
// step through associated xml files
for( nIndex = 1; nIndex <= nCount; nIndex++) {
 objChild = objXMLFolder.ChildItems.Item( nIndex);

 try{
  objDoc = objChild.Open();

  // use JScript method to access out-parameters
  var strError = new Array(1);
  var nErrorPos = new Array(1);
  var objBadData = new Array(1);

  bOK = objDoc.IsValid( strError, nErrorPos, objBadData);

  if(!bOK) {
   // if the validation fails, we should display the
   // message from XMLSpy
   // of course we have to create the form "MsgBox" and
   // define the global txtMessage variable
   //
   // txtMessage = Position:" + nErrorPos[0] + "\n" + // strError[0];
   // txtMessage += "\n\nXML:\n" + objBadData[0].Name + ", " +
   //    objBadData[0].TextValue;
   //
   // Application.ShowForm("MsgBox");

    break;
  }

  objDoc.Close( true);
  objDoc = null;
 }
 catch( Err) {
  // displaying the error description here is a good idea

  // txtMessage = Err.Description;
  // Application.ShowForm("MsgBox");

  break;
  }
 }
}
}
```

Testing the Event Handler

Switch to Authentic Desktop, and open a project to see how the `BeforeOpenProject` event is handled.

1.6 Macros

Macros automate repetitive or complex tasks. In the Scripting Environment, you can create a script that calls application functions as well as custom functions that you have defined. This flexibility provides you with a powerful method of automating tasks within Authentic Desktop. This section about macros is organized as follows:

- Creating and Editing a Macro describes how to create a new macro and edit an existing one.
- Running a Macro explains how a macro can be run from the Scripting Editor and from the broader Authentic Desktop environment as well.
- Debugging describes how macros can be debugged.

Key points about macros
Given below is a summary of important points about macros.

- Any number of macros can be added to the active scripting project. These macros are saved in the Altova Scripting Project file (. `asprj` file).
- Functions that are used in a macro can be saved as a Global Declaration. All Global Declarations are also saved in the Altova scripting project file (. `asprj` file).
- The macro can be tested by running it from within the Scripting Editor, and it can be debugged from within the Scripting Editor.
- Authentic Desktop can have one global Scripting Project, and a second scripting project, assigned to the currently loaded project, active at any one time; the macros are available to both of them. See Running a Macro for details.

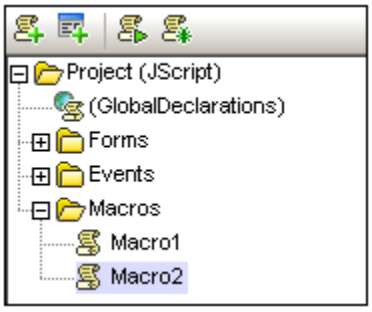
1.6.1 Creating and Editing a Macro

The following operations enable you to create a new macro and edit an existing macro.

Creating a new macro

Right-click the Macro folder in the Scripting Projects tree and select **Add Macro** from the context menu. (The **Add Macro** command can also be selected from the context menu of any item in the Scripting Projects tree.) Alternatively, click the **New Macro** icon in the toolbar of the Scripting Projects tree.

The newly created (and empty) macro document is displayed in the Main Window, and the name of the macro is displayed in the title bar of the Scripting Editor (*screenshot below*).



Naming or renaming a macro

To name or rename a macro, click the macro name in the Scripting Project tree and press the **F2** function key, or right click the name and select **Rename** from the context menu.

Opening a macro

To open a macro, right-click the macro in the Macros folder of the Scripting Project tree (*see screenshot above*), and select the **Open** command. The macro is displayed in the Main Window and its name is displayed in the title bar of the Scripting Editor (*screenshot below*). Alternatively, double-clicking a macro in the Scripting Project tree opens it in the Main Window.

Editing the macro

To edit a macro, enter or edit its code in the Main Window. For example, the following code creates the Form named `Form1` in memory and then shows it. `Form1` must already have been created (using the Scripting Editor's Form creation) before this macros is run.

```
objForm = CreateForm( 'Form1' );
objForm.ShowDialog();
```

The following macro uses the `RemoveAllNamespaces` function to remove all namespaces in the active XML document.

```
if( Application.ActiveDocument != null)  {

RemoveAllNamespaces( Application.ActiveDocument.RootElement);
              Application.ActiveDocument.UpdateViews();
}
```

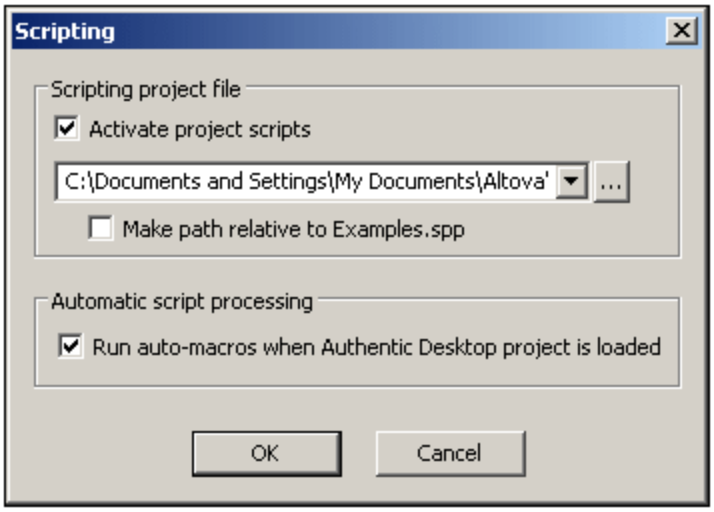
The `RemoveAllNamespaces` function itself will have to be defined in the Global Declarations script. After the `RemoveAllNamespaces` function has been defined, the macro is complete and

can be run.

Note: Macros do not support parameters or return values.

Setting a macro as an Auto-Macro
When a macro is set as an Auto-Macro it can be run automatically when: (i) Authentic Desktop is started, or (ii) an Altova Authentic Desktop project is loaded in Authentic Desktop. To specify whether Auto-Macros should be run in each of these two events, check the *Run Auto-Macros* option in the Automatic Script Processing pane of the relevant dialogs:

- *When Authentic Desktop is started:* the Scripting tab of the Authentic Desktop Options dialog (**Tools | Options** menu command).
- *When an Authentic Desktop project is loaded into Authentic Desktop:* the Scripting dialog (*screenshot below*, **Project | Scripting Settings** menu command).



To set a macro as an Auto-Macro, right-click the macro in the Scripting Project tree and select the command **Set as Auto-Macro**. This is a toggle command; so to remove the Auto-Macro setting of a macro, select the command again.

1.6.2 Running a Macro

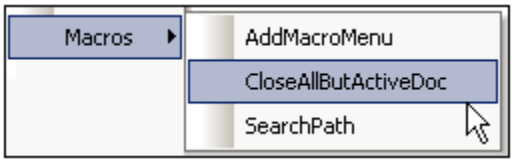
To run a macro in the Scripting Editor, right-click the macro in the Scripting Project tree and select the command **Run Macro**.

There are different ways to run a macro from Authentic Desktop:

- Via the Tools | Macros menu of Authentic Desktop.
- By creating and using a toolbar button for a macro.
- By creating and using a menu item for a macro.

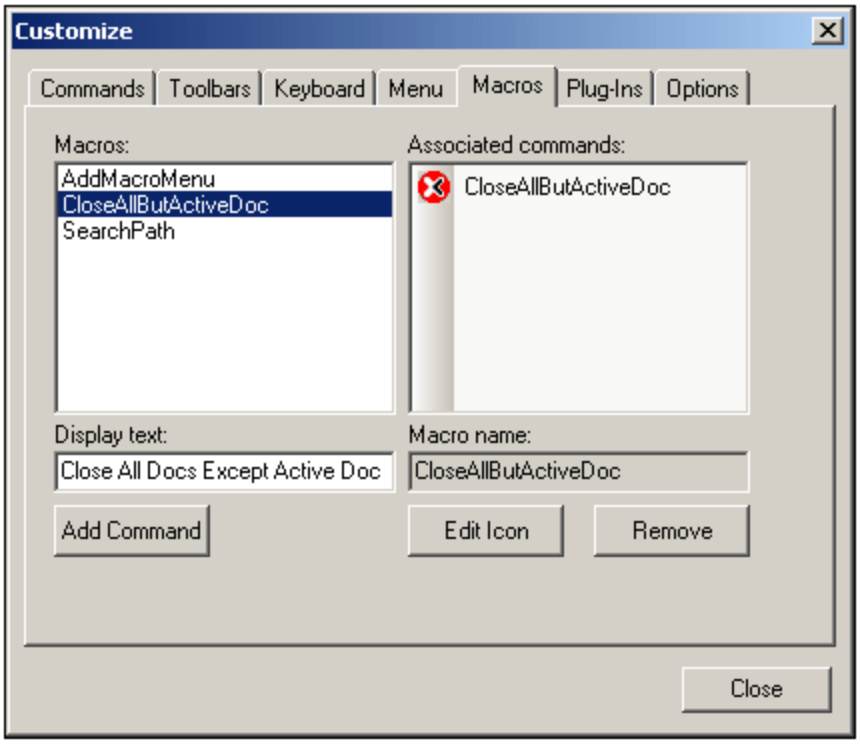
Note that only one macro can be run at a time. After a macro (or event) is executed, the script is closed and global variables lose their values.

The Authentic Desktop command to run Macros
The **Tools | Macros** menu command (*screenshot below*) opens a submenu containing the macros defined in the Scripting Project that is currently active in Authentic Desktop. The active Scripting Projects are specified in the Scripting tab of the Options dialog, or in the Scripting tab of the project settings.



From the submenu of available macros, select the macro to run. The macro will be executed.

Toolbar icon
You can create an icon in the toolbar or a menu item that runs a selected macro. To do this, click **Tools | Customize | Macros**. This causes the Customize dialog to be displayed (*screenshot below*).



Now do the following:

1. In the Macros tab of the Customize dialog, select the required macro from the Macros pane. The macros in the Macros pane are those in the active Scripting Project (which is specified in the Scripting tab of the Options dialog).
2. In the *Display Text* input field enter the name of the icon. This name will appear when the cursor is placed over the icon when it is in the toolbar.
3. Click **Add Command** to add it to the list of commands.
4. Select the command and click **Edit Icon** to create a new icon.
5. Drag the finished icon from the *Associated Commands* pane and drop it on to the toolbar or menu when the cursor changes from an arrow to an I-beam or line.
6. Macros can even be assigned their own shortcuts in the Keyboard tab of the Customize dialog (*see screenshot above*).

To remove the toolbar icon, open the Macros tab of the Customize dialog and drag the icon out of the toolbar and into the *Associated Commands* pane. Select the command in the *Associated Commands* pane and click **Remove** to remove the command from the pane.

Item in the Tools menu

The XMLSpy API includes a function, `AddMacroMenuItem()`, to add macros as menu items to the **Tools** menu. This function can be used to add one or more macros to the **Tools | Macros** list of macros. Typically, you should do this as follows:

1. Add the macro menu item by calling the XMLSpy API function, `AddMacroMenuItem()`.

```
Application.AddMacroMenuItem("DeleteElements","Delete Elements
Dialog");
```

   - The function's first parameter (`DeleteElements` in the example listing above) is the name of the macro. If you run the macro and there is an open project having scripts associated with it, Authentic Desktop searches for the macro in the project scripts

first.
If there are no project scripts, or if Authentic Desktop cannot find the macro, then it looks for the macro in the global scripts.

- The second parameter (`Delete Elements Dialog`) is the display text for the menu item.

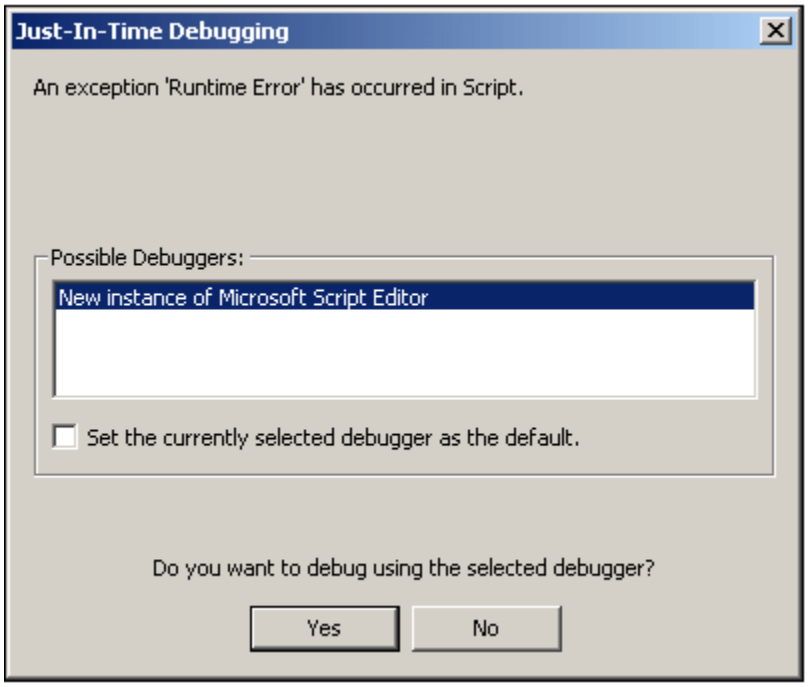
2. Reset the **Tools** menu by calling `ClearMacroMenu()`. This removes all previously added menu items

The best way to call these two functions is with the `Autorun` macro of the global scripting project or the `On_OpenProject` event.

1.6.3 Debugging a Macro

You can debug a macro using an installed debugger. To do this, right-click the macro in the Scripting Project tree and select the command **Debug Macro**.

This pops up the Just-In-Time Debugging dialog (*screenshot below*), which lists the debuggers available on the machine. Select the debugger you wish to use and click **Yes**.



The selected debugger starts.

1.7 Programming Points

The following programming points should be noted:

- All namespaces and types of the following .NET assemblies can be accessed in the Microsoft .NET Framework per default:

```
System
System.Data
System.Design
System.Drawing
System.Windows.Forms
System.XML
```

  Additional assemblies can be added to the scripting project via the project's context menu, or dynamically (at runtime) in the source code by using `CLR.LoadAssembly`.

- Out-parameters from methods of the XMLSpy API require special variables in JScript. Given below are some examples.

```
// use JScript method to access out-parameters
var strError = new Array(1);
var nErrorPos = new Array(1);
var objBadData = new Array(1);
bOK = objDoc.IsValid(strError,nErrorPos,objBadData);END
```

- Out-parameters from methods of the.NET Framework require special variables in JScript.
  For example:

```
    var dictionary = CLR.Create( "System.Collections.Generic.Dictionary<
System.String,
    System.String >" );
    dictionary.Add("1", "A");
    dictionary.Add("2", "B");

    // use JScript method to access out-parameters
    var strOut = new Array(1);
    if ( dictionary.TryGetValue("1", strOut) ) // TryGetValue will set
the out parameter
        alert( strOut[0] ); // use out parameter
```

- .NET Methods that require integer arguments should not be called directly with JScript Number Objects which are Floating Point Values.

  For example, instead of:

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

use:

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

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

  For example:

```
// iterate using the JScript iterator
var itr = new Enumerator( coll );
for ( ; !itr.atEnd(); itr.moveNext() )
    alert( itr.item() );

// iterate using the .NET iterator
var itrNET = coll.GetEnumerator();
while( itrNET.MoveNext() )
    alert( itrNET.Current );
```

- .NET templates can be instantiated as shown below:

```
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 Enum values are accessed as shown below:

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

- Enumeration literals, as defined in the Altova type libraries, can now be used instead of numerical values.

```
    objExportXMIFileDlg.XMIType = eXMI21ForUML23;
```

1.7.1 Built-in Commands

This section lists:

- Built-in commands
  alert
  conform
  doevents
  CreateForm
  lastform
  prompt
  ShowForm
  watchdog

- .NET interoperability commands
  CLR.Create
  CLR.Import
  CLR.LoadAssembly
  CLR.ShowImports
  CLR.ShowLoadedAssemblies
  CLR.Static

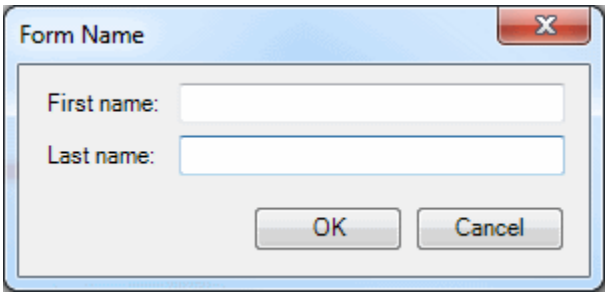
Built-in commands
The following built-in commands are available.

ShowForm(strFormName : String)
Instantiates a New Form object from the given form name and immediately shows it as Dialog.
Return Value: A Number that represents the generated DialogResult (`System. Windows. Forms. DialogResult`).

Example:

```
var dialogResult = ShowForm( "FormName" );
```

Shows Form "FormName" as Dialog:



The DialogResult can be evaluated e.g. by:

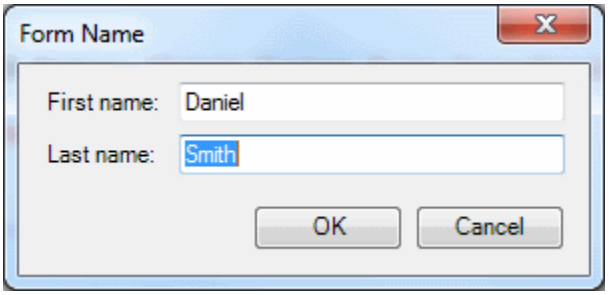
```
if ( dialogResult == CLR. Static( "System. Windows. Forms. DialogResult" ).
OK )
      alert( "ok" );
else
      alert( "cancel" );
```

CreateForm(strFormName : String)
Instantiates a New Form object from the given Form name.
Return Value: The Form object (`System. Windows. Forms. Form`) of the given name, or `null`

if no Form with such name exists.

```
Example:
        var myForm = CreateForm( "FormName" );
        if ( myForm != null )
        {
                myForm.textboxFirstName.Text = "Daniel";
                myForm.textboxLastName.Text = "Smith";
                var dialogResult = myForm.ShowDialog();
        }
```

Shows Form "FormName" as Dialog - TextBoxes are initialized:



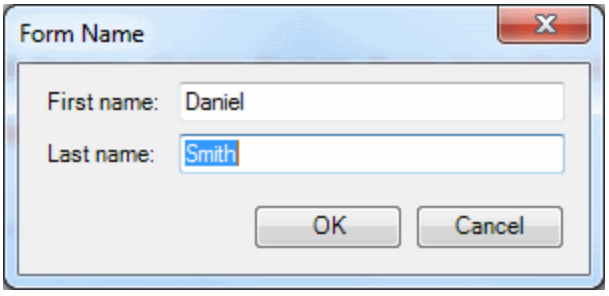
The DialogResult can be evaluated e.g. by:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).
OK )
        alert( "ok" );
else
        alert( "cancel" );
```

lastform
This global field can be used to conveniently access the last form object that was created. *Return Value:* Returns a reference to the last form object (`System.Windows.Forms.Form`) that was successfully instantiated via `CreateForm()` or `ShowForm()`.

```
Example:
        CreateForm( "FormName" );
        if ( lastform != null )
        {
                lastform.textboxFirstName.Text = "Daniel";
                lastform.textboxLastName.Text = "Smith";
                var dialogResult = lastform.ShowDialog();
        }
```

Shows Form "FormName" as Dialog - TextBoxes are initialized (similar to the CreateForm example above):



doevents()
Processes all Windows messages currently in the message queue.
Return Value: None

```
Example:
	for ( i=0; i < nLongLastingProcess; ++i )
	{
		// do long lasting process

		doevents(); // process windows messages; give UI a chance to
	update
	}
```

watchdog(bEnable : boolean)
Long running CPU-intensive scripts cause the watchdog to ask the user if the script should be terminated. The `watchdog()` method is used to disable or enable this behavior.
Per default the watchdog is enabled.
Return Value: None

```
Example:
	watchdog( false );  // disable watchdog - we know the next statement is
	CPU intensive but it will terminate for sure
	doCPUIntensiveScript();
	watchdog( true );   // re-enable watchdog
```

Usage tip:
Calling watchdog(true) can also be used to reset the watchdog. This can be useful before executing long running (CPU intensive) tasks to ensure they have the maximum allowed script processing quota.

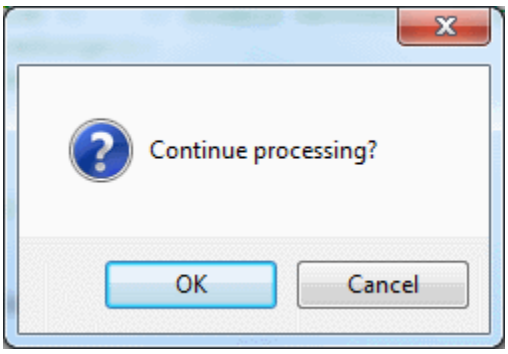
alert(strMessage : String) or MsgBox(strMessage : String)
An alert box is used to show a given message. The user will have to click "OK" to proceed.
Return Value: None

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

confirm(strMessage : String)
Opens a dialog that shows the given confirm message.
A confirm box is often used to verify or accept something. The user will have to click either "OK" or "Cancel" to proceed.
Return Value: A Boolean that represents the users answer. If the user clicks "OK", the dialog returns true, if the user clicks "Cancel", the dialog returns false.

Example:

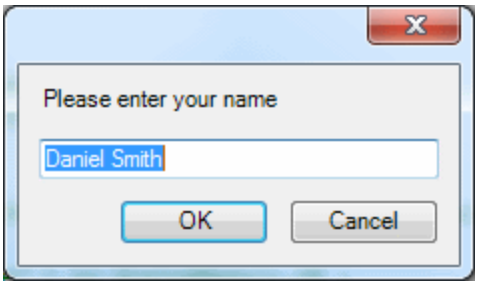
```
if ( confirm( "Continue processing?" ) == false )
    return;
```



prompt(strMessage : String, strDefault : String)
Opens a dialog that shows the given prompt message and a TextBox control with a default answer.
A prompt box is often used to input a simple string value.
Return Value: A String that contains the TextBox value or null if the user selected "Cancel".

Example:

```
var name = prompt( "Please enter your name", "Daniel Smith" );
if ( name != null )
    alert( "Hello " + name + "!" );
```



.NET interoperability commands

To allow further interoperability with the .NET Framework additional functions are provided under CLR.

CLR. Import(strNamespaceCLR : String)
This is the scripting equivalent to the C# *using* / VB.Net *imports* keyword. This allows to leave out the given namespaces in successive calls like `CLR. Create()` and `CLR. Static()`.
Return Value: None

Example:

Instead of always having to use full qualified names:

```
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 and use the short form:

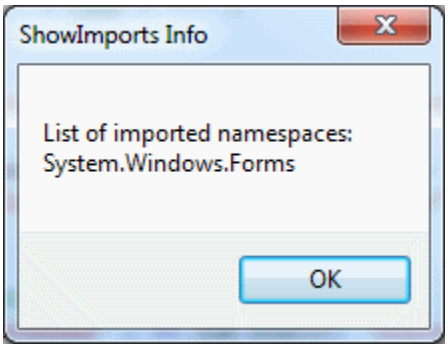
```
CLR.Import( "System.Windows.Forms" );
if ( ShowForm( "FormName" ) == CLR.Static( "DialogResult" ).OK )
{
	var sName = lastform.textboxFirstName.Text + " " + lastform.
textboxLastName.Text;
	CLR.Static( "MessageBox" ).Show( "Hello " + sName );
}
```

Please note:
Importing a namespace does not add or load the corresponding assembly to the scripting project!
Assemblies can be added to the scripting project by..... or dynamically (at runtime) in the source code by using CLR.LoadAssembly.

CLR.ShowImports()
Opens a MessageBox dialog that shows the currently imported namespaces. The user will have to click "OK" to proceed.
Return Value: None

Example:

```
CLR.ShowImports();
```



CLR.LoadAssembly(strAssemblyNameCLR : String)
Loads the .NET assembly with the given long assembly name or file path.
Return Value: A Boolean value. True if the assembly could be loaded, false otherwise.

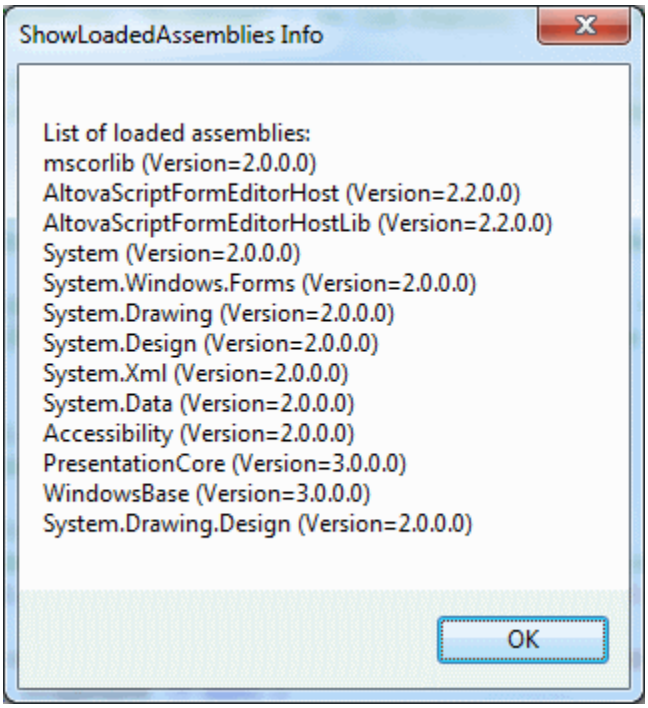
Example:

```
// set clipboard text (if possible)
// System.Windows.Clipboard is part of the PresentationCore assembly,
so load this assembly first:
if ( CLR.LoadAssembly( "PresentationCore, Version=3.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35", true ) )
{
	var clipboard = CLR.Static( "System.Windows.Clipboard" );
	if ( clipboard != null )
		clipboard.SetText( "HelloClipboard" );
}
```

CLR.ShowLoadedAssemblies()
Opens a MessageBox dialog that shows the currently loaded assemblies. The user will have to click "OK" to proceed.
Return Value: None

Example:

```
CLR.ShowLoadedAssemblies();
```



CLR.Create(strTypeNameCLR : String, constructor arguments ...)
Creates a new .NET object instance for the given typename. If more than one argument is passed the successive arguments are interpreted as the arguments for the constructor of the .NET object.
Return Value: A reference to the created .NET object

Examples:

```
var objArray = CLR.Create("System.Collections.ArrayList");

var newItem = CLR.Create( "System.Windows.Forms.ListViewItem",
"NewItemText" );

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

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

CLR.Static(strTypeNameCLR : String)
Gives access to .NET types that have no instances and contain only static members.
Return Value: A reference to the static .NET object

Examples:

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

var clipboard = CLR.Static( "System.Windows.Clipboard" );
clipboard.SetText( "HelloClipboard" );
```

```
if ( ShowForm( "FormName" ) == CLR.Static(
"System.Windows.Forms.DialogResult" ).OK )
       alert( "ok" );
else
       alert( "cancel" );
```

Form usage and commands

Form usage is as follows:

With Form objects, the Form Component Tree can be accessed naturally via field access:

For example, suppose there is a Form designed as follows:

```
MyForm
   ButtonPanel
      OkButton
      CancelButton
   TextEditor
   AxMediaPlayer1

TrayComponents:
   MyTimer
```

The Form can then be instantiated from script as:

```
var objForm = CreateForm("MyForm");
```

To access one its components the field access can be used:

```
objForm.ButtonPanel.OkButton.Enabled = false;
```

or

```
objForm.TextEditor.Text = "Hello World";
```

To access Tray Components use the following method on the Form object:

```
var objTrayComponent = <A form object>.GetTrayComponent(strComponentName
: String);
```

In our example to get a reference to the Timer Component to enable it use the following:

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

For ActiveX Controls the underlying COM object can be accessed 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";
```

1.8 Migrating to Scripting Editor 2010 and Later

The Scripting Editor in Authentic Desktop from version 2010 onwards uses a different underlying technology than earlier versions used. Consequently, scripting projects that were created with versions of Authentic Desktop prior to version 2010 might need to be modified. The following points need to be noted.

- If a previous Scripting Projects (`.prj` file) is opened with the new Scripting Editor (version 2010 and later), the visual layout of Forms will be migrated as faithfully as possible and scripts will be copied as they are in the `.prj` file. You will then need to modify the scripts to be in accordance with the new technology used by the Scripting Editor, and which is described in this documentation.
- `TheView` object: The old Scripting Environment provided an artificial property named `TheView` that was only accessible from inside event handlers. It was used to access the Form that trigged the event (either directly or from one of its child controls). The new Scripting IDE does **not** provide this artificial property but instead provides the same functionality, and much more, with orthogonal built-in scripting helper functions combined with the power of the .NET framework.
- Since all event handlers in the new Scripting Environment get a sender object as a first parameter, the source that triggered the event is always available. By calling the .NET function `FindForm()` on the sender object one can access the Form object easily. Alternatively (if only one Form is involved) the built-in property `lastform` can be used. Note that the use of `lastform` is not constrained to event handlers (as was the case with `TheView`). It can be used everywhere in script code.

Given below is a list of methods and properties of the `TheView` object, each accompanied by an alternative mechanism offered by the new Scripting Environment.

Methods

The following methods were provided by the `TheView` object and must be migrated as explained:

`Cancel()`

In the new scripting environment the same can be achieved with: `lastform.Close(); // Use .NET Form.Close()`

`IsFormOpen(Name as String) as Boolean`

Since for .NET Forms there is a distinction between showing a Form and instantiating a Form, the previous concept does not directly translate. Instead the user can ask if a certain Form is currently shown. For example:

```
var objFormPencilSelector = CreateForm("PencilSelector");
var objFormColorSelector = CreateForm("ColorSelector");
   ...
// Anywhere in code ...

if(objFormColorSelector.Visible)
{
   ...
}
```

`FormFind(Name as String) as Object`

The new Scripting Environment allows you to instantiate more Forms of the same kind. In the old Scripting Environment each Form could only exist once (as a Singleton). Thus there is no equivalent of `FormFind()`. In the new Scripting Environment.

OpenDoc(File as String)
The same can be achieved with: `Application.OpenDocument( File as String )`

PumpData()
This corresponds to the built-in function `doevents()` which processes all Windows messages currently in the message queue.

RunClick(), RunInitialize(), RunTerminate()
There is no direct replacement for these methods. Call the corresponding handlers directly instead.

Properties

The following properties were provided by the `TheView` object and must be migrated as explained:

ToolTipText as String
To use tooltips in the new scripting environment, the .NET infrastructure can be used. This allows fine-grained control of tooltip behaviour (adjusting delays, when to show, etc). For example, to provide tooltips for a Form with two controls, the following code could to be added to the Form's `Load` event handler:

```
//Occurs whenever the user loads the form.
function MyForm_Load( objSender, e_EventArgs )
{
    // Create the ToolTip and associate with the Form container.
    var toolTip = CLR.Create("System.Windows.Forms.ToolTip");

    // Set up the delays for the ToolTip.
    toolTip.AutoPopDelay = 3000;
    toolTip.InitialDelay = 1000;
    toolTip.ReshowDelay = 500;

    // Force the ToolTip text to be displayed whether or not
    // the form is active.
    toolTip.ShowAlways = true;

    // Set up the ToolTip text for several Controls.
    toolTip.SetToolTip(objSender.ProgressBar1,
           "Shows the progress of the operation");
    toolTip.SetToolTip(objSender.Button1,
           "Click Button to start the processing");
}
```

Color as Long
Since all Form/controls in the new Scripting Environment are .NET controls from the `System.Windows.Forms` namespace, the possibilities to modify colors, background image, fonts, and all other visual aspects are numerous. For example, every Visual Component has the properties `BackColor` and `ForeColor` to modify the visual appearance. The following handler could be used to change the color of a button at runtime:

```
function TestForm_Button1_Click( objSender, e_EventArgs )
{
    objSender.BackColor = CLR.Static( "System.Drawing.Color" ).SlateBlue;
}
```

Please refer to the .NET documentation to find out more about this topic:

http://msdn.microsoft.com/en-us/library/system.windows.forms.aspx

2 IDE Plugins

Authentic Desktop allows you to create your own IDE plug-ins and integrate them into Authentic Desktop.

Use plug-ins to:

- Configure your version of Authentic Desktop, add commands through menus, icons, buttons etc.
- React to events from Authentic Desktop.
- Run your specific code within Authentic Desktop with access to the complete Authentic Desktop API

Authentic Desktop expects your plug-in to implement the IXMLSpyPlugIn interface. See ATL sample files for an example using C++. The relevant files are in the following folder of your Authentic Desktop installation:

```
C:\Documents and Settings\<username>\My Documents\Altova\Authentic2012
\AuthenticExamples\CppIDEPlugin
```

For a sample using VisualBasic, look in the following folder of your Authentic Desktop installation:

```
C:\Documents and Settings\<username>\My Documents\Altova\Authentic2012
\AuthenticExamples\XMLSpyPlugInActiveX
```

2.1 Registration of IDE PlugIns

Authentic Desktop maintains a specific key in the Registry where it stores all registered IDE plug-ins:

HKEY_CURRENT_USER\Software\Altova\XML Spy\PlugIns

All values of this key are treated as references to registered plug-ins and must conform to the following format:

Value name:	ProgID of the plug-in
Value type:	must be REG_SZ
Value data:	CLSID of the component

Each time the application starts the values of the "PlugIns" key is scanned, and the registered plug-ins are loaded.

Register plug-in manually
To register a plug-in manually, use the "Customize" dialog box of the Authentic Desktop "Tools" menu. Use the "Add Plug-In..." button to specify the DLL that implements your plug-in. Authentic Desktop registers the DLL as a COM server and adds the corresponding entry in its "PlugIns" key.

If you experience problems with manual registration you can check if the CLSID of your plug-in is correctly registered in the "PlugIns" key. If this is not the case, the name of your plug-in DLL was probably not sufficiently unique. Use a different name or perform direct registration.

Register plug-in directly
A plug-in can be directly registered as an IDE plug-in by first registering the DLL and then adding the appropriate value to the "PlugIns" key of Authentic Desktop during plug-in setup for example. The new plug-in will be activated the next time Authentic Desktop is launched.

2.2 ActiveX Controls

ActiveX controls are supported. Any IDE PlugIn which is also an ActiveX control will be displayed in a Dialog Control Bar. A sample PlugIn that is also an ActiveX control is included in the `XMLSpyPlugInActiveX` folder in the `Examples` folder of your application folder.

2.3 Configuration XML

The IDE plug-in allows you to change the user interface (UI) of Authentic Desktop. This is done by describing each separate modification using an XML data stream. The XML configuration is passed to Authentic Desktop using the GetUIModifications method of the IXMLSpyPlugIn interface.

The XML file containing the UI modifications for the IDE PlugIn, must have the following structure:

```
<ConfigurationData>
 <ImageFile>path To image file</ImageFile>
 <Modifications>
  <Modification>
   ...
  </Modification>
  ...
 </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for the new menu items which are added to the UI of Authentic Desktop by the plug-in. The path to the file containing the images is set using the ImageFile element. Each image must be 16 x 16 pixels using max. 256 colors. The image references must be arranged from left to right in a single (<ImageFile>...) line. The rightmost image index value, is zero.

The Modifications element can have any number of Modification child elements. Each Modification element defines a specific change to the standard UI of Authentic Desktop. Starting with version 4.3, it is also possible to remove UI elements from Authentic Desktop.

Structure of Modification elements
All Modification elements consist of the following two child elements:

```
<Modification>
 <Action>Type of action</Action>
 <UIElement Type="type of UI element">
 </UIElement>
</Modification>
```

Valid values for the Action element are:

Add - to add the following UI element to Authentic Desktop
Hide - to hide the following UI element in Authentic Desktop
Remove - to remove the UI element from the "Commands" list box, in the customize dialog

You can combine values of the Action element e.g. "Hide Remove"

The UIElement element describes any new, or existing UI element for Authentic Desktop. Possible elements are currently: new toolbars, buttons, menus or menu items. The **type** attribute, defines which UI element is described by the XML element.

Common UIElement children
The ID and Name elements are valid for all different types of XML UIElement fragments. It is however possible, to ignore one of the values for a specific type of UIElement e.g. Name is ignored for a separator.

```
<ID></ID>
<Name></Name>
```

If UIElement describes an existing element of the UI, the value of the ID element is predefined by Authentic Desktop. Normally these ID values are not known to the public. If the XML fragment describes a new part of the UI, then the ID is arbitrary and the value should be less than 1000.
The Name element sets the textual value. Existing UI elements can be identified just by name, for e.g. menus and menu items with associated sub menus. For new UI elements, the Name element sets the caption e.g. the title of a toolbar, or text for a menu item.

Toolbars and Menus
To define a toolbar its necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name, or by the ID if it is known. To create a **new** toolbar both values must be set. The **type** attribute must be equal to "ToolBar".

```
<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 short description string which is displayed in the status bar, when the mouse pointer is over the associated command (button or menu item). ImageID defines the index of the icon the external image file. Please note that all icons are stored in one image file.

To define a toolbar button create an UIElement with this structure:

```
<UIElement Type="ToolBarItem">
 <!--don't reuse local IDs even the commands do the same-->
 <ID>5</ID>
 <Name>Open file from repository...</Name>
 <!--Set Place To -1 If this is the first button To be inserted-->
 <Place>-1</Place>
 <ImageID>0</ImageID>
 <ToolBarID>1</ToolBarID>
 <!--instead of the toolbar ID the toolbar name could be used-->
 <ToolBarName>TestPlugIn</ToolBarName>
</UIElement>
```

Additional elements to declare a toolbar button are Place, ToolBarID and ToolBarName. ToolBarID and ToolBarName are used to identify the toolbar which contains the new or existing button. The textual value of ToolBarName is case sensitive. The (UIElement) **type** attribute must equal "ToolBarItem".

To define a menu item, the elements MenuID, Place and Parent are available in addition to the standard elements used to declare a command. MenuID can be either 128 or 129. Please see "Toolbars and Menus" for more information on these values.

The Parent element is used to identify the **menu** where the new menu entry should be inserted. As sub menu items have no unique Windows ID, we need some other way to identify the parent of the menu item.

The value of the Parent element is a path to the menu item.
The text value of the Parent element, must equal the **parent menu name** of the submenu, where the submenu name is separated by a colon. If the menu has no parent, because its not a submenu, add a colon to the beginning of the name. The **type** attribute must be set to "MenuItem". Example for an UIElement defining a menu item:

```
<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>
```

Authentic Desktop makes it possible to add toolbar separators and menus if the value of the ID element is set to 0.

2.4 ATL sample files

The following pages show how to create a simple Authentic Desktop IDE plug-in DLL using ATL. To build the DLL it is necessary to know about ATL, the wizards that generate new ATL objects, as well as MS VisualStudio.

To access the API the implementation imports the Type Library of Authentic Desktop. The code reads various properties and calls methods using the smart pointers provided by the #import statement.

In addition, the sample code uses the MFC class CString and the ATL conversion macros such as W2T.

At a glance the steps to create an ATL DLL are as follows:

1. Open VisualStudio and select "New..." from the "File" menu.
2. Select the "Projects" tab.
3. Select "ATL COM AppWizard" and type in a project name.
4. Select "Support for MFC" if you want to use MFC classes, or if you want to create a project for the sample code.

   Having created the project files you can add an ATL object to implement the IXMLSpyPlugIn interface:

1. Select "New ATL Object..." from the "Insert" menu.
2. Select "Simple Object" from the wizard and click "Next".
3. Type in a name for the object.
4. On the "Attributes" tab, select "Custom" for the type of interface, and disable Aggregation.

These steps produce the skeleton code for the implementation of the IDE plug-in interface. Please see the following pages on how to modify the code and achieve some basic functionality.

2.4.1 Interface description (IDL)

The IDL of the newly created ATL object contains a declaration for one COM interface.

- This interface declaration must be replaced by the declaration of IXMLSpyPlugIn as shown below.
- The IDL must also contain the definition of the SPYUpdateAction enumeration.
- Replace the generated default interface name, (created by the wizard) with "IXMLSpyPlugIn" in the coclass declaration. The IDL should then look something like the example code below:

Having created the ATL object, you then need to implement the IDE plug-in interface of Authentic Desktop.

```
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-A1F3-270692EO8AFC),

//    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 -----
 };
};
```

2.4.2 Class definition

In the class definition of the ATL object, several changes must be made. The class has to derive from IXMLSpyPlugIn, the "Interface Map" needs an entry for IXMLSpyPlugIn, and the methods of the IDE plug-in interface must be declared:

```
#ifndef __XMLSPYIDEPLUGIN_H_
#define __XMLSPYIDEPLUGIN_H_

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn
class ATL_NO_VTABLE CXMLSpyIDEPlugIn :
 public CComObjectRootEx<CComSingleThreadModel>,
 public CComCoClass<CXMLSpyIDEPlugIn, &CLSID_XMLSpyIDEPlugIn>,
 public IXMLSpyPlugIn
{
public:
 CXMLSpyIDEPlugIn()
 {
 }

DECLARE_REGISTRY_RESOURCEID(IDR_XMLSPYIDEPLUGIN)
DECLARE_NOT_AGGREGATABLE(CXMLSpyIDEPlugIn)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CXMLSpyIDEPlugIn)
 COM_INTERFACE_ENTRY(IXMLSpyPlugIn)
END_COM_MAP()

// IXMLSpyIDEPlugIn
public:
 virtual HRESULT _stdcall OnCommand(long nID, IDispatch* pXMLSpy);

 virtual HRESULT _stdcall OnUpdateCommand(long nID, IDispatch* pXMLSpy,
SPYUpdateAction* pAction);

 virtual HRESULT _stdcall OnEvent(long nEventID, SAFEARRAY **arrayParameters,
IDispatch* pXMLSpy, VARIANT* pReturnValue);

 virtual HRESULT _stdcall GetUIModifications(BSTR* pModificationsXML);

 virtual HRESULT _stdcall GetDescription(BSTR* pDescription);
};

#endif //__XMLSPYIDEPLUGIN_H_
```

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 Seperator 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);
}
```

2.5 IXMLSpyPlugIn

See also

Methods

OnCommand
OnUpdateCommand
OnEvent
GetUIModifications
GetDescription

Description

If a DLL is added to Authentic Desktop as an IDE plug-in, it is necessary that it registers a COM component that answers to an IXMLSpyPlugIn interface with the reserved uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85), for it to be recognized as a plug-in.

2.5.1 OnCommand

See also

Declaration: `OnCommand( nID as long, pXMLSpy as IDispatch)`

Description

The `OnCommand()` method of the interface implementation, is called each time a command added by the the IDE plug-in (menu item or toolbar button) is processed. nID stores the command ID defined by the `ID` element of the respective `UIElement`.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of Authentic Desktop.

Example

```
Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
    If (Not (pXMLSpy Is Nothing)) Then
        Dim objDlg
        Dim objDoc As XMLSpyLib.Document
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If nID = 3 Or nID = 5 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "XML Files (*.xml)|*.xml|All Files (*.*)|*.*||"
            objDlg.FilterIndex = 1
            objDlg.ShowOpen

            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.Documents.OpenFile(objDlg.FileName, False)
                Set objDoc = Nothing
            End If
        End If

        If nID = 4 Or nID = 6 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "All Files (*.*)|*.*||"
            objDlg.Flags = cdlOFNPathMustExist
            objDlg.ShowSave

            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.ActiveDocument

                If Not (objDoc Is Nothing) Then
                    objDoc.SetPathName objDlg.FileName
                    objDoc.Save
                    Set objDoc = Nothing
                End If
            End If
        End If

        Set objSpy = Nothing
    End If
End Sub
```

2.5.2 OnUpdateCommand

See also

Declaration: `OnUpdateCommand( nID as long, pXMLSpy as IDispatch) as SPYUpdateAction`

Description

The `OnUpdateCommand()` method is called each time the visible state of a button or menu item needs to be set. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object.

Possible return values to set the update state are:

```
spyEnable         = 1
spyDisable        = 2
spyCheck          = 4
spyUncheck        = 8
```

Example

```
Public Function IXMLSpyPlugIn_OnUpdateCommand( ByVal nID As Long, ByVal
pXMLSpy As Object) As SPYUpdateAction
    IXMLSpyPlugIn_OnUpdateCommand = spyDisable

    If ( Not ( pXMLSpy Is Nothing)) Then
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If nID = 3 Or nID = 5 Then
            IXMLSpyPlugIn_OnUpdateCommand = spyEnable
        End If
        If nID = 4 Or nID = 6 Then
            If objSpy.Documents.Count > 0 Then
                IXMLSpyPlugIn_OnUpdateCommand = spyEnable
            Else
                IXMLSpyPlugIn_OnUpdateCommand = spyDisable
            End If
        End If
    End If
End Function
```

2.5.3 OnEvent

See also

Declaration: `OnEvent( nEventID as long, arrayParameters as SAFEARRAY( VARIANT), pXMLSpy as IDispatch) as VARIANT`

Description

`OnEvent()` is called each time an event is raised from Authentic Desktop.

Possible values for `nEventID` are:

On_BeforeStartEditing	= 1
On_EditingFinished	= 2
On_FocusChanged	= 3
On_Beforedrag	= 4
On_BeforeDrop	= 5
On_OpenProject	= 6
On_OpenDocument	= 7
On_CloseDocument	= 8
On_SaveDocument	= 9

Events available since Authentic Desktop 4r4:

On_DocEditDragOver	= 10
On_DocEditDrop	= 11
On_DocEditKeyDown	= 12
On_DocEditKeyUp	= 13
On_DocEditKeyPressed	= 14
On_DocEditMouseMove	= 15
On_DocEditButtonUp	= 16
On_DocEditButtonDown	= 17
On_DocEditContextMenu	= 18
On_DocEditPaste	= 19
On_DocEditCut	= 20
On_DocEditCopy	= 21
On_DocEditClear	= 22
On_DocEditSelectionChanged	= 23

Events available since Authentic Desktop 2004:

On_DocEditDragOver	= 10

Events available since Authentic Desktop 2004r4 (type library version 1.4):

On_BeforeOpenProject	= 25
On_BeforeOpenDocument	= 26
On_BeforeSaveDocument	= 27
On_BeforeCloseDocument	= 28
On_ViewActivation	= 29
On_DocEditKeyboardEvent	= 30
On_DocEditMouseEvent	= 31

Events available since Authentic Desktop 2006 SP1 (type library version 1.5):

On_BeforeValidate	= 32

Events available since Authentic Desktop 2007 (type library version 1.6):

On_BeforeShowSuggestions	= 33
On_ProjectOpened	= 34
On_Char	= 35

Events available since Authentic Desktop 2009 (type library version 2.2):

On_Initialize	= 36
On_Running	= 37
On_Shutdown	= 38

Events available since Authentic Desktop 2012 (type library version 2.8):

On_AuthenticBeforeSave	= 39
On_AuthenticContextMenuActivated	= 40
On_AuthenticLoad	= 41
On_AuthenticToolbarButtonClicked	= 42
On_AuthenticToolbarButtonExecuted	= 43
On_AuthenticUserAddedXMLNode	= 44

The names of the events are the same as they appear in the Scripting Environment of Authentic Desktop. For IDE plug-ins the names used are immaterial. The events are identified using the `ID` value.

`arrayParameters` is an array which is filled with the parameters of the currently raised event. Order, type and meaning of the single parameters are available through the scripting environment of Authentic Desktop. The events module of a scripting project, contains predefined functions for all events prior to version 4.4. The parameters passed to the predefined functions are identical to the array elements of the `arrayParameters` parameter.

Events raised from the Authentic View of Authentic Desktop do not pass any parameters directly. An "event" object is used instead. The event object can be accessed through the Document object of the active document.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of Authentic Desktop.

If the return value of `OnEvent()` is set, then neither the IDE plug-in, nor an event handler inside of the scripting environment will get this event afterwards. Please note that all IDE plug-ins get/process the event before the Scripting Environment does.

2.5.4 GetUIModifications

See also

Declaration: `GetUIModifications()` as String

Description

The `GetUIModifications()` method is called during initialization of the plug-in, to get the configuration XML data that defines the changes to the UI of Authentic Desktop. The method is called when the plug-in is loaded for the first time, and at every start of Authentic Desktop.

See also Configuration XML for a detailed description how to change the UI.

Example

```
Public Function IXMLSpyPlugIn_GetUIModifications() As String
    ' GetUIModifications() gets the XML file with the specified modifications
of
    ' the UI from the config.xml file in the plug-in folder
    Dim strPath As String
    strPath = App.Path

    If Len(strPath) > 0 Then
        Dim fso As New FileSystemObject
        Dim file As file

        Set file = fso.GetFile(strPath & "\config.xml")

        If (Not (file Is Nothing)) Then
            Dim stream As TextStream
            Set stream = file.OpenAsTextStream(ForReading)

            ' this replaces the token '**path**' from the XML file with
            ' the actual installation path of the plug-in to get the image
file
            Dim strMods As String
            strMods = stream.ReadAll
            strMods = Replace(strMods, "**path**", strPath)

            IXMLSpyPlugIn_GetUIModifications = strMods
        Else
            IXMLSpyPlugIn_GetUIModifications = ""
        End If
    End If
End Function
```

2.5.5 GetDescription

See also

Declaration: `GetDescription()` as String

Description

`GetDescription()` is used to define the description string for the plug-in entries visible in the Customize dialog box.

Example

```
Public Function IXMLSpyPlugIn_GetDescription() As String
    IXMLSpyPlugIn_GetDescription = "Sample Plug-in for XMLSpy;This Plug-in
demonstrates the implementation of a simple VisualBasic DLL as a Plug-in for
XMLSpy."
End Function
```

3 Application API

The COM-based API of Authentic Desktop (also called the Application API from now on) enables other applications to use the functionality of Authentic Desktop. As a result, it is possible to automate a wide range of tasks, from validating an XML file to modifying complex XML content (with the XMLData interface).

Authentic Desktop and its Application API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the Application API from common development environments, such as those using C, C++, VisualBasic, and Delphi, and with scripting languages like JScript and VBScript.

Execution environments for the Application API
The Application API can be accessed from the following execution environments:

- External programs (described below and in the Overview part of this section)
- From within the built-in Scripting Editor of Authentic Desktop. For a description of the scripting environment, see the section, Scripting Editor.
- Authentic Desktop allows you to create and integrate your own plug-ins into the application using a special interface for plug-ins. A description of how to create plug-ins is given in the section IDE Plug-ins.
- Via an ActiveX Control, which is available if the integration package is installed. For more information, see the section ActiveX Integration.

External programs
In the Overview part of this section, we describe how the functionality of Authentic Desktop can be accessed and automated from external programs.

Using the Application API from outside Authentic Desktop requires an instance of Authentic Desktop to be started first. How this is done depends on the programming language used. See the section, Programming Languages, for information about individual languages.

Essentially, Authentic Desktop will be started via its COM registration. Then the `Application` object associated with the Authentic Desktop instance is returned. Depending on the COM settings, an object associated with an already running Authentic Desktop can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript and VBScript script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C# is a a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#..
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for Applications, Perl, and Python.

Programming points
The following limitations must be considered in your client code:

- Be aware that if your client code crashes, instances of Authentic Desktop may still remain in the system.
- Don't hold references to objects in memory longer than you need them, especially those from the `XMLData` interface. If the user interacts between two calls of your client, then there is no guarantee that these references are still valid.
- Don't forget to disable dialogs if the user interface is not visible.
- See Error handling in JScript (and in C# and Java) for details of how to avoid annoying error messages.
- Free references explicitly if you are using C or C++.

This documentation
This documentation section about the Application API is broadly divided into two parts.

- The first part consists of an Overview, which describes the object model for the API and explains how the API is accessed via various programming languages.
- The second part is a reference section (Interfaces and Enumerations) that contains descriptions of the interface objects of the Application API.

3.1 Overview

This overview of the Application API is organized as follows:

- The Object Model describes the relationships between the objects of the Application API.
- Programming Languages explains how the most commonly used programming languages (JScript, VBScript, C#, and Java) can be used to access the functionality of the Application API. Code listings from the example files supplied with your application package are used to describe basic mechanisms.
- The DOM and XMLData explains the relationship between the Application API's `XMLData` interface and the DOM.
- Obsolete: Authentic View Row Operations supplies information about obsolete objects for Authentic View table row operations.
- Obsolete: Authentic View Editing Operations supplies information about obsolete objects for Authentic View editing operations.

3.1.1 Object Model

The starting point for every application which uses the Application API is the `Application` object. This object contains general methods like import/export support and references to the open documents and any open project.

To create an instance of the `Application` object, call `CreateObject("XMLSpy.Application")` from VisualBasic or a similar function from your preferred development environment to create a COM object. There is no need to create any other objects in order to use the complete Application API; it is in fact not even possible. All other interfaces are accessed through other objects with the `Application` object as the starting point.

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`

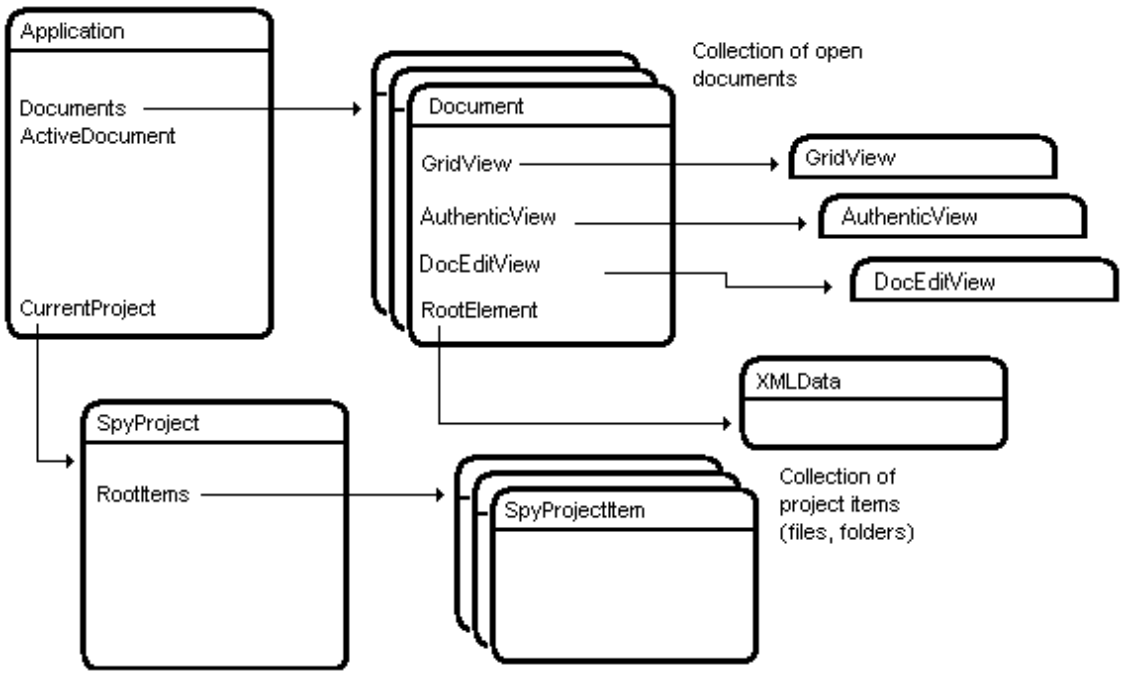
If only Authentic Desktop is installed (and not XMLSpy), then

```
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 picture below shows the links between the main objects of the Application API:



The application object consists of the following parts:

1. Document collection and reference to the active document.
2. Reference to current project and methods for creating and opening projects.
3. Methods to support the export to and import from databases, text files, and Word documents.
4. URL management.
5. Methods for macro menu items.

Once you have created an `Application` object you can start using the functionality of Authentic Desktop. In most cases, you either open a project and access the documents from there or you directly open a document via the `Documents` interface.

3.1.2 Programming Languages

Programming languages differ in the way they support COM access. A few examples for the most frequently used languages (*links below*) will help you get started. The code listings in this section show how basic functionality can be accessed. This basic functionality is included in the files in the API Examples folder and can be tested straight away. The path to the API Examples folder is given below:

Windows XP	`C:/Documents and Settings/<username>/My Documents/ Altova/Authentic20XX/AuthenticExamples/API/`
Windows Vista, Windows 7	`C:/Users/<username>/Documents/ Altova/Authentic20XX/AuthenticExamples/API/`

JScript
The JScript listings demonstrate the following basic functionality:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling
- Events

VBScript
VBScript is different than JScript only syntactically; otherwise it works in the same way. The listings below describe is an example of how VBScript can be used. For more information, refer to the JScript examples.

- Events: Shows how events are handled using VBScript.

C#
C# can be used to access the Application API functionality. The code listings show how to access the API for certain basic functionality.

- Start Authentic Desktop: Starts Authentic Desktop, which is registered as an automation server, or activates the application if it is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with Authentic Desktop and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off: Shows how to listen to Authentic Desktop events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml: Opens another example document.
- Toggle View Mode: Changes the view of all open documents between Browser View and Authentic View. The code shows how to iterate through open documents.
- Validate: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shutdown Authentic Desktop: Stops Authentic Desktop.

Java
The Authentic Desktop API can be accessed from Java code. This section explains how some basic Authentic Desktop functionality can be accessed from Java code. It is organized into the

following sub-sections:

- Mapping Rules for the Java Wrapper
- Example Java Project
- Application Startup and Shutdown
- Simple Document Access
- Iterations
- Use of Out-Parameters
- Event Handlers

JScript

This section contains listings of JScript code that demonstrate the following basic functionality:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling
- Events
- Example: Bubble Sort Dynamic Tables

Example files
The code listings in this section are available in example files that you can test as is or modify to suit your needs. The JScript example files are located in the `JScript` folder of the API Examples folder:

Windows XP	`C:/Documents and Settings/<username>/My Documents/Altova/Authentic20XX/AuthenticExamples/API/`
Windows Vista, Windows 7	`C:/Users/<username>/Documents/Altova/Authentic20XX/AuthenticExamples/API/`

The example files can be run in one of two ways:

- *From the command line:* Open a command prompt window and type the name of one of the example scripts (for example, `Start.js`). The Windows Scripting Host that is packaged with newer versions of Windows (XP, Vista, 7) will execute the script.
- *From Windows Explorer:* In Windows Explorer, browse for the JScript file and double-click it. The Windows Scripting Host that is packaged with newer versions of Windows (XP, Vista, 7) will execute the script. After the script is executed, the command console gets closed automatically.

Start Application

The JScript below starts the application and shuts it down. If an instance of the application is already running, the running instance will be returned.

Script listing
The JScript listing below is explained with comments in the code.

```
// Initialize application's COM object. This will start a new instance of the
application and
```

```
// return its main COM object. Depending on COM settings, a the main COM object of an
already
// running application might be returned.
try
{
        objAuthentic = WScript.GetObject("", "AuthenticDesktop.Application");
}
catch(err)
{
        Exit("Can't access or create AuthenticDesktop.Application");
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objAuthentic.Visible = true;

WScript.Echo("Hello");

objAuthentic.Visible = false; // will shutdown application if it has no more COM
connections
//objAuthentic.Visible = true;        // will keep application running with UI visible
```

Running the script
The JScript code listed above is available in the file `Start.js` located in the `JScript` folder of the API Examples folder:

Windows XP	`C:/Documents and Settings/<username>/My Documents/Altova/Authentic20XX/AuthenticExamples/API/`
Windows Vista, Windows 7	`C:/Users/<username>/Documents/Altova/Authentic20XX/AuthenticExamples/API/`

To run the script, start it from a command prompt window or from Windows Explorer.

Simple Document Access

The JScript listing below shows how to open documents, set a document as the active document, iterate through the open documents, and close documents.

Running the script
The JScript code listed below is available in the file `DocumentAccess.js` located in the `JScript` folder of the API Examples folder:

Windows XP	`C:/Documents and Settings/<username>/My Documents/Altova/Authentic20XX/AuthenticExamples/API/`
Windows Vista, Windows 7	`C:/Users/<username>/Documents/Altova/Authentic20XX/AuthenticExamples/API/`

To run the script, start it from a command prompt window or from Windows Explorer.

Script listing
The JScript listing below is explained with comments in the code.

```
// Initialize application's COM object. This will start a new instance of the
```



```
application and
// return its main COM object. Depending on COM settings, a the main COM object of an
already
// running application might be returned.
try
{
        objAuthentic = WScript.GetObject("", "AuthenticDesktop.Application");
}
catch(err)
{
        Exit("Can't access or create AuthenticDesktop.Application");
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objAuthentic.Visible = true;

// ************************** code snippet for "Simple Document Access"
***************************************

// Locate examples via USERPROFILE shell variable. The path needs to be adapted to major
release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\Authentic2012\\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 grid view.
objDoc2.SwitchViewMode(5);     // SPYViewModes.spyViewAuthentic = 5

// Now switch back to the document opened first
objDoc1.SetActiveDocument();

// ************************** code snippet for "Simple Document Access"
***************************************

// ************************** code snippet for "Iteration"
***************************************

// 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);


// ************************** code snippet for "Iteration"
***************************************

//objAuthentic.Visible = false;              // will shutdown application if it has no
more COM connections
objAuthentic.Visible = true;  // will keep application running with UI visible
```

Iteration

The JScript listing below shows how to iterate through the open documents.

Running the script

You can test this script by copying the listing below to a file, naming the file with a `.js` extension, and running the file from either the command line or Windows Explorer. You could copy the file to the `JScript` folder of the API Examples folder:

Windows XP	`C:/Documents and Settings/<username>/My Documents/ Altova/Authentic20XX/AuthenticExamples/API/`
Windows Vista, Windows 7	`C:/Users/<username>/Documents/ Altova/Authentic20XX/AuthenticExamples/API/`

To run the script, start it from a command prompt window or from Windows Explorer.

Script listing

The JScript listing below is explained with comments in the code.

```
// Initialize application's COM object. This will start a new instance of the
application and
// return its main COM object. Depending on COM settings, a the main COM object of an
already
// running application might be returned.
try
{
        objAuthentic = WScript.GetObject("", "AuthenticDesktop.Application");
}
catch(err)
{
        Exit("Can't access or create AuthenticDesktop.Application");
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objAuthentic.Visible = true;

// ************************** code snippet for "Simple Document Access"
***************************************

// Locate examples via USERPROFILE shell variable. The path needs to be adapted to major
release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\Authentic2012\\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 grid view.
objDoc2.SwitchViewMode(1);     // SPYViewModes.spyViewText = 1

// Now switch back to the document opened first
objDoc1.SetActiveDocument();

// *************************** code snippet for "Simple Document Access"
***************************************

// *************************** code snippet for "Iteration"
***************************************

// 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);


// *************************** code snippet for "Iteration"
***************************************

//objAuthentic.Visible = false;              // will shutdown application if it has no
more COM connections
objAuthentic.Visible = true;  // will keep application running with UI visible
```

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.

JScript error handling

JScript provides a try-catch mechanism to deal with errors raised from COM calls. An error object containing the necessary information is declared. The listing below shows how.

```
// 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.");
        }
}
```

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;
objDoc = objSpy.Documents.OpenFile ("C:\\Program
Files\\Altova\\Authentic2012\\AuthenticExamples\\OrgChart.xml", 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();
```

Example: Bubble Sort Dynamic Tables

The following JScript snippet will sort any dynamic table by the table column identified by the current cursor position. The sort process is performed on screen. The undo buffer is available for all performed operations.

If you can run JScript on you computer - as you most likely will - copy the following code into a file with extension 'js'. Execute the script by double-clicking it in Windows Explorer, or run it from the command line.

```
// some useful XMLSpy enum constants
var spyAuthenticTag = 6;
var spyAuthenticTable = 9;
var spyAuthenticTableRow = 10;
var spyAuthenticTableColumn = 11;

var spyAuthenticRangeBegin = 2;

// example call for the sort table function
try
{
  var objSpy = WScript.GetObject("", "XMLSpy.Application");
  // If only Authentic is installed (and XMLSpy is not installed) use:
  // var objSpy = WScript.GetObject("", "AuthenticDesktop.Application")

  // use current selection to indicate which column to sort
  SortCurrentTable (objSpy.ActiveDocument.AuthenticView.Selection);
}
catch (err)
  { WScript.Echo ("Please open a document in authentic view, and select a
  table column\n" +
                  "Error : (" + (err.number & 0xffff) + ")" +
err.description); }

// we assume that XMLSpy is running, a document with a dynamic table
// is open, and the cursor is in a table column that will
```

```
// be used for sorting.
function SortCurrentTable (objCursor)
{
  if (objCursor.IsInDynamicTable())
  {
    // calculate current column index
    var nColIndex = 0;
    while (true)
    {
      // go left column-by-column
      try { objCursor.GotoPrevious(spyAuthenticTableColumn); }
      catch (err) { break; }
      nColIndex++;
    }

    // count number of table rows, so the bubble loops become simpler.
    // goto begin of table
    var objTableStart =
    objCursor.ExpandTo(spyAuthenticTable).CollapsToBegin().Clone();
    var nRows = 1;
    while (true)
    {
      // go down row-by-row
      try { objTableStart.GotoNext(spyAuthenticTableRow); }
      catch (err) { break; }
      nRows++;
    }

    // bubble sort through table
    for (var i = 0; i < nRows - 1; i++)
    {
      // select correct column in first table row
      var objBubble =
      objCursor.ExpandTo(spyAuthenticTable).CollapsToBegin().Clone();
      objBubble.Goto (spyAuthenticTableColumn, nColIndex,
      spyAuthenticRangeBegin).ExpandTo(spyAuthenticTag);

      // bubble this row down as far as necessary
      for (var j = 0; j < nRows - i - 1; j++)
      {
        var strField1 = objBubble.Text;
        // now look for the comparison table cell: start of next row and right
        of the correct column
        var strField2 = objBubble.GotoNext(spyAuthenticTableRow).
                                  Goto (spyAuthenticTableColumn, nColIndex,
        spyAuthenticRangeBegin).
                                  ExpandTo(spyAuthenticTag).Text;
        if (strField1 > strField2)
        {
          objBubble.MoveRowUp();  // swap the rows
          // and re-calculate objBubble to select the cell to bubble
          objBubble.GotoNext(spyAuthenticTableRow).
                    Goto (spyAuthenticTableColumn, nColIndex,
          spyAuthenticRangeBegin).
                    ExpandTo(spyAuthenticTag);
        }
      }
    }
  }
  else
    WScript.Echo ("please, select a table cell first");
}
```

VBScript

VBScript is syntactically different than JScript but works in the same way. This section contains a listing showing how events are used with VBScript and an example.

For information about other functionality, refer to the JScript examples listed below:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling

Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using VBScript.

The method `WScript.ConnectObject` is used to receive events.

```
' 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")

' create document object and connect to its events
objSpy.Visible = True
Set objDoc = objSpy.Documents.OpenFile ("C:\\Program
Files\\Altova\\XMLSPY2004\\Examples\\OrgChart.xml", 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
```

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.

```
Rem ----------------------------------------------------------------------
Rem VBScript example that demonstrates how to use events.
Rem ----------------------------------------------------------------------
```

```
' 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

' Here starts the main code.
' Find out user's personal folder and locate one of the installed XMLSpy 2011 examples.
Set WshShell = WScript.CreateObject("WScript.Shell")
personalFolder = WshShell.ExpandEnvironmentStrings("%UserProfile%")
xmlspyExamplesFolder = personalFolder & "\My Documents\Altova\XMLSpy2011\Examples\"
docPath = xmlspyExamplesFolder & "OrgChart.xml"

' Start/access XMLSpy and connect to its automation interface.
Set objSpy = GetObject("", "XMLSpy.Application")
' Make the UI of XMLSpy visible.
objSpy.Visible = True

' 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
```

C#

The C# programming language can be used to access the Application API functionality. You could use Visual Studio 2008 or Visual Studio 2010 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 AuthenticDesktop Type Library by clicking **Project | Add Reference**. The Add Reference dialog pops up, displaying a list of installed COM components. Select the AuthenticDesktop Type Library component from the list to add it.
3. Enter the code you want.
4. Compile the code and run it.

Example C# project

Your AuthenticDesktop package contains an example C# project, which is located in the C# folder of the API Examples folder:

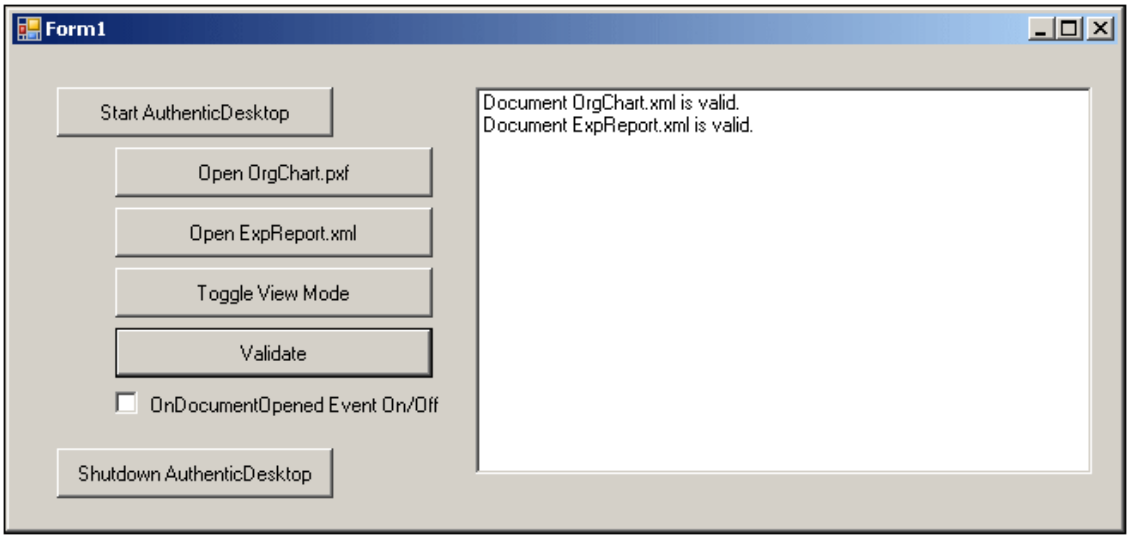
Windows XP	C:/Documents and Settings/<username>/My Documents/ Altova/Authentic20XX/AuthenticExamples/API/
Windows Vista, Windows 7	C:/Users/<username>/Documents/ Altova/Authentic20XX/AuthenticExamples/API/

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

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.

What the code listing below does

The example code listing below creates a simple user interface (*screenshot below*) with buttons that invoke basic AuthenticDesktop operations:



- Start Authentic Desktop: Starts Authentic Desktop, which is registered as an automation server, or activates the application if it is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with Authentic Desktop and opens it. If this document is already open it becomes the active document.
- Open ExpReport.xml: Opens another example document.
- Toggle View Mode: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- Validate: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shut down Authentic Desktop: Stops Authentic Desktop.

You can modify the code (of the code listing below or of the example C# project in the API Examples folder) in any way you like and run it.

Compiling and running the example

In the API Examples folder, double-click the file `AutomateAuthenticDesktop_VS2008.sln` (to open it in Visual Studio 2008) or the file `AutomateAuthenticDesktop_VS2010.sln` (to open it in Visual Studio 2010). 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 explicitely 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)
```

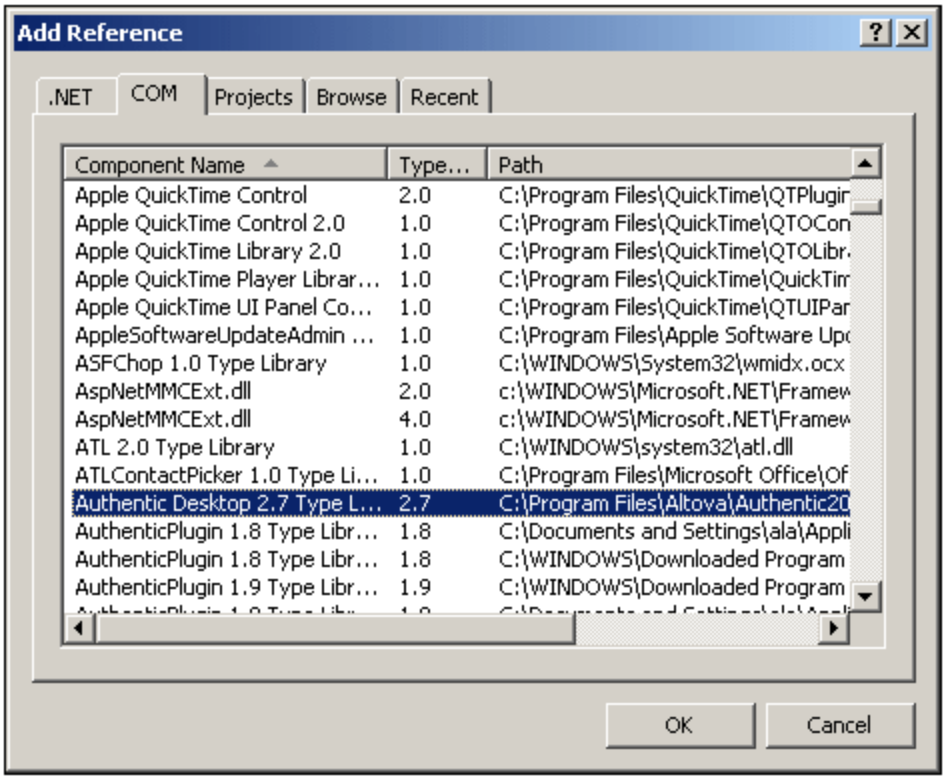


```
            {
                if (checkBoxEventOnOff.Checked)
                    AuthenticDesktop.OnDocumentOpened += new XMLSpyLib.
_IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    AuthenticDesktop.OnDocumentOpened -= new XMLSpyLib.
_IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
            }
        }
    }
}
```

Add Reference to XMLSpy Application API

To add the application's type library as a reference in a .NET project, do the following:

1. With the .NET project open, click **Project | Add Reference**. The Add Reference dialog (*screenshot below*) pops up, displaying a list of installed COM components.



2. Select `AuthenticDesktop X.Y Type Library` from the component list and finish.

Declare a variable to access the AuthenticDesktop API

```
// An instance of AuthenticDesktop is accessed via its automation interface.
XMLSpyLib.Application AuthenticDesktop;
```

Application Startup and Shutdown

In the code snippets below, the methods `StartAuthenticDesktop_Click` and `ShutdownAuthenticDesktop_Click` are those assigned to buttons in the AutomateAuthenticDesktop example that, respectively, start up and shut down the application.

This example is located in the C# folder of the API Examples folder (*see the file Form1.cs*):

Windows XP	C:/Documents and Settings/<username>/My Documents/ Altova/Authentic20XX/AuthenticExamples/API/
Windows Vista, Windows 7	C:/Users/<username>/Documents/ Altova/Authentic20XX/AuthenticExamples/API/

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

Starting Authentic Desktop

The following code snippet from the AutomateAuthenticDesktop example 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 shows how to shut down the application.

```
        // Handler for the "Shut down AuthenticDesktop" button
        // Shut down application instance by explicitely releasing the COM object.
        private void shutdownAuthenticDesktop_Click(object sender, EventArgs e)
        {
            if (AuthenticDesktop != null)
            {
                // Allow shut down of AuthenticDesktop by releasing the UI
                AuthenticDesktop.Visible = false;

                // Explicitly release the COM object
                try
                {
                    while (System.Runtime.InteropServices.Marshal
.ReleaseComObject(AuthenticDesktop) > 0) ;
                }
                finally
                {
                    // Avoid subsequent access to this object.
                    AuthenticDesktop = null;
                }
            }
        }
```


Opening Documents

The code snippets below (from the AutomateAuthenticDesktop example) show how two files are opened via two separate methods assigned to two buttons in the user interface. Both methods use the same Application API access mechanism: `Documents.OpenFile(string, boolean)`.

The AutomateAuthenticDesktop example (*see the file Form1.cs*) is located in the C# folder of the API Examples folder:

Windows XP	`C:/Documents and Settings/<username>/My Documents/Altova/Authentic20XX/AuthenticExamples/API/`
Windows Vista, Windows 7	`C:/Users/<username>/Documents/Altova/Authentic20XX/AuthenticExamples/API/`

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

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)
        {
            // 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);
        }
```

The file opened last will be the active file.

Iterating through Open Documents

The code snippet below (from the AutomateAuthenticDesktop example; *see the file Form1.cs*) shows how to iterate through open documents. A condition is then tested within the iteration loop, and the document view is switched between Text View and Authentic View.

```
        // Handler for the "Toggle View Mode" button
        private void toggleView_Click(object sender, EventArgs e)
        {
            // Make sure there's a running Authentic Desktop instance, and that it's
visible
            StartAuthenticDesktop_Click(null, null);
```

```
            // Iterate through all open documents and toggle the current view between
Text View and Authentic View.
            foreach (XMLSpyLib.Document doc in AuthenticDesktop.Documents)
                if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewAuthentic)
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewBrowser);
                else
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
        }
```

The AutomateAuthenticDesktop example example is located in the C# folder of the API Examples folder:

Windows XP	C:/Documents and Settings/<username>/My Documents/ Altova/Authentic20XX/AuthenticExamples/API/
Windows Vista, Windows 7	C:/Users/<username>/Documents/ Altova/Authentic20XX/AuthenticExamples/API/

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

Errors and COM Output Parameters

The code snippet below (from the AutomateAuthenticDesktop example) shows how to handle errors and COM output parameters. The method AuthenticDesktop.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData) uses output parameters that are used, in the code snippet below, to generate an error-message text.

The AutomateAuthenticDesktop example (*see the file Form1.cs*) is located in the C# folder of the API Examples folder:

Windows XP	C:/Documents and Settings/<username>/My Documents/ Altova/Authentic20XX/AuthenticExamples/API/
Windows Vista, Windows 7	C:/Users/<username>/Documents/ Altova/Authentic20XX/AuthenticExamples/API/

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

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);
            }
        }
```

Events

The code snippet below (from the AutomateAuthenticDesktop example) lists the code for two event handlers. The AutomateAuthenticDesktop example (*see the file Form1.cs*) is located in the C# folder of the API Examples folder:

Windows XP	C:/Documents and Settings/<username>/My Documents/ Altova/Authentic20XX/AuthenticExamples/API/
Windows Vista, Windows 7	C:/Users/<username>/Documents/ Altova/Authentic20XX/AuthenticExamples/API/

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

```
        delegate void addListBoxItem_delegate(string sText);
        // Called from the UI thread
        private void addListBoxItem(string sText)
        {
            listBoxMessages.Items.Add(sText);
        }
        // Wrapper method to call UI control methods from a worker thread
        void syncWithUIthread(Control ctrl, addListBoxItem_delegate methodToInvoke,
String sText)
        {
            // Control.Invoke: Executes on the UI thread, but calling thread waits for
completion before continuing.
            // Control.BeginInvoke: Executes on the UI thread, and calling thread
doesn't wait for completion.
            if (ctrl.InvokeRequired)
                ctrl.BeginInvoke(methodToInvoke, new Object[] { sText });
        }

        // Event handler for OnDocumentOpened event
        private void handleOnDocumentOpened(XMLSpyLib.Document i_ipDocument)
        {
            String sText = "";

            if (i_ipDocument.Name.Length > 0)
                sText = "Document " + i_ipDocument.Name + " was opened!";
            else
```

```
                sText = "An empty document was created.";

            // Synchronize the calling thread with the UI thread because
            // COM events are triggered from a working thread
            addListBoxItem_delegate methodToInvoke = new addListBoxItem_delegate
(addListBoxItem);
            // Call syncWithUIthread with the following arguments:
            // 1 - listBoxMessages - list box control to display messages from COM
events
            // 2 - methodToInvoke  - a C# delegate which points to the method which will
be called from the UI thread
            // 3 - sText           - the text to be displayed in the list box
            syncWithUIthread(listBoxMessages, methodToInvoke, sText);
        }


        private void checkBoxEventOnOff_CheckedChanged(object sender, EventArgs e)
        {
            if (AuthenticDesktop != null)
            {
                if (checkBoxEventOnOff.Checked)
                    AuthenticDesktop.OnDocumentOpened += new XMLSpyLib.
_IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    AuthenticDesktop.OnDocumentOpened -= new XMLSpyLib.
_IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
            }
        }
```

Java

The Application API can be accessed from Java code. To allow accessing the XMLSpy automation server directly from Java code, the libraries listed below must reside in the `classpath`. They are installed in the folder: `JavaAPI` in the XMLSpy application folder.

- `AltovaAutomation.dll`: a JNI wrapper for Altova automation servers
- `AltovaAutomation.jar`: Java classes to access Altova automation servers
- `XMLSpyAPI.jar`: Java classes that wrap the XMLSpy automation interface
- `XMLSpyAPI_JavaDoc.zip`: a `Javadoc` file containing help documentation for the Java API

Note: In order to use the Java API, the DLL and Jar files must be on the Java Classpath.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details of the example Java project, see the section, Example Java Project.

Rules for mapping the Application API names to Java

The rules for mapping between the Application API and the Java wrapper are as follows:

- ***Classes and class names***
  For every interface of the XMLSpy 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 `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

This section
This section explains how some basic XMLSpy functionality can be accessed from Java code. It is organized into the following sub-sections:

- Example Java Project
- Application Startup and Shutdown
- Simple Document Access
- Iterations
- Use of Out-Parameters
- Event Handlers

Example Java Project

The AuthenticDesktop installation package contains an example Java project, located in the Java folder of the API Examples folder:

Windows XP	`C:/Documents and Settings/<username>/My Documents/Altova/Authentic20XX/AuthenticExamples/API/`
Windows Vista, Windows 7	`C:/Users/<username>/Documents/Altova/Authentic20XX/AuthenticExamples/API/`

This folder contains Java examples for the AuthenticDesktop 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:

`AltovaAutomation.dll`	Java-COM bridge: DLL part
`AltovaAutomation.jar`	Java-COM bridge: Java library part
`AuthenticAPI.jar`	Java classes of the Authentic Desktop API
`RunAuthenticDesktop.java`	Java example source code
`BuildAndRun.bat`	Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
`.classpath`	Eclipse project helper file
`.project`	Eclipse project file
`Authentic_JavaDoc.zip`	`Javadoc` file containing help documentation for the Java API

What the example does
The example starts up AuthenticDesktop and performs a few operations, including opening and closing documents. When done, AuthenticDesktop stays open. You must close it manually.

- Start Authentic Desktop: Starts Authentic Desktop, which is registered as an automation server, or activates Authentic Desktop if it is already running.
- Open example files: Locates example documents installed with Authentic Desktop and opens them.
- Iteration and Changing the View Mode: Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- Iteration, validation, output parameters: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- Event Handling: Shows how to handle Authentic Desktop events.
- Shut down Authentic Desktop: Shuts down Authentic Desktop.

You can modify the example in any way you like and run it.

Running the example from the command line
To run the example from the command line, open a command prompt window, go to the Java folder of the API Examples folder (*see above for location*), and then type:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

The Java binary folder must be that of a JDK 1.5 or later installation on your computer.

Press the **Return** key. The Java source in `RunAuthenticDesktop.java` will be compiled and then executed.

Loading the example in Eclipse

Open Eclipse and use the **Import | Existing Projects into Workspace** command to add the Eclipse project file (. `project`) located in the Java folder of the API Examples folder (*see above for location*). The project `RunAuthenticDesktop` will then appear in your Package Explorer or Navigator.

Select the project and then the command **Run as | Java Application** to execute the example.

Note: You can select a class name or method of the Java API and press F1 to get help for that class or method.

Java source code listing

The Java source code in the example file `RunAuthenticDesktop.java` is listed below with comments.

```
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       // 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 }
```

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 }
```

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);
```

Iterations

The listing below shows how to iterate through open documents.

```
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   }
```

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 }
```

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;
```

3.1.3 The DOM and XMLData

The XMLData interface gives you full access to the XML structure behind the current document with less methods than DOM and is much simpler. The XMLData interface is a minimalist approach to reading and modifying existing, or newly created XML data. You might however, want to use a DOM tree because you can access one from an external source or you just prefer the MSXML DOM implementation.

The ProcessDOMNode() and ProcessXMLDataNode() functions provided below convert any segments of an XML structure between XMLData and DOM.

To use the ProcessDOMNode() function:

- pass the root element of the DOM segment you want to convert in objNode and
- pass the plugin object with the CreateChild() method in objCreator

To use the ProcessXMLDataNode() function:

- pass the root element of the XMLData segment in objXMLData and
- pass the DOMDocument object created with MSXML in xmlDoc

```
////////////////////////////////////////////////////////////////
// DOM To XMLData conversion
Function ProcessDOMNode(objNode,objCreator)
{
 var objRoot;
 objRoot = CreateXMLDataFromDOMNode(objNode,objCreator);

 If(objRoot) {
  If((objNode.nodeValue != Null) && (objNode.nodeValue.length > 0))
   objRoot.TextValue = objNode.nodeValue;
   // add attributes
  If(objNode.attributes) {
   var Attribute;
   var oNodeList = objNode.attributes;

   For(var i = 0;i < oNodeList.length; i++) {
    Attribute = oNodeList.item(i);

    var newNode;
    newNode = ProcessDOMNode(Attribute,objCreator);

    objRoot.AppendChild(newNode);
   }
  }
  If(objNode.hasChildNodes){
   try {
    // add children
    var Item;
    oNodeList = objNode.childNodes;

    For(var i = 0;i < oNodeList.length; i++) {
       Item = oNodeList.item(i);

     var newNode;
     newNode = ProcessDOMNode(Item,objCreator);

     objRoot.AppendChild(newNode);
    }
   }
   catch(err) {
   }
  }
 }
```

```
 Return objRoot;
}

Function CreateXMLDataFromDOMNode( objNode, objCreator)
{
 var bSetName = True;
 var bSetValue = True;

 var nKind = 4;

 switch( objNode. nodeType){
  Case 2: nKind = 5; break;
  Case 3: nKind = 6; bSetName = False; break;
  Case 4: nKind = 7; bSetName = False; break;
  Case 8: nKind = 8; bSetName = False; break;
  Case 7: nKind = 9; break;
 }
 var objNew = Null;
 objNew = objCreator. CreateChild( nKind);

 If( bSetName)
  objNew. Name = objNode. nodeName;

 If( bSetValue && ( objNode. nodeValue != Null))
  objNew. TextValue = objNode. nodeValue;

 Return objNew;
}
////////////////////////////////////////////////////////////
// XMLData To DOM conversion

Function ProcessXMLDataNode( objXMLData, xmlDoc)
{
 var objRoot;
 objRoot = CreateDOMNodeFromXMLData( objXMLData, xmlDoc);

 If( objRoot) {
  If( IsTextNodeEnabled( objRoot) && ( objXMLData. TextValue. length > 0))
   objRoot. appendChild( xmlDoc. createTextNode( objXMLData. TextValue));

  If( objXMLData. HasChildren){
   try {
    var objChild;
    objChild = objXMLData. GetFirstChild( -1);

    While( True) {
     If( objChild) {
      var newNode;
      newNode = ProcessXMLDataNode( objChild, xmlDoc);

      If( newNode. nodeType == 2){
       // child node is an attribute
       objRoot. attributes. setNamedItem( newNode);
      }
      Else
       objRoot. appendChild( newNode);
     }
     objChild = objXMLData. GetNextChild();
    }
   }
   catch( err) {
   }
  }
 }
 Return objRoot;
}
```

```
Function CreateDOMNodeFromXMLData(objXMLData,xmlDoc)
{
 switch(objXMLData.Kind){
  Case 4:Return xmlDoc.createElement(objXMLData.Name);
  Case 5:Return xmlDoc.createAttribute(objXMLData.Name);
  Case 6:Return xmlDoc.createTextNode(objXMLData.TextValue);
  Case 7:Return xmlDoc.createCDATASection(objXMLData.TextValue);
  Case 8:Return xmlDoc.createComment(objXMLData.TextValue);
  Case 9:Return
xmlDoc.createProcessingInstruction(objXMLData.Name,objXMLData.TextValue);
 }

 Return xmlDoc.createElement(objXMLData.Name);
}
Function IsTextNodeEnabled(objNode)
{
 switch(objNode.nodeType) {
  Case 1:
  Case 2:
  Case 5:
  Case 6:
  Case 11:Return True;
 }
 Return False;
}
```

3.1.4 Obsolete: Authentic View Row operations

If the schema on which an XML document is based specifies that an element is repeatable, such a structure can be represented in Authentic View as a table. When represented as a table, rows and their contents can be manipulated individually, thereby allowing you to manipulate each of the repeatable elements individually. Such row operations would be performed by an external script.

If an external script is to perform row operations then two steps must occur:

- The first step checks whether the cursor is currently in a row using a property. Such a check could be, for example, `IsRowInsertEnabled`, which returns a value of either `TRUE` or `FALSE`.
- If the return value is `TRUE` then a row method, such as `RowAppend`, can be called. (`RowAppend` has no parameters and returns no value.)

The following is a list of properties and methods available for table operations. Each property returns a `BOOL`, and the methods have no parameter.

Property	Method	Table operations
`IsRowInsertEnabled`	**RowInsert**, superseded by `AuthenticRange.InsertRow`	Insert row operation
`IsRowAppendEnabled`	**RowAppend**, superseded by `AuthenticRange.AppendRow`	Append row operation
`IsRowDeleteEnabled`	**RowDelete**, superseded by `AuthenticRange.DeleteRow`	Delete row operation
`IsRowMoveUpEnabled`	**RowMoveUp**, superseded by `AuthenticRange.MoveRowUp`	Move XML data up one row
`IsRowMoveDownEnabled`	**RowMoveDown**, superseded by `AuthenticRange.MoveRowDown`	Move XML data down one row
`IsRowDuplicateEnabled`	**RowDuplicate**, superseded by `AuthenticRange.DuplicateRow`	Duplicate currently selected row

3.2 Interfaces

Object Hierarchy

Application
- SpyProject
  - SpyProjectItems
    - SpyProjectItem
- Documents
  - Document
    - GridView
    - AuthenticView
      - AuthenticRange
      - AuthenticDataTransfer (previously DocEditDataTransfer)
    - OldAuthenticView (previously DocEditView, **now obsolete**, superseded by AuthenticView and AuthenticRange)
      - AuthenticSelection (previously DocEditSelection, **now obsolete**, superseded by AuthenticRange)
      - AuthenticEvent (previously DocEditEvent, **now obsolete**)
      - AuthenticDataTransfer (previously DocEditDataTransfer)
    - TextView
    - XMLData
- Dialogs
  - CodeGeneratorDlg
  - FileSelectionDlg
  - SchemaDocumentationDlg
  - GenerateSampleXMLDlg
  - DTDSchemaGeneratorDlg
  - FindInFilesDlg
  - WSDLDocumentationDlg
  - WSDL20DocumentationDlg
  - XBRLDocumentationDlg
- DatabaseConnection
- ExportSettings
- TextImportExportSettings
- ElementList
  - ElementListItem

Enumerations

Description

This chapter contains the reference of the Authentic Desktop 1.5 Type Library.

Most of the given examples are written in VisualBasic. These code snippets assume that there is a variable defined and set, called **objSpy of type Application**. There are also some code samples written in JavaScript.

3.2.1 Application

See also

Methods

GetDatabaseImportElementList
GetDatabaseSettings
GetDatabaseTables
ImportFromDatabase
CreateXMLSchemaFromDBStructure

GetTextImportElementList
GetTextImportExportSettings
ImportFromText

ImportFromWord

ImportFromSchema

GetExportSettings

NewProject
OpenProject

AddMacroMenuItem
ClearMacroMenu

ShowForm

ShowApplication

URLDelete
URLMakeDirectory

FindFiles

Quit

Properties

Application
Parent

ActiveDocument
Documents

CurrentProject

Dialogs

WarningNumber
WarningText

Status
MajorVersion
MinorVersion
Edition
IsAPISupported
ServicePackVersion

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")
```

Events

OnBeforeOpenDocument

See also

Event: OnBeforeOpenDocument(*objDialog* as FileSelectionDlg)

Description
This event gets fired whenever a document gets opened via the OpenFile or OpenURL menu command. It is sent after a document file has been selected but before the document gets opened. The file selection dialog object is initialized with the name of the selected document file. You can modify this selection. To continue the opening of the document leave the FileSelectionDlg.DialogAction property of *io_objDialog* at its default value spyDialogOK. To abort the opening of the document set this property to spyDialogCancel.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_BeforeOpenDocument(objDialog)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeOpenDocument(objDialog)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (26, ...)  //nEventID=26
```

OnBeforeOpenProject

See also

Event: OnBeforeOpenProject(*objDialog* as FileSelectionDlg)

Description
This event gets fired after a project file has been selected but before the project gets opened. The file selection dialog object is initialized with the name of the selected project file. You can modify this selection. To continue the opening of the project leave the FileSelectionDlg.DialogAction property of *io_objDialog* at its default value spyDialogOK. To abort the opening of the project set this property to spyDialogCancel.

Examples

Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_BeforeOpenProject(objDialog)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeOpenProject(objDialog)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (25, ...)  //nEventID=25
```

OnDocumentOpened

See also

Event: `OnDocumentOpened(objDocument as Document)`

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
```

OnProjectOpened

See also

Event: `OnProjectOpened(objProject as SpyProject)`

Description
This event gets fired whenever a project gets opened in Authentic Desktop. The new project gets passed as parameter.

Examples

Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_OpenProject(objProject)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_OpenProject(objProject)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (6, ...)    //nEventID=6
```

ActiveDocument

See also

Property: ActiveDocument as Document

Description
Reference to the active document. If no document is open, ActiveDocument is null (nothing).

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

AddMacroMenuItem

See also

Method: AddMacroMenuItem(*strMacro* as String,*strDisplayText* as String)

Description
Adds a menu item to the **Tools** menu. This new menu item invokes the macro defined by strMacro. See also "Calling macros from Authentic Desktop".

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1108	Number of macro items is limited to 16 items.

Application

See also

Property: Application as Application (read-only)

Description
Accesses the Authentic Desktop application object.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ClearMacroMenu

See also

Method: `ClearMacroMenu()`

Return Value
None

Description
Removes all menu items from the **Tools** menu. See also Calling macros from Authentic Desktop".

Errors
1111 The application object is no longer valid.

CreateXMLSchemaFromDBStructure

See also

Method: `CreateXMLSchemaFromDBStructure(pImportSettings as DatabaseConnection, pTables as ElementList)`

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.

Errors
1112 Invalid database specified.
1120 Database import failed.

CurrentProject

See also

Property: `CurrentProject as SpyProject`

Description
Reference to the active document. If no project is open, `CurrentProject` is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Dialogs

See also

Property: `Dialogs` as `Dialogs` (read-only)

Description
Access the built-in dialogs of Authentic Desktop.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Documents

See also

Property: `Documents` as `Documents`

Description
Collection of all open documents. See also Simple document access.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Edition

See also

Property: `Edition` as String

Description
Returns the edition of the application. Eg: Enterprise, Professional, Basic

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

FindInFiles

See also

Method: FindInFiles(*pSettings* as FindInFilesDlg) as FindInFilesResults

Description
Returns a FindInFilesResults object containing information about the files that matched the specified settings.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetDatabaseImportElementList

See also

Method: GetDatabaseImportElementList(*pImportSettings* as DatabaseConnection) as ElementList

Description
The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the fields that can be selected for import and the properties ElementListItem.ElementKind are initialized either to *spyXMLDataAttr* or *spyXMLDataElement*, depending on the value passed in DatabaseConnection.AsAttributes. This list serves as a filter to what finally gets imported by a future call to ImportFromDatabase. Use ElementList.RemoveElement to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File, DatabaseConnection.ADOConnection and DatabaseConnection.ODBCConnection, as well as DatabaseConnection.SQLSelect. Use the property DatabaseConnection.AsAttributes to initialize ElementListItem.ElementKind of the resulting element list to either *spyXMLDataAttr* or *spyXMLDataElement*, respectively.

Example
See example at ImportFromDatabase.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1112	Invalid database specified.
1114	Select statement is missing.
1119	database element list import failed.

GetDatabaseSettings

See also

Method: GetDatabaseSettings() as DatabaseConnection

Description
GetDatabaseSettings creates a new object of database settings. The object is used to specify database connection parameters for the methods GetDatabaseTables, GetDatabaseImportElementList, ImportFromDatabase, ImportFromSchema and ExportToDatabase.

Example
See example of ImportFromDatabase.

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

GetDatabaseTables

See also

Method: GetDatabaseTables(*pImportSettings* as DatabaseConnection) as ElementList

Description

GetDatabaseTables reads the table names from the database specified in *pImportSettings*. Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File, DatabaseConnection.ADOConnection and DatabaseConnection.ODBCConnection. All other properties are ignored. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of tables stored in the specified database. The remaining properties of ElementListItem are unused.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1112 Invalid database specified.
1113 Error while reading database table information.
1118 Database table query failed.

Example

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
objImpSettings.ADOConnection = TxtADO.Text

'store table names in list box
ListTables.Clear

Dim objList As ElementList
Dim objItem As ElementListItem
On Error GoTo ErrorHandler
Set objList = objSpy.GetDatabaseTables(objImpSettings)

For Each objItem In objList
 ListTables.AddItem objItem.Name
Next
```

GetExportSettings

See also

Method: GetExportSettings() as ExportSettings (read-only)

Description

GetExportSettings creates a new object of common export settings. This object is used to pass the parameters to the export functions and defines the behaviour of the export calls. See also the export functions from Document and the examples at Import and Export.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetTextImportElementList

See also

Method: GetTextImportElementList(pImportSettings as TextImportExportSettings) as ElementList

Description

GetTextImportElementList retrieves importing information about the text-file as specified in pImportSettings. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the fields found in the file. The values of remaining properties are undefined.

If the text-file does not contain a column header, set pImportSettings.HeaderRow to false. The resulting element list will contain general column names like 'Field1' and so on.

See also Import and export of data.

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
```

GetTextImportExportSettings

See also

Method: GetTextImportExportSettings() as TextImportExportSettings (read-only)

Description

GetTextImportExportSettings creates a new object of common import and export settings for text files. See also the example for Application.GetTextImportElementList and Import and Export.

See also `Import and export of data.`

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ImportFromDatabase

See also

Method: `ImportFromDatabase(pImportSettings as DatabaseConnection pElementList as ElementList) as Document`

Return Value

Creates a new document containing the data imported from the database.

Description

`ImportFromDatabase` imports data from a database as specified in `pImportSettings` and creates a new document containing the data imported from the database. Properties mandatory to be filled out are one of `DatabaseConnection.File`, `DatabaseConnection.ADOConnection` or `DatabaseConnection.ODBCConnection` and `DatabaseConnection.SQLSelect`. Additionally, you can use `DatabaseConnection.AsAttributes`, `DatabaseConnection.ExcludeKeys`, `DatabaseConnection.IncludeEmptyElements` and `NumberDateTimeFormat` to further parameterize import.

The parameter `pElementList` specifies which fields of the selected data gets written into the newly created document, and which are created as elements and which as attributes. This parameter can be NULL, specifying that all selected fields will be imported as XML elements.

See `GetDatabaseSettings` and `GetDatabaseImportElementList` for necessary steps preceding any import of data from a database.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1117 Transformation to XML failed.
1120 Database import failed.

Example

```
 Dim objImpSettings As DatabaseConnection
 Set objImpSettings = objSpy.GetDatabaseSettings

 objImpSettings.ADOConnection = strADOConnection
 objImpSettings.SQLSelect = "SELECT * FROM MyTable"

 Dim objDoc As Document
 On Error Resume Next
 Set objDoc = objSpy.ImportFromDatabase(objImpSettings,
objSpy.GetDatabaseImportElementList(objImpSettings))
 ' CheckForError here
```

ImportFromSchema

See also

Method: ImportFromSchema(*pImportSettings* as DatabaseConnection,*strTable* as String,*pSchemaDoc* as Document) as Document

Return Value
Creates a new document filled with data from the specified database as specified by the schema definition in *pSchemaDoc*.

Description
ImportFromSchema imports data from a database specified in pImportSettings. Properties mandatory to be filled out are one of DatabaseConnection.File, DatabaseConnection.ADOConnection or DatabaseConnection.ODBCConnection. Additionally, you can use DatabaseConnection.AsAttributes, DatabaseConnection.ExcludeKeys and NumberDateTimeFormat to further parameterize import. All other properties get ignored.

ImportFromSchema does not use and explicit SQL statement to select the data. Instead, it expects a structure definition of the document to create in form of an XML schema document in *pSchemaDoc*. From this definition the database select statement is automatically deduced. Specify in *strTable* the table name of the import root that will become the root node in the new document.

See GetDatabaseSettings and GetDatabaseTables for necessary steps preceding an import from a database based on a schema definition. To create the schema definition file use command 'create database schema' from the 'convert' menu of Authentic Desktop.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1112	Invalid database specified.
1120	Database import failed.
1121	Could not create validator for the specified schema.
1122	Failed parsing schema for database import.

ImportFromText

See also

Method: ImportFromText(*pImportSettings* as TextImportExportSettings, *pElementList* as ElementList) as Document

Description
ImportFromText imports the text file as specified in pImportSettings. The parameter pElementList can be used as import filter. Either pass the list returned by a previous call to GetTextImportElementList or nul to import all columns. To avoid import of unnecessary columns use ElementList.RemoveElement to remove the corresponding field names from pElementList before calling ImportFromText.
The method returns the newly created document containing the imported data. This document is the same as the active document of Authentic Desktop.

See also `Import and export of data`.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from text file failed.
1117 Transformation to XML failed.

Example

```
' ----------------------------------------------------------
' VBA client code fragment - import from text file
' ----------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = strFileName
objImpSettings.HeaderRow = False

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings,
  objSpy.GetTextImportElementList(objImpSettings))

CheckForError
```

ImportFromWord

See also

Method: `ImportFromWord(`*`strFile`* as String) as `Document`

Description

`ImportFromWord` imports the MS-Word Document `strFile` into a new XML document.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
Import from document failed.

IsAPISupported

See also

Property: `IsAPISupported` as Boolean

Description

Returns whether the API is supported in this version or not.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

MajorVersion

See also

Property: MajorVersion as Integer

Description
Returns the application version's major number.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

MinorVersion

See also

Property: MinorVersion as Integer

Description
Returns the application version's minor number.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

NewProject

See also

Method: NewProject(*strPath* as String,*bDiscardCurrent* as Boolean)

Description
NewProject creates a new project.

If there is already a project open that has been modified and bDiscardCurrent is false, then NewProject() fails.

Errors
1111 The application object is no longer valid.
1102 A project is already open but *bDiscardCurrent* is *true*.
1103 Creation of new project failed.

OpenProject

See also

Method: OpenProject(*strPath* as String,*bDiscardCurrent* as Boolean,*bDialog* as Boolean)

Parameters
strPath
Path and file name of the project to open. Can be empty if bDialog is true.

bDiscardCurrent
Discard currently open project and possible lose changes.

bDialog
Show dialogs for user input.

Return Value
None

Description
`OpenProject` opens an existing project. If there is already a project open that has been modified and `bDiscardCurrent` is false, then `OpenProject()` fails.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1101 Cannot open specified project.
1102 A project is already open but *bDiscardCurrent* is *true*.

Parent

See also

Property: `Parent` as `Application` (read-only)

Description
Accesses the Authentic Desktop application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Quit

See also

Method: Quit()

Return Value
None

Description
This method terminates Authentic Desktop. All modified documents will be closed without saving the changes. This is also true for an open project.

If Authentic Desktop was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any document is still open. Use the Quit method to ensure automatic shut-down.

Errors
1111 The application object is no longer valid.

ReloadSettings

See also

Method: `ReloadSettings`

Return Value

Description
The application settings are reloaded from the registry.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

RunMacro

See also

Method: `RunMacro(strMacro as String)`

Return Value

Description
Calls the specified macro either from the project scripts (if present) or from the global scripts.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

ScriptingEnvironment

See also

Property: `ScriptingEnvironment` as IUnknown (read-only)

Description
Reference to any active scripting environment. This property makes it possible to access the TypeLibrary of the XMLSpyFormEditor.exe application which is used as the current scripting environment.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ServicePackVersion

See also

Property: `ServicePackVersion` as Long

Description
Returns the Service Pack version number of the application. Eg: 1 for 2010 R2 SP1

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ShowApplication

See also

Method: ShowApplication(*bShow* as Boolean)

Return Value
None

Description
The method shows (bShow = True) or hides (bShow = False) Authentic Desktop.

Errors
1110 The application object is no longer valid.

ShowFindInFiles

See also

Method: ShowFindInFiles(pSettings as FindInFilesDlg) as Boolean

Return Value
Returns false if the user pressed the Cancel button, true otherwise.

Description
Displays the FindInFiles dialog preset with the given settings. The user modifications of the settings are stored in the passed dialog object.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

ShowForm

See also

Method: ShowForm(*strFormName* as String) as Long

Return Value
Returns zero if the user pressed a Cancel button or the form calls TheView.Cancel() .

Description
Displays the form strFormName .

Forms, event handlers and macros can be created with the Scripting Environment. Select "Switch to scripting environment" from the **Tools** menu to invoke the Scripting Environment.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

Status

See also

Property: Status as ENUMApplicationStatus

Description
Returns the current status of the running application.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

URLDelete

See also

Method: URLDelete(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method deletes the file at the URL strURL .

Errors
1111 The application object is no longer valid.
1109 Error deleting file at specified URL.

URLMakeDirectory

See also

Method: URLMakeDirectory(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method creates a new directory at the URL strURL .

Errors
1111 The application object is no longer valid.
1100 Invalid parameter specified.

Visible

See also

Property: Visible as VARIANT_BOOL

Description
Sets or gets the visibility attribute of Authentic Desktop. This standard automation property makes

usage of ShowApplication obsolete.

Errors
1110 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningNumber

See also

Property: `WarningNumber` as integer

Description
Some methods fill the property `WarningNumber` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningText

See also

Property: `WarningText` as String

Description
Some methods fill the property `WarningText` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

3.2.2 AuthenticContextMenu

The context menu interface provides the mean for the user to customize the context menus shown in Authentic. The interface has the methods listed in this section.

CountItems

Method: `CountItems()` nItems as long

Return Value
Returns the number of menu items.

Errors
2501 Invalid object.

DeleteItem

Method: `DeleteItem( IndexPosition as long)`

Return Value
Deletes the menu item that has the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

GetItemText

Method: `GetItemText( IndexPosition as long)` MenuItemName as string

Return Value
Gets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

InsertItem

Method: `InsertItem( IndexPosition as long, MenuItemName as string, MacroName as string)`

Return Value
Inserts a user-defined menu item at the position in the menu specified in the first parameter and having the name submitted in the second parameter. The menu item will start a macro, so a valid macro name must be submitted.

Errors
2501 Invalid object
2502 Invalid index
2503 No such macro
2504 Internal error

SetItemText

Method: `SetItemText( IndexPosition as long, MenuItemName as string)`

Return Value
Sets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

3.2.3 AuthenticDataTransfer

Renamed from DocEditDataTransfer to AuthenticDataTransfer

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage—except for AuthenticDataTransfer—is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

getData

Properties

dropEffect
ownDrag
type

Description

The events OnDragOver and OnBeforeDrop provide information about the object being dragged with an instance of type AuthenticDataTransfer. It contains a description of the dragged object and its content. The latter is available either as string or a pointer to a COM object supporting the IUnkown interface.

dropEffect

See also

Property: dropEffect as long

Description
The property stores the drop effect from the default event handler. You can set the drop effect if you change this value and return TRUE for the event handler (or set AuthenticEvent.cancelBubble to TRUE if you are still using the now obsolete AuthenticEvent interface).

Errors
2101 Invalid address for the return parameter was specified.

getData

See also

Method: `getData()` as Variant

Description
Retrieve the data associated with the dragged object. Depending on `AuthenticDataTransfer.type`, that data is either a string or a COM interface pointer of type `IUnknown`.

Errors
2101 Invalid address for the return parameter was specified.

ownDrag

See also

Property: `ownDrag` as Boolean (read-only)

Description
The property is `TRUE` if the current dragging source comes from inside Authentic View.

Errors
2101 Invalid address for the return parameter was specified.

type

See also

Property: `type` as String (read-only)

Description
Holds the type of data you get with the `DocEditDataTransfer.getData` method.

Currently supported data types are:

OWN	data from Authentic View itself
TEXT	plain text
UNICODETEXT	plain text as UNICODE

Errors
2101 Invalid address for the return parameter was specified.

3.2.4 AuthenticEventContext

The `EventContext` interface gives access to many properties of the context in which a macro is executed.

EvaluateXPath

Method: `EvaluateXPath` (`strExpression as string`) as strValue as string

Return Value
The method evaluates the XPath expression in the context of the node within which the event was triggered and returns a string.

Description
`EvaluateXPath()` executes an XPath expressions with the given event context. The result is returned as string, in the case of a sequence it is a space-separated string.

Errors
2201 Invalid object.
2202 No context.
2209 Invalid parameter.
2210 Internal error.
2211 XPath error.

GetEventContextType

Method: `GetEventContextType()` Type as `AuthenticEventContextType` enumeration

Return Value
Returns the context node type.

Description
`GetEventContextType` allows the user to determine whether the macro is in an XML node or in an XPath atomic item context. The enumeration `AuthenticEventContextType` is defined as follows:

```
authenticEventContextXML,
authenticEventContextAtomicItem,
authenticEventContextOther
```

If the context is a normal XML node, the `GetXMLNode()` function gives access to it (returns `NULL` if not).

Errors
2201 Invalid object.
2202 No context.
2209 Invalid parameter.

GetNormalizedTextValue

Method: `GetNormalizedTextValue()` strValue as string

Return Value
Returns the value of the current node as string

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

GetVariableValue

Method: `GetVariableValue( strName as string) strValue as string`

Return Value
Gets the value of the variable submitted as the parameter.

Description
`GetVariableValue` gets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
        AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

GetXMLNode

Method: `GetXMLNode()` Node as XMLData object

Return Value
Returns the context XML node or `NULL`

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

IsAvailable

Method: `IsAvailable()` as Boolean

Return Value
Returns true if `EventContext` is set, false otherwise.

Errors
2201 Invalid object.

SetVariableValue

Method: `SetVariableValue( strName as string, strValue as string)`

Return Value

Sets the value (second parameter) of the variable submitted in the first parameter.

Description

`SetVariableValue` sets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
        AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors

2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

3.2.5 AuthenticRange

See also

The first table lists the properties and methods of AuthenticRange that can be used to navigate through the document and select specific portions.

Properties

Application
FirstTextPosition
FirstXMLData
FirstXMLDataOffset
LastTextPosition
LastXMLData
LastXMLDataOffset
Parent

Clone
CollapsToBegin
CollapsToEnd
ExpandTo
Goto
GotoNext
GotoPrevious
IsEmpty
IsEqual

Methods

MoveBegin
MoveEnd
NextCursorPosition
PreviousCursorPosition
Select
SelectNext
SelectPrevious
SetFromRange

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties

Text

Edit operations

Copy
Cut
Delete
IsCopyEnabled
IsCutEnabled
IsDeleteEnabled
IsPasteEnabled
Paste

Dynamic table operations

AppendRow
DeleteRow
DuplicateRow
InsertRow
IsFirstRow
IsInDynamicTable
IsLastRow
MoveRowDown
MoveRowUp

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows

Elements

CanPerformActionWith
CanPerformAction
PerformAction

Attributes

GetElementAttributeValue
GetElementAttributeNames
GetElementHierarchy
HasElementAttribute
IsTextStateApplied
SetElementAttributeValue

Entities

GetEntityNames
InsertEntity

Description

AuthenticRange objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for AuthenticRange objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of AuthenticRange objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use AuthenticView.Selection, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial

ranges are accessible via the read/only properties AuthenticView.DocumentBegin, AuthenticView.DocumentEnd, and AuthenticView.WholeDocument. The most flexible method is AuthenticView.Goto, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above, with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see FirstTextPosition and LastTextPosition. This method requires complex internal calculations and should be used with care.
- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see FirstXMLData, FirstXMLDataOffset, LastXMLData, and LastXMLDataOffset. This method is very efficient but requires knowledge on the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The Text property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions Cut, Copy, Paste and Delete.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the XMLData objects of the underlying document to modify them directly.

AppendRow

See also

Method: AppendRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns *true* if the append operation was successful, otherwise *false*.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Append row at end of current dynamically growable table
' ------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
     objRange.AppendRow
     ' objRange points to beginning of new row
     objRange.Select
```

```
End If
```

Application

See also

Property: Application as Application (read-only)

Description
Accesses the Authentic Desktop application object.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CanPerformAction

See also

Method: CanPerformAction (*eAction* as SPYAuthenticActions, *strElementName* as String) as Boolean

Description
CanPerformAction and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns *True*, otherwise it returns *False*.

HINT: To find out all valid element names for a given action, use CanPerformActionWith.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction.

CanPerformActionWith

See also

Method: CanPerformActionWith (*eAction* as SPYAuthenticActions, *out_arrElementNames* as Variant)

Description
PerformActionWith and its related methods, enable access to the entry-helper functions of Authentic. These function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use CanPerformActionWith.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See `PerformAction`.

Clone

See also

Method: `Clone()` as `AuthenticRange`

Description
Returns a copy of the range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CollapsToBegin

See also

Method: `CollapsToBegin()` as `AuthenticRange`

Description
Sets the end of the range object to its begin. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CollapsToEnd

See also

Method: `CollapsToEnd()` as `AuthenticRange`

Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Copy

See also

Method: `Copy()` as Boolean

Description
Returns *False* if the range contains no portions of the document that may be copied.
Returns *True* if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/paste buffer.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Cut

See also

Method: `Cut()` as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/paste buffer.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Delete

See also

Method: `Delete()` as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DeleteRow

See also

Method: `DeleteRow()` as Boolean

Description
If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns *true,* if the delete operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------------------------
' XMLSpy 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
```

DuplicateRow

See also

Method: `DuplicateRow()` as Boolean

Description

If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns *true* if the duplicate operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------------------------
' XMLSpy 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
```

EvaluateXPath

Method: `EvaluateXPath` (strExpression as string) strValue as string

Return Value

The method returns a string

Description

`EvaluateXPath()` executes an XPath expressions with the context node being the beginning of the range selection. The result is returned as string, in the case of a sequence it is a space-separated string. If XML context node is irrelevant, the user may provide any node, like `AuthenticView.XMLDataRoot`.

Errors

2001 Invalid object
2005 Invalid parameter
2008 Internal error
2202 Missing context node
2211 XPath error

ExpandTo

See also

Method: `ExpandTo` (*eKind* as SPYAuthenticElementKind), as AuthenticRange

Description

Selects the whole element of type `eKind`, that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Range expansion would be beyond end of document.
2005 Invalid address for the return parameter was specified.

FirstTextPosition

See also

Property: `FirstTextPosition` as Long

Description

Set or get the left-most text position index of the range object. This index is always less or equal to LastTextPosition. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect as the cursor-left key.

If you set FirstTextPosition to a value greater than the current LastTextPosition, LastTextPosition gets set to the new FirstTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to `XMLData` based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
' ----------------------------------------
' XMLSpy 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
```

FirstXMLData

See also

Property: `FirstXMLData` as `XMLData`

Description

Set or get the first `XMLData` element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the `FirstXMLDataOffset` attribute.

Whenever you set `FirstXMLData` to a new data object, `FirstXMLDataOffset` gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set `FirstXMLData` / `FirstXMLDataOffset` selects a position greater then the current `LastXMLData` / `LastXMLDataOffset`, the latter gets moved to the new start position.

HINT: You can use the `FirstXMLData` and `LastXMLData` properties, to directly access and manipulate the underlying XML document in those cases where the methods available with the `AuthenticRange` object are not sufficient.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2008	Internal error
2009	The XMLData object cannot be accessed.

Examples

```
  ' ------------------------------------------------
  ' XMLSpy scripting environment - VBScript
  ' show name of currently selected XMLData element
  ' ------------------------------------------------
  Dim objAuthenticView
  ' we assume that the active document is open in authentic view mode
  Set objAuthenticView = Application.ActiveDocument.AuthenticView

  Dim objXmlData
  Set objXMLData = objAuthenticView.Selection.FirstXMLData
  ' authentic view adds a 'text' child element to elements
  ' of the document which have content.  So we have to go one
  ' element up.
  Set objXMLData = objXMLData.Parent
  MsgBox "Current selection selects element " & objXMLData.Name
```

FirstXMLDataOffset

See also

Property: FirstXMLDataOffset as Long

Description
Set or get the cursor position offset inside FirstXMLData element for the beginning of the range. Offset positions are based on the characters returned by the Text property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If FirstXMLData / FirstXMLDataOffset selects a position after the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid offset was specified. Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' XMLSpy scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
     objRange.Select()
Else
     MsgBox "Oops"
End If
```

GetElementAttributeNames

See also

Method: GetElementAttributeNames (*strElementName* as String, *out_arrAttributeNames* as Variant)

Description
Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods GetElementAttributeValue and SetElementAttributeValue.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid element name was specified.
	Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetElementAttributeValue

See also

Method: GetElementAttributeValue (*strElementName* as String, *strAttributeName* as String) as String

Description
Retrieve the value of the attribute specified in strAttributeName, for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid element name was specified.
	Invalid attribute name was specified.
	Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetElementHierarchy

See also

Method: GetElementHierarchy (*out_arrElementNames* as Variant)

Description
Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single XMLData element.

The names of the element hierarchy, together with the range object uniquely identify XMLData

elements in the document. The attributes of these elements can be directly accessed by GetElementAttributeNames, and related methods.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

C# Examples

```
' ------------------------------------------
' C#
' ------------------------------------------

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            XMLSpyLib.Application app = new XMLSpyLib.Application();

            app.ShowApplication(true);

            XMLSpyLib.AuthenticView view = app.ActiveDocument.AuthenticView;
            XMLSpyLib.AuthenticRange range = view.DocumentBegin;

            object o = null;
            range.GetElementHierarchy(ref o);

            object[] elements = (object[])o;

            foreach (string e in elements)
            {
                Console.WriteLine(e);
            }
        }
    }
}
```

Also see: SetElementAttributeValue.

GetEntityNames

See also

Method: GetEntityNames (*out_arrEntityNames* as Variant)

Description

Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the InsertEntity function.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

See: GetElementHierarchy and InsertEntity.

GetVariableValue

Method: GetVariableValue(strName as string) strVal as string

Return Value
Gets the value of the variable named as the method's parameter.

Errors

2001	Invalid object.
2202	No context.
2204	No such variable in scope
2205	Variable cannot be evaluated
2206	Variable returns sequence
2209	Invalid parameter

Goto

See also

Method: Goto (*eKind* as SPYAuthenticElementKind, *nCount* as Long, *eFrom* as SPYAuthenticDocumentPosition) as AuthenticRange

Description
Sets the range to point to the beginning of the nCount element of type eKind . The start position is defined by the parameter eFrom .

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified. Invalid start position specified. Invalid address for the return parameter was specified.

GotoNext

See also

Method: GotoNext (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Sets the range to the beginning of the next element of type eKind . The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy 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
```

GotoNextCursorPosition

See also

Method: `GotoNextCursorPosition()` as `AuthenticRange`

Description
Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid address for the return parameter was specified.

GotoPrevious

See also

Method: `GotoPrevious` (*`eKind`* as `SPYAuthenticElementKind`) as `AuthenticRange`

Description
Sets the range to the beginning of the element of type `eKind` which is before the beginning of the current range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy 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 bEndOfDocument
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
```

GotoPreviousCursorPosition

See also

Method: `GotoPreviousCursorPosition()` as AuthenticRange

Description
Set the range to the cursor position immediately before the current position. Returns the modified object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before begin of document.
2005	Invalid address for the return parameter was specified.

HasElementAttribute

See also

Method: `HasElementAttribute` (*`strElementName`* as String, *`strAttributeName`* as String) as Boolean

Description
Tests if the enclosing element with name `strElementName`, supports the attribute specified in `strAttributeName`.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid element name was specified. Invalid address for the return parameter was specified.

InsertEntity

See also

Method: `InsertEntity` (*`strEntityName`* as String)

Description
Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by GetEntityNames.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Unknown entry name was specified.

Examples
```
' ----------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Insert the first entity in the list of available entities
' ----------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we get the names of all available entities as they
' are shown in the entry helper of XMLSpy
Dim arrEntities
objRange.GetEntityNames arrEntities

' we insert the first one of the list
If UBound(arrEntities) >= 0 Then
     objRange.InsertEntity arrEntities(0)
Else
     MsgBox "Sorry, no entities are available for this document"
End If
```

InsertRow

See also

Method: InsertRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row before the current one. The selection of the range, gets modified to point to the beginning of the newly inserted row. The function returns *true* if the insert operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
```
' ----------------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Insert row at beginning of current dynamically growing table
' ----------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
     objRange.InsertRow
     ' objRange points to beginning of new row
     objRange.Select
End If
```


IsCopyEnabled

See also

Property: `IsCopyEnabled` as Boolean (read-only)

Description
Checks if the copy operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsCutEnabled

See also

Property: `IsCutEnabled` as Boolean (read-only)

Description
Checks if the cut operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsDeleteEnabled

See also

Property: `IsDeleteEnabled` as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsEmpty

See also

Method: `IsEmpty()` as Boolean

Description
Tests if the first and last position of the range are equal.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsEqual

See also

Method: `IsEqual` (*`objCmpRange`* as `AuthenticRange`) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors
2001 One of the two range objects being compared, is invalid.
2005 Invalid address for a return parameter was specified.

IsFirstRow

See also

Property: `IsFirstRow` as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsInDynamicTable

See also

Method: `IsInDynamicTable()` as Boolean

Description
Test if the whole range is inside a table that supports the different row operations like 'insert', 'append', duplicate, etc.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsLastRow

See also

Property: `IsLastRow` as Boolean (read-only)

Description
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsPasteEnabled

See also

Property: `IsPasteEnabled` as Boolean (read-only)

Description
Checks if the paste operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsSelected

Property: `IsSelected` as Boolean

Description
Returns true() if selection is present. The selection range still can be empty: that happens when e.g. only the cursor is set.

IsTextStateApplied

See also

Method: `IsTextStateApplied` (*`i_strElementName`* as String) as Boolean

Description
Checks if all the selected text is embedded into an XML Element with name `i_strElementName`. Common examples for the parameter `i_strElementName` are "strong", "bold" or "italic".

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

LastTextPosition

See also

Property: `LastTextPosition` as Long

Description
Set or get the rightmost text position index of the range object. This index is always greater or equal to `FirstTextPosition`. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

If you set LastTextPosition to a value less then the current FirstTextPosition, FirstTextPosition gets set to the new LastTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
' ---------------------------------------
' XMLSpy 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
```

LastXMLData

See also

Property: LastXMLData as XMLData

Description

Set or get the last XMLData element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the LastXMLDataOffset attribute.

Whenever you set LastXMLData to a new data object, LastXMLDataOffset gets set to the last cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set LastXMLData / LastXMLDataOffset, select a position less then the current FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

HINT: You can use the FirstXMLData and LastXMLData properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the AuthenticRange object are not sufficient.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.

LastXMLDataOffset

See also

Property: LastXMLDataOffset as Long

Description

Set or get the cursor position inside LastXMLData element for the end of the range.

Offset positions are based on the characters returned by the Text property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If LastXMLData / LastXMLDataOffset selects a position before FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------
' XMLSpy 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
```

MoveBegin

See also

Method: MoveBegin (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description

Move the beginning of the range to the beginning of the nCount element of type eKind . Counting starts at the current beginning of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

MoveEnd

See also

Method: MoveEnd (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description

Move the end of the range to the begin of the nCount element of type eKind . Counting starts at the current end of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

MoveRowDown

See also

Method: `MoveRowDown()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MoveRowUp

See also

Method: `MoveRowUp()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
See `JScript - Bubble Sort Dynamic Tables`.

Parent

See also

Property: `Parent` as `AuthenticView` (read-only)

Description
Access the view that owns this range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Paste

See also

Method: `Paste()` as Boolean

Description
Returns *False* if the copy/paste buffer is empty, or its content cannot replace the current selection.

Otherwise, deletes the current selection, inserts the content of the copy/paste buffer, and returns *True*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

PerformAction

See also

Method: `PerformAction` (*`eAction`* as `SPYAuthenticActions`, *`strElementName`* as String) as Boolean

Description
`PerformAction` and its related methods, give access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without a need to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If no such location can be found, the method returns *False*. Otherwise, the document gets modified and the range points to the beginning of the modification.

HINT: To find out element names that can be passed as the second parameter use `CanPerformActionWith`.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Insert the innermost element
' ------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' we determine the elements that can be inserted at the current position
Dim arrElements()
objRange.CanPerformActionWith spyAuthenticInsertBefore, arrElements

' we insert the first (innermost) element
If UBound(arrElements) >= 0 Then
     objRange.PerformAction spyAuthenticInsertBefore, arrElements(0)
     ' objRange now points to the beginning of the inserted element
     ' we set a default value and position at its end
     objRange.Text = "Hello"
     objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
     MsgBox "Can't insert any elements at current position"
End If
```


Select

See also

Method: Select()

Description

Makes this range the current user interface selection. You can achieve the same result using: '*objRange.Parent.Selection = objRange*'

Errors

2001 The authentic range object or its related view object is no longer valid.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' set current selection to end of document
objAuthenticView.DocumentEnd.Select()
```

SelectNext

See also

Method: SelectNext (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description

Selects the element of type eKind after the current end of the range. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document word-by-word
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
     objRange.SelectNext(spyAuthenticWord).Select
     If ((Err.number - vbObjecterror) = 2003) Then
            bEndOfDocument = True
```

```
            Err.Clear
      ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
      End If
Wend
```

SelectPrevious

See also

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description

Selects the element of type eKind before the current beginning of the range. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy 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 bEndOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
     objRange.SelectPrevious(spyAuthenticTag).Select
     If ((Err.number - vbObjecterror) = 2004) Then
            bBeginOfDocument = True
            Err.Clear
     ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
     End If
Wend
```

SetElementAttributeValue

See also

Method: SetElementAttributeValue (*strElementName* as String, *strAttributeName* as String, *strAttributeValue* as String)

Description

Retrieve the value of the attribute specified in strAttributeName for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is

returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid attribute value was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Get and set element attributes
' ------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
     If UBound(arrElements) >= 0 Then
            ' we use the top level element and find out its valid attributes
            Dim arrAttrs()
            objRange.GetElementAttributeNames arrElements(0), arrAttrs

            If UBound(arrAttrs) >= 0 Then
                  ' we retrieve the current value of the first valid
attribute
                  Dim strAttrVal
                  strAttrVal = objRange.GetElementAttributeValue
(arrElements(0), arrAttrs(0))
                  msgbox "current value of " & arrElements(0) & "//" &
arrAttrs(0) & " is: " & strAttrVal

                  ' we change this value and read it again
                  strAttrVal = "Hello"
                  objRange.SetElementAttributeValue arrElements(0),
arrAttrs(0), strAttrVal
                  strAttrVal = objRange.GetElementAttributeValue
(arrElements(0), arrAttrs(0))
                  msgbox "new value of " & arrElements(0) & "//" &
arrAttrs(0) & " is: " & strAttrVal
            End If
     End If
End If
```

SetFromRange

See also

Method: SetFromRange (*objSrcRange* as AuthenticRange)

Description

Sets the range object to the same beginning and end positions as objSrcRange .

Errors

2001 One of the two range objects, is invalid.

2005 Null object was specified as source object.

SetVariableValue

Method: `SetVariableValue( strName as string, strValue as string)`

Return Value
Sets the value (second parameter) of the variable named in the first parameter.

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

Text

See also

Property: `Text` as String

Description
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more then one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&'.

Setting the text to the empty string, does not delete any XML elements. Use `Cut`, `Delete` or `PerformAction` 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.

3.2.6 AuthenticView

See also

Properties	Methods	Events
Application	Goto	OnBeforeCopy
AsXMLString	IsRedoEnabled	OnBeforeCut
DocumentBegin	IsUndoEnabled	OnBeforeDelete
DocumentEnd	Print	OnBeforeDrop
Event	Redo	OnBeforePaste
MarkupVisibility	Undo	OnDragOver
Parent	UpdateXMLInstanceEntities	OnKeyBoardEvent
Selection		OnMouseEvent
XMLDataRoot		OnSelectionChanged
WholeDocument		

Description

AuthenticView and its child objects `AuthenticRange` and `AuthenticDataTransfer` provide you with an interface for Authentic View, which allow easy and consistent modification of document contents. These interfaces replace the following interfaces which are marked now as **obsolete**:

`OldAuthenticView` (old name was `DocEditView`)
`AuthenticSelection` (old name was `DocEditSelection`, superseded by `AuthenticRange`)
`AuthenticEvent` (old name was `DocEditEvent`)Interfaces

`AuthenticView` gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

`AuthenticView` uses objects of type `AuthenticRange` to make navigation inside the document straight-forward, and to allow for the flexible selection of logical text elements. Use the properties `DocumentBegin`, `DocumentEnd`, or `WholeDocument` for simple selections, while using the `Goto` method for more complex selections. To navigate relative to a given document range, see the methods and properties of the `AuthenticRange` object.

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
```

Events

OnBeforeCopy

See also

Event: `OnBeforeCopy()` as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticBeforeCopy()
	'On_AuthenticBeforeCopy=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticBeforeCopy()
{
	//return false;/*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (21, ...)  //nEventID=21
```

Description
This event gets triggered before a copy operation gets performed on the document. Return *True* (or nothing) to allow copy operation. Return *False* to disable copying.

OnBeforeCut

See also

Event: `OnBeforeCut()` as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticBeforeCut()
	'On_AuthenticBeforeCut=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticBeforeCut()
{
	//return false;/*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (20, ...)  //nEventID=20
```

Description
This event gets triggered before a cut operation gets performed on the document. Return *True* (or nothing) to allow cut operation. Return *False* to disable operation.

OnBeforeDelete

See also

Event: `OnBeforeDelete()` as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticBeforeDelete()
    'On_AuthenticBeforeDelete=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticBeforeDelete()
{
    //return false;/*to disable operation*/
}
```

XMLSpy 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.

OnBeforeDrop

See also

Event: `OnBeforeDrop ( i_nXPos` as Long, `i_nYPos` as Long, `i_ipRange` as `AuthenticRange`, `i_ipData` as `cancel` Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
    'On_AuthenticBeforeDrop=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
{
    //return false;/*to disable operation*/
}
```

XMLSpy 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 cancle the drop operation. Return *True* (or nothing) to continue normal operation.

Examples

```
'
--------------------------------------------------------------------------
' VB code snippet - connecting to object level events
'
--------------------------------------------------------------------------
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnBeforeDrop
' event of object objView
Private Function objView_OnBeforeDrop(ByVal i_nXPos As Long, ByVal i_nYPos
As Long,
                                      ByVal i_ipRange As IAuthenticRange,
                                      ByVal i_ipData As
IAuthenticDataTransfer) As Boolean

    If (Not i_ipRange Is Nothing) Then
        MsgBox ("Dropping on content is prohibited");
        Return False;
    Else
        Return True;
    End If
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

OnBeforePaste

See also

Event: `OnBeforePaste` (*`objData`* as Variant, *`strType`* as String) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticBeforePaste(objData, strType)
    'On_AuthenticBeforePaste=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticBeforePaste(objData, strType)
{
    //return false; /*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (19, ...)      //nEventID=19
```

Description

This event gets triggered before a paste operation gets performed on the document. The parameter *`strType`* is one of "TEXT", "UNICODETEXT" or "IUNKNOWN". In the first two cases *`objData`* contains a string representation of the object that will be pasted. In the later

case, *objData* contains a pointer to an IUnknown COM interface.

Return *True* (or nothing) to allow paste operation. Return *False* to disable operation.

OnBeforeSave

Event: OnBeforeSave (SaveAs flag) as Boolean

Description: OnBeforeSave gives the opportunity to e.g. warn the user about overwriting the existing XML document, or to make the document read-only when specific circumstances are not met. The event will be fired before the file dialog is shown. (Please note, that the event fires when saving the XML document, and not when saving the SPS design in StyleVision.)

OnDragOver

See also

Event: OnDragOver (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent, *objRange* as AuthenticRange, *objData* as AuthenticDataTransfer) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange,
objData)
      'On_AuthenticDragOver=False 'to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
{
      //return false; /*to disable operation*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn. OnEvent (10, ...)  //nEventID=10
```

Description
This event gets triggered whenever an object from within our 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.

Examples

```
'
-------------------------------------------------------------------------
' VB code snippet - connecting to object level events
'
-------------------------------------------------------------------------
```

```
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnDragOver
' event of object objView
Private Function objView_OnDragOver(ByVal i_nXPos As Long, ByVal i_nYPos As
Long,
                                    ByVal i_eMouseEvent As SPYMouseEvent,
                                    ByVal i_ipRange As IAuthenticRange,
                                    ByVal i_ipData As
IAuthenticDataTransfer) As Boolean

    If (((i_eMouseEvent And spyShiftKeyDownMask) <> 0) And
              (Not i_ipRange Is Nothing)) Then
        MsgBox ("Floating over element " &
i_ipRange.FirstXMLData.Parent.Name);
    End If

    Return True;
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

OnKeyboardEvent

See also

Event: `OnKeyboardEvent` (*eKeyEvent* as SPYKeyEvent, *nKeyCode* as Long, *nVirtualKeyStatus* as Long) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode,
nVirtualKeyStatus)
    'On_AuthenticKeyboardEvent=True 'to cancel bubbling of event
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
{
    //return true; /*to cancel bubbling of event*/
}
```

XMLSpy 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, to test for the different keys or their combinations.

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.

OnMouseEvent

See also

Event: `OnMouseEvent` (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as `SPYMouseEvent`, *objRange* as `AuthenticRange`) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticMouseEvent(nXPos,nYPos,eMouseEvent,objRange)
	'On_AuthenticMouseEvent=True 'to cancel bubbling of event
End Function
```

XMLSpy scripting environment - JScript:

```
function  On_AuthenticMouseEvent(nXPos,nYPos,eMouseEvent,objRange)
{
	//return true;/*to cancel bubbling of event*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (31, ...)  //nEventID=31
```

Description
This event gets triggered for every mouse movement and mouse button Windows message.

The actual message type and the mouse buttons status, is available in the *eMouseEvent* parameter. Use the bit-masks defined in the enumeration datatype `SPYMouseEvent` to test for the different messages, button status, and their combinations.

The parameter *objRange* identifies the part of the document found at the current mouse cursor position. The range objects always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no

selectable part of the document is found at the current position, the range object is *null*.

OnSelectionChanged

See also

Event: `OnSelectionChanged` (*`objNewSelection`* as `AuthenticRange`)

XMLSpy scripting environment - VBScript:

```
Function On_AuthenticSelectionChanged(objNewSelection)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_AuthenticSelectionChanged(objNewSelection)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (23, ...)  //nEventID=23
```

Description
This event gets triggered whenever the selection in the user interface changes.

Examples

```
'
'------------------------------------------------------------------------
' VB code snippet - connecting to object level events
'
'------------------------------------------------------------------------
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnSelectionChanged
' event of object objView
Private Sub objView_OnSelectionChanged (ByVal i_ipNewRange As
XMLSpyLib.IAuthenticRange)
    MsgBox ("new selection: " & i_ipNewRange.Text)
End Sub

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

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.

OnToolbarButtonExecuted

Event: `OnToolbarButtonExecuted` (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. See the list of predefined button identifiers.

`OnToolbarButtonExecuted` is fired after the toolbar action was executed. It is useful e.g. to add update code, as shown in the example below:

```
//event fired when a toolbar button action was executed
function On_AuthenticToolbarButtonExecuted( varBtnIdentifier )
{
	// After whatever command user has executed - make sure to update
toolbar button states
	UpdateOwnToolbarButtonStates();
}
```

In this case `UpdateOwnToolbarButtonStates` is a user function defined in the Global Declarations.

OnUserAddedXMLNode

Event: `OnUserAddedXMLNode` (XML node)

Description: `OnUserAddedXMLNode` will be fired when the user adds an XML node as a primary action. This happens in the situations, where the user clicks on

- auto-add hyperlinks (see example OnUserAddedXMLNode.sps)
- the Insert…, Insert After…, Insert Before… context menu items
- Append row, Insert row toolbar buttons
- Insert After…, Insert Before… actions in element entry helper (outside StyleVision)

The event doesn't get fired on Duplicate row, or when the node was added externally (e.g. via COM API), or on Apply (e.g. Text State Icons), or when in XML table operations or in DB operations.

The event parameter is the XML node object, which was added giving the user an opportunity to manipulate the XML node added. An elaborate example for an event handler can be found in the `OnUserAddedXMLNode.sps` file.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Accesses the Authentic Desktop application object.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

AsXMLString

See also

Property: AsXMLString as String

Description
Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new XMLString does not match the actual schema file error 2011 gets returned.

Errors
2000 The authentic view object is no longer valid.
2011 AsXMLString was set to a value which is no valid XML for the current schema file.

ContextMenu

Property: ContextMenu() as ContextMenu

Description
The property ContextMenu gives access to customize the context menu. The best place to do it is in the event handler OnContextMenuActivated.

Errors
2000 Invalid object.
2005 Invalid parameter.

CreateXMLNode

Method: CreateXMLNode (*nKind* as SPYXMLDataKind) as XMLData

Return Value
The method returns the new XMLData object.

Description
To create a new XMLData object use the CreateXMLNode() method. See also Using XMLData.

Errors
2000 Invalid object.
2012 Cannot create XML node.

DisableAttributeEntryHelper

Method: DisableAttributeEntryHelper()

Description
DisableAttributeEntryHelper() disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

DisableElementEntryHelper

Method: `DisableElementEntryHelper()`

Description

`DisableElementEntryHelper()` disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors

2000 Invalid object.

DisableEntityEntryHelper

Method: `DisableEntityEntryHelper()`

Description

`DisableEntityEntryHelper()` disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors

2000 Invalid object.

DocumentBegin

See also

Property: `DocumentBegin` as `AuthenticRange` (read-only)

Description

Retrieve a range object that points to the beginning of the document.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DocumentEnd

See also

Property: `DocumentEnd` as `AuthenticRange` (read-only)

Description

Retrieve a range object that points to the end of the document.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DoNotPerformStandardAction

Method: `DoNotPerformStandardAction ()`

Description

`DoNotPerformStandardAction()` serves as cancel bubble for macros, and stops further execution after macro has finished.

Errors
2000 Invalid object.

EvaluateXPath

Method: `EvaluateXPath` (XMLData as XMLData, strExpression as string) strValue as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expressions with the given XML context node. The result is returned as string, in the case of a sequence it is a space-separated string.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2013 XPath error.

Event

See also

Property: `Event` as AuthenticEvent (read-only)

Description
This property gives access to parameters of the last event in the same way as OldAuthenticView.event does. Since all events for the scripting environment and external clients are now available with parameters this `Event` property should only be used from within IDE-Plugins.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

EventContext

Property: `EventContext()` as `EventContext`

Description
`EventContext` property gives access to the running macros context. See the `EventContext` interface description for more details.

Errors
2000 Invalid object.

GetToolbarButtonState

Method: `GetToolbarButtonState` (`ButtonIdentifier` as `string`) as `AuthenticToolbarButtonState`

Return Value

The method returns `AuthenticToolbarButtonState`

Description

`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors

2000	Invalid object.
2005	Invalid parameter.
2008	Internal error.
2014	Invalid button identifier.

Goto

See also

Method: `Goto` (*`eKind`* as SPYAuthenticElementKind, *`nCount`* as Long, *`eFrom`* as *`SPYAuthenticDocumentPosition`*) as *`AuthenticRange`*

Description

Retrieve a range object that points to the beginning of the *`nCount`* element of type *`eKind`*. The start position is defined by the parameter *`eFrom`*. Use positive values for *`nCount`* to navigate to the document end. Use negative values to navigate towards the beginning of the document.

Errors

2000	The authentic view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. The document position to start from is not one of *spyAuthenticDocumentBegin* or *spyAuthenticDocumentEnd*. Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------
' XMLSpy 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
```

IsRedoEnabled

See also

Property: `IsRedoEnabled` as Boolean (read-only)

Description
True if redo steps are available and `Redo` is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsUndoEnabled

See also

Property: `IsUndoEnabled` as Boolean (read-only)

Description
True if undo steps are available and `Undo` is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MarkupVisibility

See also

Property: `MarkupVisibility` as `SPYAuthenticMarkupVisibility`

Description
Set or get current visibility of markup.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Document` (read-only)

Description
Access the document shown in this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Print

See also

Method: `Print` (*`bWithPreview`* as Boolean, *`bPromptUser`* as Boolean)

Description
Print the document shown in this view. If *bWithPreview* is set to *True*, the print preview dialog pops up. If *bPromptUser* is set to *True*, the print dialog pops up. If both parameters are set to *False*, the document gets printed without further user interaction.

Errors
2000 The authentic view object is no longer valid.

Redo

See also

Method: `Redo()` as Boolean

Description
Redo the modification undone by the last undo command.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Selection

See also

Property: `Selection` as `AuthenticRange`

Description
Set or get current text selection in user interface.

Errors
2000 The authentic view object is no longer valid.
2002 No cursor selection is active.

2005 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------
' XMLSpy 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
```

SetToolbarButtonState

Method: `SetToolbarButtonState` (`ButtonIdentifier` as string, `AuthenticToolbarButtonState` state)

Description

`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors

2000 Invalid object.
2008 Internal error.
2014 Invalid button identifier.

Undo

See also

Method: Undo( ) as Boolean

Description

Undo the last modification of the document from within this view.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

UpdateXMLInstanceEntities

See also

Method: UpdateXMLInstanceEntities()

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example

```
// ----------------------------------------
// XMLSpy 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();
     }
}
```

WholeDocument

See also

Property: WholeDocument as AuthenticRange (read-only)

Description
Retrieve a range object that selects the whole document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

XMLDataRoot

See also

Property: XMLDataRoot as XMLData (read-only)

Description

Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

3.2.7 CodeGeneratorDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Properties and Methods

Standard automation properties
Application
Parent

Programming language selection properties
ProgrammingLanguage
TemplateFileName
CompatibilityMode

Settings for C++ code
CPPSettings_DOMType
CPPSettings_LibraryType
CPPSettings_UseMFC
CPPSettings_GenerateVC6ProjectFile
CPPSettings_GenerateVSProjectFile

Settings for C# code
CSharpSettings_ProjectType

Dialog handling for above code generation properties
PropertySheetDialogAction

Output path selection properties
OutputPath
OutputPathDialogAction

Presentation of result
OutputResultDialogAction

Description
Use this object to configure the generation of program code for schema files. The method GenerateProgramCode expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

Application

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: `Application` as `Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

CompatibilityMode

Property: `CompatibilityMode` as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Set to true to generate code compatible to XMLSpy 2005R3. Set to false to use newly added code-generation features.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_DOMType

Property: `CPPSettings_DOMType` as `SPYDOMType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateVC6ProjectFile

Property: `CPPSettings_GenerateVC6ProjectFile` as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateGCCMakefile

Property: `CPPSettings_GenerateGCCMakefile` as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Creates makefiles to compile the generated code under Linux with GCC.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateVSProjectFile

Property: `CSharpSettings_GenerateVSProjectFile` as `SPYProjectType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code. Only `spyVisualStudio2005Project (=4)` and `spyVisualStudio2008Project (=5)` and `spyVisualStudio2010Project (=6)` are valid project types.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_LibraryType

Property: `CPPSettings_LibraryType` as `SPYLibType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_UseMFC

Property: `CPPSettings_UseMFC` as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CSharpSettings_ProjectType

Property: `CSharpSettings_ProjectType` as `SPYProjectType`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines the only setting to configure generation of C# code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputPath

Property: `OutputPath` as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the base directory for all generated code.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

OutputPathDialogAction

Property: `OutputPathDialogAction` as `SPYDialogAction`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog for selecting the code generation output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputResultDialogAction

Property: `OutputResultDialogAction` as `SPYDialogAction`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that asks to show the result of the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

Parent

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ProgrammingLanguage

Property: `ProgrammingLanguage` as `ProgrammingLanguage`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the output language for the code to be generated.

CAUTION: Setting this property to one of C++, C# or Java, changes the property TemplateFileName to the appropriate template file delivered with Authentic Desktop as well. If you want to generate C++, C# or Java code based on your own templates, set first the programming language and then select your template file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

PropertySheetDialogAction

Property: `PropertySheetDialogAction` as `SPYDialogAction`
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that configures the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current values as defaults. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

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 property to its appropriate value.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

3.2.8 DatabaseConnection

See also

Properties for import and export
File or
ADOConnection or
ODBCConnection

Properties for import only
DatabaseKind
SQLSelect
AsAttributes
ExcludeKeys
IncludeEmptyElements
NumberDateTimeFormat
NullReplacement
CommentIncluded

Properties for export only
CreateMissingTables
CreateNew
TextFieldLen
DatabaseSchema

Properties for XML Schema from DB Structure generation
PrimaryKeys
ForeignKeys
UniqueKeys
SchemaExtensionType
SchemaFormat
ImportColumnsType

Description
DatabaseConnection specifies the parameters for the database connection.

Please note that the properties of the DatabaseConnection interface are referring to the settings of the import and export dialogs of Authentic Desktop.

ADOConnection

See also

Property: ADOConnection as String

Description
The property ADOConnection contains a connection string. Either use this property or ODBCConnection or File to refer to a database.

Errors
No error codes are returned.

Example

```
Dim objSpyConn As DatabaseConnection
Set objSpyConn = objSpy.GetDatabaseSettings
```

```
Dim objADO As DataLinks
Set objADO = CreateObject("DataLinks")

If Not (objADO Is Nothing) Then
 Dim objConn As Connection
 Set objConn = objADO.PromptNew
 objSpyConn.ADOConnection = objConn.ConnectionString
End If
```

AsAttributes

See also

Property: `AsAttributes` as Boolean

Description
Set `AsAttributes` to true if you want to initialize all import fields to be imported as attributes. Default is false and will initialize all fields to be imported as elements. This property is used only in calls to Application.GetDatabaseImportElementList.

Errors
No error codes are returned.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated XML. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

CreateMissingTables

See also

Property: `CreateMissingTables` as Boolean

Description
If `CreateMissingTables` is true, tables which are not already defined in the export database will be created during export. Default is true. This property is used only when exporting to databases.

Errors
No error codes are returned.

CreateNew

See also

Property: `CreateNew` as Boolean

Description
Set `CreateNew` true if you want to create a new database on export. Any existing database will be overwritten. See also `DatabaseConnection.File`. Default is false. This property is used only when exporting to databases.

Errors
No error codes are returned.

DatabaseKind

See also

Property: `DatabaseKind` as `SPYDatabaseKind`

Description
Select the kind of database that gets access. The default value is `spyDB_Unspecified(7)` and is sufficient in most cases. This property is used only when importing from databases.

Errors
No error codes are returned.

DatabaseSchema

See also

Property: DatabaseSchema as String

Description
This property specifies the Schema used for export in Schema aware databases. Default is "". This property is used only when exporting to databases.

Errors
No error codes are returned.

ExcludeKeys

See also

Property: `ExcludeKeys` as Boolean

Description
Set `ExcludeKeys` to true if you want to exclude all key columns from the import data. Default is false. This property is used only when importing from databases.

Errors
No error codes are returned.

File

See also

Property: `File` as String

Description
The property `File` sets the path for the database during export or import. This property can only be used in conjunction with a Microsoft Access database. Either use this property or `ODBCConnection` or `ADOConnection` to refer to the database.

See also Import and Export.

Errors
No error codes are returned.

ForeignKeys

See also

Property: `ForeignKeys` as `Boolean`

Description
Specifies whether the Foreign Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

ImportColumnsType

See also

Property: `ImportColumnsType` as SPYImportColumnsType

Description
Defines if column information from the DB is saved as element or attribute in the XML Schema. Default is as element. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

IncludeEmptyElements

See also

Property: `IncludeEmptyElements` as `Boolean`

Description
Set `IncludeEmptyElements` to false if you want to exclude all empty elements. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

NullReplacement

See also

Property: NullReplacement as String

Description
This property contains the text value that is used during import for empty elements (null values). Default is "". This property is used only when importing from databases.

Errors
No error codes are returned.

NumberDateTimeFormat

See also

Property: `NumberDateTimeFormat` as `SPYNumberDateTimeFormat`

Description
The property `NumberDateTimeFormat` sets the format of numbers and date- and time-values. Default is `spySystemLocale`. This property is used only when importing from databases.

Errors
No error codes are returned.

ODBCConnection

See also

Property: `ODBCConnection` as String

Description
The property `ODBCConnection` contains a ODBC connection string. Either use this property or `ADOConnection` or `File` to refer to a database.

Errors
No error codes are returned.

PrimaryKeys

See also

Property: `PrimaryKeys` as Boolean

Description
Specifies whether the Primary Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

SchemaExtensionType

See also

Property: `SchemaExtensionType` as SPYSchemaExtensionType

Description
Defines the Schema extension type used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure.

Errors
No error codes are returned.

SchemaFormat

See also

Property: `SchemaFormat` as SPYSchemaFormat

Description
Defines the Schema format used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure.

Errors
No error codes are returned.

SQLSelect

See also

Property: `SQLSelect` as String

Description
The SQL query for the import is stored in the property `SQLSelect`. This property is used only when importing from databases. See also Import and Export.

Errors
No error codes are returned.

TextFieldLen

See also

Property: `TextFieldLen` as Long

Description
The property `TextFieldLen` sets the length for created text fields during the export. Default is 255. This property is used only when exporting to databases.

Errors

No error codes are returned.

UniqueKeys

See also

Property: `UniqueKeys` as `Boolean`

Description
Specifies whether the Unique Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

3.2.9 Dialogs

See also

Properties and Methods

Standard automation properties
Application
Parent

Various dialog objects
CodeGeneratorDlg
FileSelectionDlg
SchemaDocumentationDlg
GenerateSampleXMLDlg
DTDSchemaGeneratorDlg
FindInFilesDlg
WSDLDocumentationDlg
WSDL20DocumentationDlg
XBRLDocumentationDlg

Description
The Dialogs object provides access to different built-in dialogs of Authentic Desktop. These dialog objects allow to initialize the fields of user dialogs before they get presented to the user or allow to simulate complete user input by your program.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

CodeGeneratorDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: `CodeGeneratorDlg` as `CodeGeneratorDlg` (read-only)

Description
Get a new instance of a code generation dialog object. You will need this object to pass the necessary parameters to the code generation methods. Initial values are taken from last usage of the code generation dialog.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

FileSelectionDlg

See also

Property: `FileSelectionDlg` as `FileSelectionDlg` (read-only)

Description
Get a new instance of a file selection dialog object.

File selection dialog objects are passed to you with the some events that signal opening or saving of documents and projects.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

SchemaDocumentationDlg

See also

Property: `SchemaDocumentationDlg` as `SchemaDocumentationDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of schema documentation. See Document.GenerateSchemaDocumentation 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.

GenerateSampleXMLDlg

See also

Property: `GenerateSampleXMLDlg` as `GenerateSampleXMLDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a sample XML based on a W3C schema or DTD. See GenerateSampleXML for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

DTDSchemaGeneratorDlg

See also

Property: `DTDSchemaGeneratorDlg` as `DTDSchemaGeneratorDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a schema or DTD. See Document.GenerateDTDOrSchemaEx for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

FindInFilesDlg

See also

Property: FindInFilesDlg as FindInFilesDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes the search (or replacement) of strings in files. See Application.FindInFiles for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

WSDLDocumentationDlg

See also

Property: `WSDLDocumentationDlg` as `WSDLDocumentationDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL documentation. See Document.GenerateWSDLDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

WSDL20DocumentationDlg

See also

Property: `WSDL20DocumentationDlg` as `WSDL20DocumentationDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL 2.0

documentation. See Document.GenerateWSD20LDocumentation for its usage.

Errors

2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

XBRLDocumentationDlg

See also

Property: `XBRL20DocumentationDlg` as `XBRL20DocumentationDlg` (read-only)

Description

Get a new instance of a dialog object that parameterizes generation of WSDL 2.0 documentation. See Document.GenerateXBRLDocumentation for its usage.

Errors

2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

3.2.10 Document

See also

Properties and Methods

Standard automation properties
Application
Parent

Various document properties and methods
SetActiveDocument
Encoding
SetEncoding (obsolete)
Suggestions

XML validation
IsWellFormed
IsValid
SetExternalIsValid

Document conversion and transformation
AssignDTD
AssignSchema
AssignXSL
AssignXSLFO
ConvertDTDOrSchema
ConvertDTDOrSchemaEx
GenerateDTDOrSchema
GenerateDTDOrSchemaEx
CreateSchemaDiagram
ExecuteXQuery
TransformXSL
TransformXSLEx
TransformXSLFO
GenerateProgramCode (Enterprise Edition only)
GenerateSchemaDocumentation
GenerateSampleXML
GenerateWSDL20Documentation
GenerateWSDLDocumentation
GenerateXBRLDocumentation
ConvertToWSDL20

Document export
GetExportElementList
ExportToText
ExportToDatabase
CreateDBStructureFromXMLSchema
GetDBStructureList

File saving and naming
FullName
Name
Path
GetPathName (obsolete)
SetPathName (obsolete)
Title
IsModified
Saved

SaveAs
Save
SaveInString
SaveToURL
Close

View access
CurrentViewMode
SwitchViewMode
AuthenticView
GridView
DocEditView (obsolete)

Access to XMLData
RootElement
DataRoot
CreateChild
UpdateViews
StartChanges
EndChanges
UpdateXMLData

Description
Document objects represent XML documents opened in Authentic Desktop.

Use one of the following properties to access documents that are already open Authentic Desktop:
Application.ActiveDocument
Application.Documents

Use one of the following methods to open a new document in Authentic Desktop:
Documents.OpenFile
Documents.OpenURL
Documents.OpenURLDialog
Documents.NewFile
Documents.NewFileFromText
SpyProjectItem.Open
Application.ImportFromDatabase
Application.ImportFromSchema
Application.ImportFromText
Application.ImportFromWord
Document.ConvertDTDOrSchema
Document.GenerateDTDOrSchema

Events

OnBeforeSaveDocument

See also

Event: OnBeforeSaveDocument(*objDocument* as Document, *objDialog* as FileSelectionDlg)

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 FileSelectionDlgDialogAction property of *io_objDialog* at its default value spyDialogOK. To abort saving of the document set this property to spyDialogCancel.

OnBeforeCloseDocument

See also

Event: `OnBeforeCloseDocument(objDocument as Document)as Boolean`

XMLSpy scripting environment - VBScript:

```
Function On_BeforeCloseDocument(objDocument)
    'On_BeforeCloseDocument=False 'to prohibit closing of document
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeCloseDocument(objDocument)
{
    //return false;/*to prohibit closing of document*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (28, ...)  //nEventID=28
```

Description

This event gets fired on any attempt to close a document. To prevent the document from being closed return false.

OnBeforeValidate

See also

Event: OnBeforeValidate(*objDocument* as Document, *bOnLoading* as Boolean, *bOnCommand* as Boolean) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
    On_BeforeValidate=bCancelDefaultValidation          'set by the script if
necessary
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeValidate(objDocument, eViewMode, bActivated)
{
    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 SetExternalIsValid method.

bOnLoading is true if the the event is raised on the initial validation on loading the document.

bOnCommand is true whenever the user selected the Validate command from the Toolbar or menu.

Available with TypeLibrary version 1.5

OnCloseDocument

See also

Event: OnCloseDocument(*objDocument* as Document)

XMLSpy scripting environment - VBScript:

```
Function On_Close Document(objDocument)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_Close Document(objDocument)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (8, ...)  //nEventID=8
```

Description

This event gets fired as a result of closing a document. Do not modify the document from within this event.

OnViewActivation

See also

Event: OnViewActivation(*objDocument* as Document, *eViewMode* as SPYViewModes, *bActivated* as Boolean)

XMLSpy scripting environment - VBScript:

```
Function On_ViewActivation(objDocument, eViewMode, bActivated)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_ViewActivation(objDocument, eViewMode, bActivated)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (29, ...)  //nEventID=29
```

Description
This event gets fired whenever a view of a document becomes visible (i.e. becomes the active view) or invisible (i.e. another view becomes the active view or the document gets closed). However, the first view activation event after a document gets opened cannot be received, since there is no document object to get the event from. Use the Application.OnDocumentOpened event instead.

Application

See also

Property: Application as Application (read-only)

Description
Accesses the Authentic Desktop application object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

AssignDTD

See also

Method: AssignDTD(*strDTDFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the DTD file "strDTDFile" into the document. Note that no error occurs if the file does not exist, or is not accessible. If `bDialog` is true Authentic Desktop presents a dialog to set the file.

See also Simple document access.

Errors
1400 The object is no longer valid.

1409 You are not allowed to assign a DTD to the document.

AssignSchema

See also

Method: `AssignSchema` (*`strSchemaFile`* as String, *`bDialog`* as Boolean)

Description
The method places a reference to the schema file "strSchemaFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If `bDialog` is true Authentic Desktop presents a dialog to set the file.

See also Simple document access.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a schema file to the document.

AssignXSL

See also

Method: `AssignXSL` (*`strXSLFile`* as String, *`bDialog`* as Boolean)

Description
The method places a reference to the XSL file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If `bDialog` is true Authentic Desktop presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

AssignXSLFO

See also

Method: `AssignXSLFO` (*`strXSLFOFile`* as String, *`bDialog`* as Boolean)

Description
The method places a reference to the XSLFO file "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.

AsXMLString

See also

Property: AsXMLString as String

Description
This property can be used to get or set the document content.

Errors
1400 The document object is no longer valid.
1404 Cannot create XMLData object.
1407 View mode cannot be switched.

AuthenticView

See also

Method: AuthenticView as AuthenticView (read-only)

Description
Returns an object that gives access to properties and methods specific to Authentic view. The object returned is only valid if the current document is opened in Authentic view mode. The lifetime of an object ends with the next view switch. Any attempt to access objects or any of its children afterwards will result in an error indicating that the object is invalid.

AuthenticView and DocEditView both provide automation access to the Authentic view mode of XMLSpy. Functional overlap is intentional. A future version of Authentic View will include all functionality of DocEditView and its sub-objects, thereby making usage of DocEditView obsolete.

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
```

Close

See also

Method: `Close` (*`bDiscardChanges`* as Boolean)

Description
To close the document call this method. If `bDiscardChanges` is true and the document is modified, the document will be closed but not saved.

Errors
1400 The object is no longer valid.
1401 Document needs to be saved first.

ConvertDTDOrSchema

See also

Method: `ConvertDTDOrSchema` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nFrequentElements`* as `SPYFrequentElements`)

Parameters
`nFormat`
Sets the schema output format to DTD, or W3C.

`nFrequentElements`
Create complex elements as elements or complex types.

Description
`ConvertDTDOrSchema` takes an existing schema format and converts it into a different format. For a finer tuning of DTD / schema conversion, use ConvertDTDOrSchemaEx.

Errors
1400 The object is no longer valid.
1412 Error during conversion.

ConvertDTDOrSchemaEx

See also

Method: `ConvertDTDOrSchemaEx` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nFrequentElements`* as `SPYFrequentElements`, `sOutputPath` as `String`, `nOutputPathDialogAction` as `SPYDialogAction`)

Parameters
`nFormat`
Sets the schema output format to DTD, or W3C.

`nFrequentElements`
Create complex elements as elements or complex types.

sOutputPath
The file path for the newly generated file.

nOutputPathDialogAction
Defines the dialog interaction for this call.

Description
`ConvertDTDOrSchemaEx` takes an existing schema format and converts it into a different

format.

Errors
1400 The object is no longer valid.
1412 Error during conversion.

ConvertToWSDL20

Method: `ConvertToWSDL20` (*`sFilePath`* as String, *bShowDialogs* as Boolean)

Parameters
`sFilePath`
This specifies the file name of the converted WSDL. In case the source WSDL includes files which also must be converted, then only the directory part of the given path is used and the file names are generated automatically.

`bShowDialogs`
Defines whether file/folder selection dialogs are shown.

Description
Converts the WSDL 1.1 document to a WSDL 2.0 file. It will also convert any referenced WSDL files that are referenced from within this document. Note that this functionality is limited to WSDL View only. See `Document.CurrentViewMode`. and `SPYViewModes`.

Errors
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.

CreateChild

See also

Method: `CreateChild` (*`nKind`* as `SPYXMLDataKind`) as `XMLData`

Return Value
The method returns the new `XMLData` object.

Description
To create a new `XMLData` object use the `CreateChild()` method. See also Using XMLData.

Errors
1400 The object is no longer valid.
1404 Cannot create XMLData object.
1407 Invalid address for the return parameter was specified.

CreateDBStructureFromXMLSchema

See also

Method: `CreateDBStructureFromXMLSchema` (*`pDatabase`* as `DatabaseConnection`,

pTables as ElementList, *bDropTableWithExistingName* as Boolean) as String

Description
CreateDBStructureFromXMLSchema exports the given tables to the specified database. The function returns the SQL statements that were necessary to perform the changes.

See also GetDBStructureList.

Errors
1429 Database selection missing.
1430 Document export failed.

CreateSchemaDiagram

See also

Method: CreateSchemaDiagram (*nKind* as SPYSchemaDefKind, *strName* as String, *strFile* as String)

Return Value
None.

Description
The method creates a diagram of the schema type strName of kind nKind and saves the output file into strFile. Note that this functionality is limited to Schema View only. See Document. CurrentViewMode. and SPYViewModes.

Errors
1400 The object is no longer valid.
1414 Failed to save diagram.
1415 Invalid schema definition type specified.

CurrentViewMode

See also

Method: CurrentViewMode as SPYViewModes

Description
The property holds the current view mode of the document. See also Document. SwitchViewMode.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

DataRoot

See also

Property: DataRoot as XMLData (read-only)

Description
This property provides access to the document's first XMLData object of type *spyXMLDataElement*. This is typically the root element for all document content data. See XMLSpyDocument.RootElement to get the root element of the whole document including XML prolog data. If the CurrentViewMode is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData may be necessary to get access to the latest XMLData.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

DocEditView

See also

Method: `DocEditView` as `DocEditView`

Description
Holds a reference to the current Authentic View object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in authentic view mode.

Encoding

See also

Property: `Encoding` as String

Description
This property provides access to the document's encoding value. However, this property can only be accessed when the document is opened in *spyViewGrid*, *spyViewText* or *spyViewAuthentic*. See CurrentViewMode on how to detect that a document's actual view mode.

This property makes the method SetEncoding obsolete.

Possible values are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

Errors
1400 The document object is no longer valid.

1407 Invalid address for the return parameter was specified.
1416 Operation not supported in current view mode.

EndChanges

See also

Method: `EndChanges()`

Description
Use the method `EndChanges` to display all changes since the call to `Document.StartChanges`.

Errors
1400 The object is no longer valid.

ExecuteXQuery

See also

Method: `ExecuteXQuery` (*`strXMLFileName`* as String)

Description
Execute the XQuery statements contained in the document. Use the XML file specified as XML source for the transformation. If your XQuery script does not use an XML source set the parameter `strXMLFileName` to an empty string.

Errors
1400 The document object is no longer valid.
1423 XQuery transformation error.
1424 Not all files required for operation could be loaded. Most likely, the file specified in `strXMLFileName` does not exist or is not valid.

ExportToDatabase

See also

Method: `ExportToDatabase` (*`pFromChild`* as `XMLData`, *`pExportSettings`* as `ExportSettings`, *`pDatabase`* as `DatabaseConnection`)

Description
`ExportToDatabase` exports the XML document starting with the element `pFromChild`. The parameter `pExportSettings` defines the behaviour of the export (see `Application.GetExportSettings`). The parameter `pDatabase` specifies the destination of the export (see `Application.GetDatabaseSettings`). `UpdateXMLData()` might be indirectly needed as you have to pass the `XMLData` as parameter to this function.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1416 Error during export.
1429 Database selection missing.

1430 Document export failed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument

'set the behaviour of the export with ExportSettings
Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings

'set the destination with DatabaseConnection
Dim objDB As DatabaseConnection
Set objDB = objSpy.GetDatabaseSettings

objDB.CreateMissingTables = True
objDB.CreateNew = True
objDB.File = "C:\Export.mdb"

objDoc.ExportToDatabase objDoc.RootElement, objExpSettings, objDB
If Err.Number <> 0 Then
 a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
   "Description: " & Err.Description)
End If
```

ExportToText

See also

Method: ExportToText (*pFromChild* as XMLData, *pExportSettings* as ExportSettings, *pTextSettings* as TextImportExportSettings)

Description

ExportToText exports tabular information from the document starting at pFromChild into one or many text files. Columns of the resulting tables are generated in alphabetical order of the column header names. Use GetExportElementList to learn about the data that will be exported. The parameter pExportSettings defines the specifics for the export. Set the property ExportSettings.ElementList to the - possibly modified - list returned by GetExportElementList to avoid exporting all contained tables. The parameter pTextSettings defines the options specific to text export and import. You need to set the property TextImportExportSettings.DestinationFolder before you call ExportToText. UpdateXMLData() might be indirectly needed as you have to pass the XMLData as parameter to this function.

See also Import and export of data.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1416 Error during export.
1430 Document export failed.

Example

```
' ----------------------------------------------------------
' VBA client code fragment - export document to text files
' ----------------------------------------------------------
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument
```

```
 Dim objExpSettings As ExportSettings
 Set objExpSettings = objSpy.GetExportSettings
 objExpSettings.ElementList = objDoc.GetExportElementList(
        objDoc.RootElement,
        objExpSettings)

 Dim objTextExp As TextImportExportSettings
 Set objTextExp = objSpy.GetTextExportSettings
 objTextExp.HeaderRow = True
 objTextExp.DestinationFolder = "C:\Exports"

 On Error Resume Next
 objDoc.ExportToText objDoc.RootElement, objExpSettings, objTextExp

 If Err.Number <> 0 Then
  a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
"Description: "     & Err.Description)
 End If
```

FullName

See also

Property: `FullName` as String

Description

This property can be used to get or set the full file name - including the path - to where the document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName and SetPathName obsolete.

Errors

1400 The document object is no longer valid.
1402 Empty string has been specified as full file name.

GenerateDTDOrSchema

See also

Method: `GenerateDTDOrSchema` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nValuesList`* as integer, *`nDetection`* as `SPYTypeDetection`, *`nFrequentElements`* as `SPYFrequentElements`)

Parameters

`nFormat`
Sets the schema output format to DTD, or W3C.

`nValuesList`
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

`nDetection`
Specifies granularity of simple type detection.

`nFrequentElements`
Shall the types for all elements be defined as global? Use that value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Description

Use this method to automatically generate a DTD or schema for the current XML document. For a finer tuning of DTD / schema generation, use GenerateDTDOrSchemaEx.
Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode. and SPYViewModes.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateDTDOrSchemaEx

See also

Method: GenerateDTDOrSchemaEx (*objDlg* as DTDSchemaGeneratorDlg) as Document

Description

Use this method to automatically generate a DTD or schema for the current XML document. A DTDSchemaGeneratorDlg object is used to pass information to the schema/DTD generator. The generation process can be configured to allow user interaction or run without further user input.
Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode. and SPYViewModes.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateProgramCode

Method: GenerateProgramCode (*objDlg* as CodeGeneratorDlg)
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description

Generate Java, C++ or C# class files from the XML Schema definitions in your document. A CodeGeneratorDlg object is used to pass information to the code generator. The generation process can be configured to allow user interaction or run without further user input.

Errors

1400 The document object is no longer valid.
1407 An empty file name has been specified.
1421 Feature not available in this edition

GenerateSampleXML

Method: GenerateSampleXML (*objDlg* as GenerateSampleXMLDlg) as Document

Description

Generates a sample XML if the document is a schema or DTD. Use Dialogs.GenerateSampleXMLDlg to get an initialized set of options.

Available with TypeLibrary version 1.5

Errors

1400 The document object is no longer valid.

GenerateSchemaDocumentation

Method: GenerateSchemaDocumentation (*objDlg* as SchemaDocumentationDlg)

Description

Generate documentation for a schema definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.SchemaDocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property SchemaDocumentationDlg.OutputFile before starting the generation process. Note that this functionality is limited to Schema View only. See Document.CurrentViewMode and SPYViewModes.

Errors

1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GenerateWSDL20Documentation

Method: GenerateWSDL20Documentation (*objDlg* as WSDL20DocumentationDlg)

Description

Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.WSDL20DocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property WSDL20DocumentationDlg.OutputFile before starting the generation process. Note that this functionality is limited to WSDL View only. See Document.CurrentViewMode and SPYViewModes.

Errors

1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GenerateWSDLDocumentation

Method: GenerateWSDLDocumentation (*objDlg* as WSDLDocumentationDlg)

Description

Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.WSDLDocumentationDlg to get an initialized set of options. As a minimum, you will

need to set the property WSDLDocumentationDlg.OutputFile before starting the generation process. Note that this functionality is limited to WSDL View only. See Document.CurrentViewMode and SPYViewModes.

Errors

1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GenerateXBRLDocumentation

Method: GenerateXBRLDocumentation (*objDlg* as XBRLDocumentationDlg)

Description

Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.XBRLDocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property XBRLDocumentationDlg.OutputFile before starting the generation process. Note that this functionality is limited to XBRL View only. See Document.CurrentViewMode and SPYViewModes.

Errors

1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GetDBStructureList

See also

Method: GetDBStructureList (*pDatabase* as DatabaseConnection) as ElementList

Description

GetDBStructureList creates a collection of elements from the Schema document for which tables in the specified database are created. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the tables.

See also CreateDBStructureFromXMLSchema.

Errors

1400 The object is no longer valid.
1427 Failed creating parser for the specified XML.
1428 Export of element list failed.
1429 Database selection missing.

GetExportElementList

See also

Method: GetExportElementList (*pFromChild* as XMLData, *pExportSettings* as ExportSettings) as ElementList

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 ElementListItems where the properties ElementListItem.Name contain the names of the tables that can be exported from the document. The property ElementListItem.FieldCount contains the number of columns in the table. The property ElementListItem.RecordCount contains the number of records in the table. The property ElementListItem.ElementKind is unused. UpdateXMLData() might be indirectly needed as you have to pass the XMLData as parameter to this function.

See also Import and export of data.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1427	Failed creating parser for the specified XML.
1428	Export of element list failed.

GetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

See also

Method: GetPathName() as String

Description
The method GetPathName gets the path of the active document.

See also Document.SetPathName (obsolete).

GridView

See also

Property: GridView as GridView

Description
This property provides access to the grid view functionality of the document.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in enhanced grid view mode.

IsModified

See also

Property: `IsModified` as Boolean

Description
True if the document is modified.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

IsValid

See also

Method: `IsValid` (*`strError`* as Variant, *`nErrorPos`* as Variant, *`pBadData`* as Variant) as Boolean

Return Value
True if the document is valid, false if not.

Description
IsValid() validates the document against its associated schema or DTD. strError gives you the same error message as Authentic Desktop, if you validate the file within the editor.

nErrorPos is obsolete and only present for compatibility reasons. pBadData is a reference to the XMLData object raising the error. It is only available if the document is currently displayed in TextView, GridView or AuthenticView.

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.

Examples

For an example written in VBA see Overview - Simple document access

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' ---------------------------------------
Dim errorText
Dim errorPos
```

```
Dim badData

' validate the active document
Set doc = Application.ActiveDocument
valid = doc.IsValid(errorText, errorPos, badData)

If (Not valid) Then
	' create the validation error message text
	Text = errorText

	If ( Not IsEmpty(badData) ) Then
		Text = Text & "(" & badData.Name & "/" & badData.TextValue & ")"
	End If

	Call MsgBox("validation error[" & errorPos & "]: " & Text)
End If
```

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// ---------------------------------------
// define as arrays to support their usage as return parameters
var errorText = new Array(1);
var errorPos = new Array(1);
var badData = new Array(1);

// validate the active document
var doc = Application.ActiveDocument;
var valid = doc.IsValid(errorText, errorPos, badData);

if (! valid)
{
	// compose the error description
	var text = errorText;

	// access that XMLData object only if filled in
	if (badData[0] != null)
		text += "(" + badData[0].Name + "/" + badData[0].TextValue + ")"
;

	MsgBox("validation error[" + errorPos + "]: " + text);
}
```

IsWellFormed

See also

Method: `IsWellFormed` (*`pData`* as `XMLData`, *`bWithChildren`* as `Boolean`, *`strError`* as Variant, *`nErrorPos`* as Variant, *`pBadXMLData`* as Variant) as Boolean

Return Value

True if the document is well formed.

Description

`IsWellFormed` checks the document for well-formedness starting at the element `pData`.

If the document is not well formed, `strError` contains an error message, `nErrorPos` the position in the file and `pBadXMLData` holds a reference to the element which breaks the well-formedness. These out-parameters are defined as VARIANTs to support scripting languages like VBScript.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

Example

See IsValid.

Name

See also

Property: `Name` as String (read-only)

Description
Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Documents` (read-only)

Description
Access the parent of the document object.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Property: `Parent` as `Application` (read-only)

Path

See also

Property: `Path` as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

RootElement

See also

Property: `RootElement` as `XMLData` (read-only)

Description
The property `RootElement` provides access to the root element of the XML structure of the document including the XML prolog data. To access the first element of a document's content navigate to the first child of kind *spyXMLDataElement* or use the Document.DataRoot property. If the CurrentViewMode is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData may be necessary to get access to the latest XMLData.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Save

See also

Method: `Save()`

Description
The method writes any modifications of the document to the associated file. See also `Document.FullName`.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

SaveAs

See also

Method: `SaveAs` (*`strFileName`* as String)

Description
Save the document to the file specified. If saving was successful, the FullName property gets set to the specified file name.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

Saved

See also

Property: `Saved` as Boolean (read-only)

Description
This property can be used to check if the document has been saved after the last modifications. It returns the negation of IsModified.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

SaveInString

See also

Method: `SaveInString` (*`pData`* as `XMLData`, *`bMarked`* as Boolean) as String

Parameters
`pData`
`XMLData` element to start. Set `pData` to `Document.RootElement` if you want to copy the complete file.

`bMarked`
If `bMarked` is true, only the elements selected in the grid view are copied.

Return Value
Returns a string with the XML data.

Description
`SaveInString` starts at the element `pData` and converts the `XMLData` objects to a string representation. `UpdateXMLData()` might be indirectly needed as you have to pass the `XMLData` as parameter to this function.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

SaveToURL

See also

Method: `SaveToURL` (*`strURL`* as String, *`strUser`* as String, *`strPassword`* as String)

Return Value

Description
`SaveToURL()` writes the document to the URL `strURL`. This method does not set the permanent file path of the document.

Errors
1400 The object is no longer valid.
1402 Invalid URL specified.
1403 Error while saving to URL.

SetActiveDocument

See also

Method: `SetActiveDocument()`

Description
The method sets the document as the active and brings it to the front.

Errors
1400 The object is no longer valid.

SetEncoding (obsolete)

Superseded by Document.Encoding

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetEncoding("UTF-16");
// use now:
Application.ActiveDocument.Encoding = "UTF-16";
```

See also

Method: `SetEncoding` (*`strEncoding`* as String)

Description
`SetEncoding` sets the encoding of the document like the menu item "File/Encoding..." in Authentic Desktop. Possible values for `strEncoding` are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

SetExternalIsValid

See also

Method: `SetExternalIsValid` (*`bValid`* as Boolean)

Parameters

`bValid`
Sets the result of an external validation process.

Description
The internal information set by this method is only queried on cancelling the default validation in any OnBeforeValidate handler.

Available with TypeLibrary version 1.5

Errors
1400 The object is no longer valid.

SetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

See also

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.

StartChanges

See also

Method: `StartChanges()`

Description
`After StartChanges` is executed Authentic Desktop will not update its editor windows until `Document.EndChanges` is called. This increases performance of complex tasks to the XML structure.

Errors

1400 The object is no longer valid.

Suggestions

Property: Suggestions as Array

Description
This property contains the last valid user suggestions for this document. The XMLSpy generated suggestions can be modified before they are shown to the user in the OnBeforeShowSuggestions event.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

SwitchViewMode

See also

Method: `SwitchViewMode` (*`nMode`* as `SPYViewModes`) as Boolean

Return value
Returns true if view mode is switched.

Description
The method sets the current view mode of the document in Authentic Desktop. See also `Document.CurrentViewMode`.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Invalid view mode specified.

TextView

See also

Property: TextView as TextView

Description
This property provides access to the text view functionality of the document.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

Title

See also

Property: `Title` as String (read-only)

Description
`Title` contains the file name of the document. To get the path and filename of the file use

FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

TransformXSL

See also

Method: TransformXSL()

Description
TransformXSL processes the XML document via the associated XSL file. See Document.AssignXSL on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLEx

See also

Method: TransformXSLEx(*nAction* as SPYDialogAction)

Description
TransformXSLEx processes the XML document via the associated XSL file. The parameter specifies whether a dialog asking for the result document name should pop up or not. See Document.AssignXSL on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLFO

See also

Method: TransformXSLFO()

Description
TransformXSLFO processes the XML document via the associated XSLFO file. See AssignXSLFO on how to place a reference to a XSLFO file into the document. You need to assign a FOP processor to Authentic Desktop before you can use this method.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TreatXBRLInconsistencies AsErrors

Property: TreatXBRLInconsistenciesAsErrors as Boolean

Description
If this is set to `true` the `Document.IsValid()` method will return `false` for XBRL instances containing inconsistencies as defined by the XBRL Specification. The default value of this property is `false`.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

UpdateViews

See also

Method: `UpdateViews()`

Description
To redraw the Enhanced Grid View and the Tree View call `UpdateViews`. This can be important after you changed the `XMLData` structure of a document. This method does not redraw the text view of Authentic Desktop.

Errors
1400 The document object is no longer valid.

UpdateXMLData

See also

Method: `UpdateXMLData()` as Boolean

Description
The XMLData tree is updated from the current view. Please note that this can fail in case of the TextView if the current XML text is not well-formed. This is not necessary if CurrentViewMode is *spyViewGrid* or *spyViewAuthentic* because these views keep the XMLData updated.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

3.2.11 Documents

See also

Properties
Count
Item

Methods
NewAuthenticFile
NewFile
NewFileFromText
OpenAuthenticFile
OpenFile
OpenURL
OpenURLDialog

Description
This object represents the set of documents currently open in Authentic Desktop. Use this object to open further documents or iterate through already opened documents.

Examples

```
' ----------------------------------------
' XMLSpy scripting environment - VBScript
' iterate through open documents
' ----------------------------------------
Dim objDocuments
Set objDocuments = Application.Documents

For Each objDoc In objDocuments
     'do something useful with your document
     objDoc.SetActiveDocument()
Next
```

```
// ----------------------------------------
// XMLSpy scripting environment - JScript
// close all open documents
// ----------------------------------------
for (var iter = new Enumerator (Application.Documents);
     ! iter.atEnd();
     iter.moveNext())
{
     // MsgBox ("Closing file " + iter.item().Name);
     iter.item().Close (true);
}
```

Count

See also

Property: Count as long

Description
Count of open documents.

Errors
1600 Invalid Documents object

1601 Invalid input parameter

Item

See also

Method: `Item` (*n* as long) as `Document`

Description
Gets the document with the index n in this collection. Index is 1-based.

Errors
1600 Invalid `Documents` object
1601 Invalid input parameter

NewAuthenticFile

See also

Method: `NewAuthenticFile` (*strSPSPath* as String, *strXMLPath* as String) as `Document`

Parameters
strSPSPath
The path to the SPS document.

strXMLPath
The new XML document name.

Return Value
The method returns the new document.

Description
NewAuthenticFile creates a new XML file and opens it in Authentic View using SPS design strSPSPath.

NewFile

See also

Method: `NewFile` (*strFile* as String, *strType* as String) as `Document`

Parameters
strFile
Full path of new file.

strType
Type of new file as string (i.e. "xml", "xsd", ...)

Return Value
Returns the new file.

Description
`NewFile` creates a new file of type `strType` (i.e. "xml"). The newly created file is also the

ActiveDocument.

NewFileFromText

See also

Method: NewFileFromText (*strText* as String, *strType* as String) as Document

Parameters

strText
The content of the new document in plain text.

strType
Type of the document to create (i.e. "xml").

Return Value

The method returns the new document.

Description

NewFileFromText creates a new document with strText as its content.

OpenAuthenticFile

See also

Method: OpenAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as Document

Parameters

strSPSPath
The path to the SPS document.

strXMLPath
The path to the XML document (can be empty).

Return Value

The method returns the new document.

Description

OpenAuthenticFile opens an XML file or database in Authentic View using SPS design strSPSPath.

OpenFile

See also

Method: OpenFile (*strPath* as String, *bDialog* as Boolean) as Document

Parameters

strPath
Path and file name of file to open.

bDialog
Show dialogs for user input.

Return Value
Returns the opened file on success.

Description
OpenFile opens the file strPath. If bDialog is TRUE, a file-dialog will be displayed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile(strFile, False)
```

OpenURL

See also

Method: OpenURL (*strURL* as String, *nURLType* as SPYURLTypes, *nLoading* as SPYLoading, *strUser* as String, *strPassword* as String) as Document

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURL opens the URL strURL.

OpenURLDialog

See also

Method: OpenURLDialog (*strURL* as String, *nURLType* as SPYURLTypes, *nLoading* as SPYLoading, *strUser* as String, *strPassword* as String) as Document

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
`OpenURLDialog` displays the "open URL" dialog to the user and presets the input fields with the given parameters.

3.2.12 DTDSchemaGeneratorDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

DTDSchemaFormat
ValueList
TypeDetection
FrequentElements
MergeAllEqualNamed
ResolveEntities
AttributeTypeDefinition
GlobalAttributes
OnlyStringEnums
MaxEnumLength
OutputPath
OutputPathDialogAction

Description
Use this object to configure the generation of a schema or DTD. The method GenerateDTDOrSchemaEx expects a DTDSchemaGeneratorDlg as parameter to configure the generation as well as the associated user interactions.

Application

Property: `Application` as `Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

AttributeTypeDefinition

Property: AttributeTypeDefinition as `SPYAttributeTypeDefinition`

Description
Specifies how attribute definitions get merged.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

DTDSchemaFormat

Property: DTDSchemaFormat as `SPYDTDSchemaFormat`

Description
Sets the schema output format to DTD, or W3C.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

FrequentElements

Property: FrequentElements as SPYFrequentElements

Description
Shall the types for all elements be defined as global? Use that value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

GlobalAttributes

Property: GlobalAttributes as Boolean

Description
Shall attributes with same name and type be resolved globally?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

MaxEnumLength

Property: MaxEnumLength as Integer

Description
Specifies the maximum number of characters allowed for enumeration names. If one value is longer than this, no enumeration will be generated.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

MergeAllEqualNamed

Property: MergeAllEqualNamed as Boolean

Description
Shall types of all elements with the same name be merged into one type?

Errors
3000 The object is no longer valid.

3001 Invalid address for the return parameter was specified.

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.

OutputPath

Property: OutputPath as String

Description
Selects the file name for the generated schema/DTD.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

OutputPathDialogAction

Property: OutputPathDialogAction as `SPYDialogAction`

Description
Defines how the sub-dialog for selecting the schema/DTD output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

ResolveEntities

Property: ResolveEntities as Boolean

Description
Shall all entities be resolved before generation starts? If yes, an info-set will be built.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

TypeDetection

Property: TypeDetection as `SPYTypeDetection`

Description
Specifies granularity of simple type detection.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

ValueList

Property: ValueList as Integer

Description
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

3.2.13 ElementList

See also

Properties
Count
Item

Methods
RemoveElement

Description
Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of ElementListItem are used.

It can hold
- a list of table names returned by a call to Application. GetDatabaseTables,
- a list of field names retuned by a call to Application. GetDatabaseImportElementList or Application. GetTextImportElementList,
- a field name filter list used in Application. ImportFromDatabase and Application. ImportFromText,
- a list of table names and counts for their rows and columns as returned by calls to GetExportElementList or
- a field name filter list used in Document. ExportToDatabase and Document. ExportToText.

Count

See also

Property: Count as long (read-only)

Description
Count of elements in this collection.

Item

See also

Method: Item(n as long) as ElementListItem

Description
Gets the element with the index n from this collection. The first item has index 1.

RemoveElement

See also

Method: RemoveElement(Index as long)

Description
RemoveElement removes the element Index from the collection. The first Item has index 1.

3.2.14 ElementListItem

See also

Properties
Name

ElementKind

FieldCount
RecordCount

Description
An element in an ElementList. Usage of its properties depends on the purpose of the element list. For details see ElementList.

ElementKind

See also

Property: ElementKind as SPYXMLDataKind

Description
Specifies if a field should be imported as XML element (data value of spyXMLDataElement) or attribute (data value of spyXMLDataAttr).

FieldCount

See also

Property: FieldCount as long (read-only)

Description
Count of fields (i.e. columns) in the table described by this element. This property is only valid after a call to DocumentGetExportElementList .

Name

See also

Property: Name as String (read-only)

Description
Name of the element. This is either the name of a table or a field, depending on the purpose of th element list.

RecordCount

See also

Property: RecordCount as long (read-only)

Description
Count of records (i.e. rows) in the table described by this element. This property is only valid

after a call to DocumentGetExportElementList.

3.2.15 ExportSettings

See also

Properties

ElementList

EntitiesToText

ExportAllElements
SubLevelLimit

FromAttributes
FromSingleSubElements
FromTextValues

CreateKeys
IndependentPrimaryKey

Namespace

ExportCompleteXML
StartFromElement

Description
ExportSettings contains options used during export of XML data to a database or text file. See Import and export of data for a general overview.

CreateKeys

See also

Property: CreateKeys as Boolean

Description
This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

ElementList

See also

Property: ElementList as ElementList

Description
Default is empty list. This list of elements defines which fields will be exported. To get the list of available fields use Document.GetExportElementList. It is possible to prevent exporting columns by removing elements from this list with ElementList.RemoveElement before passing it to Document.ExportToDatabase or Document.ExportToText.

EntitiesToText

See also

Property: EntitiesToText as Boolean

Description
Defines if XML entities should be converted to text or left as they are during export. Default is True.

ExportAllElements

See also

Property: `ExportAllElements` as Boolean

Description
If set to `true`, all elements in the document will be exported. If set to `false`, then `ExportSettings.SubLevelLimit` is used to restrict the number of sub levels to export. Default is `true`.

ExportCompleteXML

See also

Property: ExportCompleteXML as Boolean

Description
Defines whether the complete XML is exported or only the element specified by StartFromElement and its children. Default is True.

FromAttributes

See also

Property: `FromAttributes` as Boolean

Description
`SetFromAttributes` to false if no export data should be created from attributes. Default is True.

FromSingleSubElements

See also

Property: `FromSingleSubElements` as Boolean

Description
`SetFromSingleSubElements` to false if no export data should be created from elements. Default is True.

FromTextValues

See also

Property: `FromTextValues` as Boolean

Description
`SetFromTextValues` to false if no export data should be created from text values. Default is

True.

IndependentPrimaryKey

See also

Property: `IndependentPrimaryKey` as Boolean

Description
Turns creation of independent primary key counter for every element on or off. If `ExportSettings.CreateKeys` is False, this property will be ignored. Default is True.

Namespace

See also

Property: `Namespace` as `SPYExportNamespace`

Description
The default setting removes all namespace prefixes from the element names. In some database formats the colon is not a legal character. Default is `spyNoNamespace`.

StartFromElement

See also

Property: StartFromElement as String

Description
Specifies the start element for the export. This property is only considered when ExportCompleteXML is false.

SubLevelLimit

See also

Property: `SubLevelLimit` as Integer

Description
Defines the number of sub levels to include for the export. Default is 0. This property is ignored if `ExportSettings.ExportAllElements` is true.

3.2.16 FileSelectionDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Dialog properties
FullName

Acceptance or cancellation of action that caused event
DialogAction

Description
The dialog object allows you to receive information about an event and pass back information to the event handler in the same way as with a user dialog. Use the FileSelectionDlg.FullName to select or modify the file path and set the FileSelectionDlg.DialogAction property to cancel or agree with the action that caused the event.

Application

See also

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

DialogAction

Property: DialogAction as SPYDialogAction

Description
If you want your script to perform the file selection operation without any user interaction necessary, simulate user interaction by either setting the property to *spyDialogOK(0)* or *spyDialogCancel(1)*.
To allow your script to fill in the default values but let the user see and react on the dialog, use the value *spyDialogUserInput(2)*. If you receive a FileSelectionDlg object in an event handler, *spyDialogUserInput(2)* is not supported and will be interpreted as *spyDialogOK(0)*.

Errors
2400 The object is no longer valid.
2401 Invalid value for dialog action or invalid address for the return parameter was specified.

FullName

Property: `FullName` as String

Description
Access the full path of the file the gets selected by the dialog. Most events that pass a FileSelectionDlg object to you allow you modify this value and thus influence the action that caused the event (e.g. load or save to a different location).

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

3.2.17 FindInFilesDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Find
RegularExpression
Replace
DoReplace
ReplaceOnDisk
MatchWholeWord
MatchCase
SearchLocation
StartFolder
IncludeSubfolders
SearchInProjectFilesDoExternal
FileExtension
AdvancedXMLSearch
XMLElementNames
XMLElementContents
XMLAttributeNames
XMLAttributeContents
XMLComments
XMLCData
XMLPI
XMLRest
ShowResult

Description
Use this object to configure the search (or replacement) for strings in files. The method FindInFiles expects a FindInFilesDlg as parameter.

AdvancedXMLSearch

Property: AdvancedXMLSearch as Boolean

Description
Specifies if the XML search properties (XMLElementNames, XMLElementContents, XMLAttributeNames, XMLAttributeContents, XMLComments, XMLCData, XMLPI and XMLRest) are considered. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Application

Property: `Application` as `Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

DoReplace

Property: DoReplace as Boolean

Description
Specifies if the matched string is replaced by the string defined in Replace. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

FileExtension

Property: FileExtension as String

Description
Specifies the file filter of the files that should be considered during the search. Multiple file filters must be delimited with a semicolon (eg: *.xml;*.dtd;a*.xsd). Use the wildcards * and ? to define the file filter.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Find

Property: Find as String

Description
Specifies the string to search for.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

IncludeSubfolders

Property: IncludeSubfolders as Boolean

Description
Specifies if subfolders are searched too. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

MatchCase

Property: MatchCase as Boolean

Description
Specifies if the search is case sensitive. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

MatchWholeWord

Property: MatchWholeWord as Boolean

Description
Specifies whether the whole word or just a part of it must match. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

RegularExpression

Property: RegularExpression as Boolean

Description
Specifies if Find contains a regular expression. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Replace

Property: Replace as String

Description
Specifies the replacement string. The matched string is only replaced if DoReplace is set true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

ReplaceOnDisk

Property: ReplaceOnDisk as Boolean

Description
Specifies if the replacement is done directly on disk. The modified file is not opened. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

SearchInProjectFilesDoExternal

Property: SearchInProjectFilesDoExternal as Boolean

Description
Specifies if the external folders in the open project are searched, when a project search is performed. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

SearchLocation

Property: SearchLocation as SPYFindInFilesSearchLocation

Description
Specifies the location of the search. The default is spyFindInFiles_Documents.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

ShowResult

Property: ShowResult as Boolean

Description
Specifies if the result is displayed in the Find in Files output window. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

StartFolder

Property: StartFolder as String

Description
Specifies the folder where the disk search starts.

Errors

3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLAttributeContents

Property: XMLAttributeContents as Boolean

Description
Specifies if attribute contents are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLAttributeNames

Property: XMLAttributeNames as Boolean

Description
Specifies if attribute names are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLCData

Property: XMLCData as Boolean

Description
Specifies if CData tags are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLComments

Property: XMLComments as Boolean

Description
Specifies if comments are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLElementContents

Property: XMLElementContents as Boolean

Description
Specifies if element contents are searched when AdvancedXMLSearch is true. The default is

true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLElementNames

Property: XMLElementNames as Boolean

Description
Specifies if element names are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLPI

Property: XMLPI as Boolean

Description
Specifies if XML processing instructions are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLRest

Property: XMLRest as Boolean

Description
Specifies if the rest of the XML (which is not covered by the other XML search properties) is searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

3.2.18 FindInFilesResult

See also

Properties and Methods

Standard automation properties
Application
Parent

Count
Item

Path
Document

Description
This object represents a file that matched the search criteria. It contains a list of FindInFilesResultMatch objects that describe the matching position.

Application

Property: `Application` as `Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Count

Property: `Count` as long (read-only)

Description
Count of elements in this collection.

Document

Property: Path as Document (read-only)

Description
This property returns the Document object if the matched file is already open in XMLSpy.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Item

Method: `Item(n` as long) as FindInFilesResultMatch

Description
Gets the element with the index `n` from this collection. The first item has index 1.

Parent

Property: `Parent` as FindInFilesResults (read-only)

Description
Access the parent of the object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Path

Property: Path as String (read-only)

Description
Returns the path of the file that matched the search criteria.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

3.2.19 FindInFilesResultMatch

See also

Properties and Methods

Standard automation properties
Application
Parent

Line
Position
Length
LineText
Replaced

Description
Contains the exact position in the file of the matched string.

Application

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Length

Property: Length as Long (read-only)

Description
Returns the length of the matched string.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Line

Property: Line as Long (read-only)

Description
Returns the line number of the match. The line numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

LineText

Property: LineText as String (read-only)

Description
Returns the text of the line.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as FindInFilesResult (read-only)

Description
Access the parent of the object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Position

Property: Position as Long (read-only)

Description
Returns the start position of the match in the line. The position numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Replaced

Property: Replaced as Boolean (read-only)

Description
True if the matched string was replaced.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

3.2.20 FindInFilesResults

See also

Properties and Methods

Standard automation properties
Application
Parent

Count
Item

Description
This is the result of the FindInFiles method. It is a list of FindInFilesResult objects.

Application

Property: `Application` as `Application` (read-only)

Description
Access the Authentic Desktop application object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

Count

Property: `Count` as long (read-only)

Description
Count of elements in this collection.

Item

Method: `Item(n` as long) as FindInFilesResult

Description
Gets the element with the index `n` from this collection. The first item has index 1.

Parent

Property: `Parent` as Application (read-only)

Description
Access the parent of the object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

3.2.21 GenerateSampleXMLDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

NonMandatoryAttributes
RepeatCount
FillAttributesWithSampleData
FillElementsWithSampleData
ContentOfNillableElementsIsNonMandatory
TryToUseNonAbstractTypes
Optimization
SchemaOrDTDAssignment
LocalNameOfRootElement
NamespaceURIOfRootElement
OptionsDialogAction

Properties that are no longer supported
NonMandatoryElements - obsolete
TakeFirstChoice - obsolete
FillWithSampleData - obsolete

Description
Used to set the parameters for the generation of sample XML instances based on a W3C schema or DTD.

Application

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

Parent

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NonMandatoryAttributes

Property: `NonMandatoryAttributes` as Boolean

Description
If true attributes which are not mandatory are created in the sample XML instance file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NonMandatoryElements - obsolete

Property: `NonMandatoryElements` as Boolean

Description
Do no longer use this property. Use `Optimization`, instead.

Errors
0001 The property is no longer accessible.

TakeFirstChoice - obsolete

Property: `TakeFirstChoice` as Boolean

Description
Do no longer use this property.

Errors
0001 The property is no longer accessible.

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.

FillWithSampleData - obsolete

Property: `FillWithSampleData` as Boolean

Description
Do no longer access this property. Use `FillAttributesWithSampleData` and `FillElementsWithSampleData`, instead.

Errors
0001 The property is no longer accessible.

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.

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.

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.

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.

Optimization

Property: `Optimization` as `SPYSampleXMLGenerationOptimization`

Description
Specifies which elements will be generated.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

SchemaOrDTDAssignment

Property: `SchemaOrDTDAssignment` as `SPYSampleXMLGenerationSchemaOrDTDAssignment`

Description
Specifies in which way a reference to the related schema or DTD - which is this document - will be generated into the sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

LocalNameOfRootElement

Property: `LocalNameOfRootElement` as String

Description
Specifies the local name of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NamespaceURIOfRootElement

Property: `NamespaceURIOfRootElement` as String

Description
Specifies the namespace URI of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

OptionsDialogAction

Property: `OptionsDialogAction` as `SPYDialogAction`

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
2200 The object is no longer valid.
2201 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

3.2.22 GridView

See also

Methods
Deselect
Select

SetFocus

Properties
CurrentFocus

IsVisible

Description
GridView Class

Events

OnBeforeDrag

See also

Event: OnBeforeDrag() as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrag()
	'On_BeforeStartEditing=False 'to prohibit dragging
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrag()
{
	//return false;/*to prohibit dragging*/
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (4, ...)	//nEventID=4
```

Description
This event gets fired on an attempt to drag an XMLData element on the grid view. Return *false* to prevent dragging the data element to a different position.

OnBeforeDrop

See also

Event: OnBeforeDrop(*objXMLData* as XMLData) as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrop(objXMLData)
	'On_BeforeStartEditing=False 'to prohibit dropping
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeDrop(objXMLData)
{
    //return false;/*to hide dropping*/
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (5, ...)    //nEventID=5
```

Description

This event gets fired on an attempt to drop a previously dragged XMLData element on the grid view. Return *false* to prevent the data element to be moved from its original position to the drop destination position.

OnBeforeStartEditing

See also

Event: `OnBeforeStartEditing(objXMLData as XMLData, bEditingName as Boolean)as Boolean`

XMLSpy scripting environment - VBScript:

```
Function On_BeforeStartEditing(objXMLData, bEditingName)
    'On_BeforeStartEditing=False 'to hide editing the field
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeStartEditing(objXMLData, bEditingName)
{
    //return false;/*to hide 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, it its value is false, the value part will be edited.

OnEditingFinished

See also

Event: `OnEditingFinished(objXMLData as XMLData, bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:

```
Function On_EditingFinished(objXMLData, bEditingName)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_EditingFinished(objXMLData, bEditingName)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (2, ...)    //nEventId=2
```

Description
This event gets fired when the editing mode of a grid cell gets left. The parameter *bEditingName* specifies if the name part of the element has been edited.

OnFocusChanged

See also

Event: `OnFocusChanged(objXMLData as XMLData, bSetFocus as Boolean, bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:

```
Function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (3, ...)    //nEventId=3
```

Description
This event gets fired whenever a grid cell receives or looses the cursor focus. If the parameter *bEditingName* is *true*, focus of the name part of the grid element has changed. Otherwise, focus of the value part has changed.

CurrentFocus

See also

Property: `CurrentFocus` as `XMLData`

Description
Holds the XML element with the current focus. This property is read-only.

Deselect

See also

Method: `Deselect(pData as XMLData)`

Description
Deselects the element `pData` in the grid view.

IsVisible

See also

Property: `IsVisible` as Boolean

Description
True if the grid view is the active view of the document. This property is read-only.

Select

See also

Method: `Select` (*`pData`* as `XMLData`)

Description
Selects the XML element `pData` in the grid view.

SetFocus

See also

Method: `SetFocus` (*`pFocusData`* as `XMLData`)

Description
Sets the focus to the element `pFocusData` in the grid view.

3.2.23 SchemaDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OutputFile
OutputFileDialogAction
OptionsDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
MultipleOutputFiles
EmbedCSSInHTML
CreateDiagramsFolder
GenerateRelativeLinks

IncludeAll
IncludeIndex
IncludeGlobalAttributes
IncludeGlobalElements
IncludeLocalAttributes
IncludeLocalElements
IncludeGroups
IncludeComplexTypes
IncludeSimpleTypes
IncludeAttributeGroups
IncludeRedefines
IncludeReferencedSchemas

AllDetails
ShowDiagram
ShowNamespace
ShowType
ShowChildren
ShowUsedBy
ShowProperties
ShowSingleFacets
ShowPatterns
ShowEnumerations
ShowAttributes
ShowIdentityConstraints
ShowAnnotations
ShowSourceCode

Description
This object combines all options for schema document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are

initialized with the values used during the last generation of schema documentation. However, before using the object you have to set the SetOutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use FolderAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
2900 The object is no longer valid.

Application

See also

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true , to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. 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.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is PNG.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

GenerateRelativeLinks

See also

Property: GenerateRelativeLinks as Boolean

Description
Set this property to true , to create relative paths to local files. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeAll

See also

Method: `IncludeAll (i_bInclude as Boolean )`

Description
Use this method to mark or unmark all include options.

Errors
2900 The object is no longer valid.

IncludeAttributeGroups

See also

Property: `IncludeAttributeGroups` as `Boolean`

Description
Set this property to true , to include attribute groups in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeComplexTypes

See also

Property: `IncludeComplexTypes` as `Boolean`

Description
Set this property to true , to include complex types in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGlobalAttributes

See also

Property: IncludeGlobalAttributes as Boolean

Description
Set this property to true, to include global attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGlobalElements

See also

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGroups

See also

Property: IncludeGroups as Boolean

Description
Set this property to true, to include groups in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeIndex

See also

Property: IncludeIndex as Boolean

Description
Set this property to true, to include an index in the schema documentation. The property is

initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. 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.

IncludeLocalAttributes

See also

Property: `IncludeLocalAttributes` as Boolean

Description
Set this property to true, to include local attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. 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.

IncludeLocalElements

See also

Property: `IncludeLocalElements` as Boolean

Description
Set this property to true, to include local elements in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. 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.

IncludeRedefines

See also

Property: `IncludeRedefines` as Boolean

Description
Set this property to true, to include redefines in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeReferencedSchemas

See also

Property: `IncludeReferencedSchemas` as Boolean

Description
Set this property to true, to include referenced schemas in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeSimpleTypes

See also

Property: `IncludeSimpleTypes` as Boolean

Description
Set this property to true, to include simple types in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: `MultipleOutputFiles` as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: `OptionsDialogAction` as SPYDialogAction

Description

To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: `OutputFile` as String

Description

Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateSchemaDocumentation.

Errors

2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: `OutputFileDialogAction` as SPYDialogAction

Description

To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: `OutputFormat` as SPYSchemaDocumentationFormat

Description

Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is HTML.

Errors

2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowAnnotations

See also

Property: `ShowAnnotations` as `Boolean`

Description
Set this property to `true`, to show the annotations to a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowAttributes

See also

Property: `ShowAttributes` as `Boolean`

Description
Set this property to `true`, to show the type definitions attributes in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowChildren

See also

Property: `ShowChildren` as `Boolean`

Description
Set this property to true, to show the children of a type definition as links in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation(). The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowDiagram

See also

Property: ShowDiagram as Boolean

Description
Set this property to true, to show type definitions as diagrams in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation(). The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowEnumerations

See also

Property: ShowEnumerations as Boolean

Description
Set this property to true, to show the enumerations contained in a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation(). The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowIdentityConstraints

See also

Property: ShowIdentityConstraints as Boolean

Description
Set this property to true, to show a type definitions identity constraints in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation(). The default for the first run is true.

Errors
2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowNamespace

See also

Property: ShowNamespace as Boolean

Description
Set this property to true, to show the namespace of type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowPatterns

See also

Property: ShowPatterns as Boolean

Description
Set this property to true, to show the patterns of a type definition in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowProperties

See also

Property: ShowProperties as Boolean

Description
Set this property to true, to show the type definition properties in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowResult

See also

Property: `ShowResult` as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowSingleFacets

See also

Property: `ShowSingleFacets` as Boolean

Description
Set this property to true, to show the facets of a type definition in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: `ShowSourceCode` as Boolean

Description
Set this property to true, to show the XML source code for type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors

2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowType

See also

Property: `ShowType` as Boolean

Description
Set this property to true, to show the type of type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: `ShowUsedBy` as Boolean

Description
Set this property to true, to show the used-by relation for type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

SPSFile

See also

Property: `SPSFile` as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: `UseFixedDesign` as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

3.2.24 SpyProject

See also

Methods
CloseProject
SaveProject
SaveProjectAs

Properties
RootItems
ProjectFile

Description
SpyProject Class

CloseProject

See also

Declaration: `CloseProject(`*`bDiscardChanges`* as Boolean, *`bCloseFiles`* as Boolean, *`bDialog`* as Boolean)

Parameters
`bDiscardChanges`
Set `bDiscardChanges` to FALSE if you want to save the changes of the open project files and the project.

`bCloseFiles`
Set `bCloseFiles` to TRUE to close all open project files.

`bDialog`
Show dialogs for user input.

Description
`CloseProject` closes the current project.

ProjectFile

See also

Declaration: `ProjectFile` as String

Description
Path and filename of the project.

RootItems

See also

Declaration: `RootItems` as SpyProjectItems

Description
Root level of collection of project items.

SaveProject

See also

Declaration: `SaveProject`

Description
`SaveProject` saves the current project.

SaveProjectAs

See also

Declaration: `SaveProjectAs` (`strPath` as String, `bDialog` as Boolean)

Parameters
`strPath`
Full path with file name of new project file.

`bDialog`
If `bDialog` is TRUE, a file-dialog will be displayed.

Description
`SaveProjectAs` stores the project data into a new location.

3.2.25 SpyProjectItem

See also

Methods
Open

Properties
ChildItems
ParentItem
FileExtensions
ItemType
Name
Path
ValidateWith
XMLForXSLTransformation
XSLForXMLTransformation
XSLTransformationFileExtension
XSLTransformationFolder

Description
SpyProjectItem Class

ChildItems

See also

Declaration: ChildItems as SpyProjectItems

Description
If the item is a folder, ChildItems is the collection of the folder content.

FileExtensions

See also

Declaration: FileExtensions as String

Description
Used to set the file extensions if the project item is a folder.

ItemType

See also

Declaration: ItemType as SPYProjectItemTypes

Description
This property is read-only.

Name

See also

Declaration: `Name` as String

Description
Name of the project item. This property is read-only.

Open

See also

Declaration: `Open` as `Document`

Return Value
The project item opened as document.

Description
Opens the project item.

ParentItem

See also

Declaration: `ParentItem` as `SpyProjectItem`

Description
Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

Path

See also

Declaration: `Path` as String

Description
Path of project item. This property is read-only.

ValidateWith

See also

Declaration: `ValidateWith` as String

Description
Used to set the schema/DTD for validation.

XMLForXSLTransformation

See also

Declaration: `XMLForXSLTransformation` as String

Description
Used to set the XML for XSL transformation.

XSLForXMLTransformation

See also

Declaration: `XSLForXMLTransformation` as String

Description
Used to set the XSL for XML transformation.

XSLTransformationFileExtension

See also

Declaration: `XSLTransformationFileExtension` as String

Description
Used to set the file extension for XSL transformation output files.

XSLTransformationFolder

See also

Declaration: `XSLTransformationFolder` as String

Description
Used to set the destination folder for XSL transformation output files.

3.2.26 SpyProjectItems

See also

Methods
AddFile
AddFolder
AddURL
RemoveItem

Properties
Count
Item

Description
SpyProjectItems Class

AddFile

See also

Declaration: AddFile (*strPath* as String)

Parameters

strPath
Full path with file name of new project item

Description
The method adds a new file to the collection of project items.

AddFolder

See also

Declaration: AddFolder (*strName* as String)

Parameters

strName
Name of the new folder.

Description
The method AddFolder adds a folder with the name strName to the collection of project items.

AddURL

See also

Declaration: AddURL (*strURL* as String, *nURLType* as SPYURLTypes, *strUser* as String, *strPassword* as String, *bSave* as Boolean)

Description

stdURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

stdUser
Name of the user if required. Can be empty.

stdPassword
Password for authentification. Can be empty.

bSave
Save user and password information.

Description
The method adds an URL item to the project collection.

Count

See also

Declaration: `Count` as long

Description
This property gets the count of project items in the collection. The property is read-only.

Item

See also

Declaration: `Item` (*n* as long) as `SpyProjectItem`

Description
Retrieves the `n`-th element of the collection of project items. The first item has index 1.

RemoveItem

See also

Declaration: `RemoveItem` (*pItem* as `SpyProjectItem`)

Description
RemoveItem deletes the item pItem from the collection of project items.

3.2.27 TextImportExportSettings

See also

Properties for import only
ImportFile

Properties for export only
DestinationFolder
FileExtension
CommentIncluded
RemoveDelimiter
RemoveNewline

Properties for import and export
HeaderRow
FieldDelimiter
EnclosingCharacter
Encoding
EncodingByteOrder

Description
TextImportExportSettings contains options common to text import and export functions.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated text file. Default is true. This property is used only when exporting to text files.

DestinationFolder

See also

Property: `DestinationFolder` as String

Description
The property `DestinationFolder` sets the folder where the created files are saved during text export.

EnclosingCharacter

See also

Property: `EnclosingCharacter` as `SPYTextEnclosing`

Description
This property defines the character that encloses all field values for import and export. Default is `spyNoEnclosing`.

Encoding

See also

Property: Encoding as String

Description
The property Encoding sets the character encoding for the text files for importing and exporting.

EncodingByteOrder

See also

Property: EncodingByteOrder as SPYEncodingByteOrder

Description
The property EncodingByteOrder sets the byte order for Unicode characters. Default is spyNONE.

FieldDelimiter

See also

Property: FieldDelimiter as SPYTextDelimiters

Description
The property FieldDelimiter defines the delimiter between the fields during import and export. Default is spyTabulator.

FileExtension

See also

Property: FileExtension as String

Description
This property sets the file extension for files created on text export.

HeaderRow

See also

Property: HeaderRow as Boolean

Description
The property HeaderRow is used during import and export. Set HeaderRow true on import, if the first line of the text file contains the names of the columns. Set HeaderRow true on export, if the first line in the created text files should contain the name of the columns. Default value is true.

ImportFile

See also

Property: `ImportFile` as String

Description
This property is used to set the text file for import. The string has to be a full qualified path. See also Import and Export.

RemoveDelimiter

See also

Property: RemoveDelimiter as Boolean

Description
The property RemoveDelimiter defines whether characters in the text that are equal to the delimiter character are removed. Default is false. This property is used only when exporting to text files.

RemoveNewline

See also

Property: RemoveNewline as Boolean

Description
The property RemoveNewline defines whether newline characters in the text are removed. Default is false. This property is used only when exporting to text files.

3.2.28 TextView

See also

Properties and Methods

Application
Parent

LineFromPosition
PositionFromLine
LineLength
SelText
GetRangeText
ReplaceText
MoveCaret
GoToLineChar
SelectText
SelectionStart
SelectionEnd
Text
LineCount
Length

Description

Events

OnBeforeShowSuggestions

See also

Event: OnBeforeShowSuggestions() as Boolean

Description
This event gets fired before a suggestion window is shown. The Document property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeShowSuggestions()
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeShowSuggestions()
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (33, ...)  //nEventID=33
```

OnChar

See also

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 property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.
It is also possible to create a new suggestions window when none is provided by Authentic Desktop. Set the Document property Suggestions to a string array with your recommendations and return true.
This event is fired before the OnBeforeShowSuggestions event. If you prevent to show the suggestion window by returning false then OnBeforeShowSuggestions is not fired.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_Char(nChar, bExistSuggestions )
End Function
```

XMLSpy scripting environment - JScript:
```
function On_Char(nChar, bExistSuggestions )
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (35, ...)  //nEventId=35
```

Application

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

GetRangeText

Method: GetRangeText(nStart as Long, nEnd as Long) as String

Description
Returns the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

GoToLineChar

Method: GoToLineChar(nLine as Long, nChar as Long)

Description
Moves the caret to the specified line and character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Length

Property: Length as Long

Description
Returns the character count of the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineCount

Property: LineCount as Long

Description
Returns the number of lines in the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineFromPosition

Method: LineFromPosition(nCharPos as Long) as Long

Description
Returns the line number of the character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineLength

Method: LineLength(nLine as Long) as Long

Description
Returns the length of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

MoveCaret

Method: MoveCaret(nDiff as Long)

Description
Moves the caret nDiff characters.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as Document (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

PositionFromLine

Method: PositionFromLine(nLine as Long) as Long

Description
Returns the start position of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ReplaceText

Method: ReplaceText(nPosFrom as Long, nPosTill as Long, sText as String)

Description
Replaces the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectionEnd

Property: SelectionEnd as Long

Description
Returns/sets the text selection end position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectionStart

Property: SelectionStart as Long

Description
Returns/sets the text selection start position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectText

Method: SelectText(nPosFrom as Long, nPosTill as Long)

Description
Selects the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelText

Property: SelText as String

Description
Returns/sets the selected text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Text

Property: Text as String

Description
Returns/sets the document text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

3.2.29 WSDL20DocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OptionsDialogAction
OutputFile
OutputFileDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
MultipleOutputFiles
EmbedCSSInHTML
CreateDiagramsFolder

IncludeAll
IncludeBinding
IncludeImportedWSDLFiles
IncludeInterface
IncludeOverview
IncludeService
IncludeTypes

AllDetails
ShowBindingDiagram
ShowExtensibility
ShowEndpoint
ShowFault
ShowInterfaceDiagram
ShowOperation
ShowServiceDiagram
ShowSourceCode
ShowTypesDiagram
ShowUsedBy

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4300 The object is no longer valid.

Application

See also

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true , to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to DocumentGenerateWSD20LDocumentation . The default for the first run is false.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to

DocumentGenerateWSDL20Documentation. The default for the first run is PNG.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. 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.

IncludeAll

See also

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4300 The object is no longer valid.

IncludeBinding

See also

Property: `IncludeBinding` as Boolean

Description
Set this property to true , to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation . 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.

IncludeImportedWSDLFiles

See also

Property: `IncludeImportedWSDLFiles` as Boolean

Description
Set this property to true , to include imported WSDL finles in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation . The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeInterface

See also

Property: `IncludeInterface` as Boolean

Description
Set this property to true , to include interfaces in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation . The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeOverview

See also

Property: `IncludeOverview` as Boolean

Description
Set this property to true , to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation

. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeService

See also

Property: `IncludeService` as Boolean

Description
Set this property to true , to include services in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation . The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeTypes

See also

Property: `IncludeTypes` as Boolean

Description
Set this property to true , to include types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation . The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: `MultipleOutputFiles` as Boolean

Description
Set this property to true , to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation . The default for the first run is false.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: `OptionsDialogAction` as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

4300	The object is no longer valid.
4301	Invalid value has been used to set the property.
	Invalid address for the return parameter was specified.

OutputFile

See also

Property: `OutputFile` as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateWSDL20Documentation.

Errors

4300	The object is no longer valid.
4301	Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: `OutputFileDialogAction` as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

4300	The object is no longer valid.
4301	Invalid value has been used to set the property.
	Invalid address for the return parameter was specified.

OutputFormat

See also

Property: OutputFormat as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is HTML.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowBindingDiagram

See also

Property: ShowBindingDiagram as Boolean

Description
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowEndpoint

See also

Property: ShowEndpoint as Boolean

Description
Set this property to true, to show service endpoints in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowExtensibility

See also

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowFault

See also

Property: ShowFault as Boolean

Description
Set this property to true, to show faults in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowInterfaceDiagram

See also

Property: ShowInterfaceDiagram as Boolean

Description
Set this property to true, to show interface diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowOperation

See also

Property: ShowOperation as Boolean

Description
Set this property to true, to show interface and binding operations in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowServiceDiagram

See also

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors

4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowTypesDiagram

See also

Property: ShowTypesDiagram as Boolean

Description
Set this property to true, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

SPSFile

See also

Property: `SPSFile` as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: `UseFixedDesign` as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

3.2.30 WSDLDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OptionsDialogAction
OutputFile
OutputFileDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
MultipleOutputFiles
EmbedCSSInHTML
CreateDiagramsFolder

IncludeAll
IncludeBinding
IncludeImportedWSDLFiles
IncludeMessages
IncludeOverview
IncludePortType
IncludeService
IncludeTypes

AllDetails
ShowBindingDiagram
ShowExtensibility
ShowMessageParts
ShowPort
ShowPortTypeDiagram
ShowPortTypeOperations
ShowServiceDiagram
ShowSourceCode
ShowTypesDiagram
ShowUsedBy

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4300 The object is no longer valid.

Application

See also

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true , to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation . The default for the first run is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for

HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is PNG.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. 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.

IncludeAll

See also

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4300 The object is no longer valid.

IncludeBinding

See also

Property: `IncludeBinding` as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeImportedWSDLFiles

See also

Property: `IncludeImportedWSDLFiles` as Boolean

Description
Set this property to true, to include imported WSDL finles in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeMessages

See also

Property: `IncludeMessages` as Boolean

Description
Set this property to true, to include messages in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeOverview

See also

Property: `IncludeOverview` as Boolean

Description
Set this property to true, to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation.

The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludePortType

See also

Property: `IncludePortType` as Boolean

Description
Set this property to true, to include port types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeService

See also

Property: `IncludeService` as Boolean

Description
Set this property to true, to include services in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeTypes

See also

Property: `IncludeTypes` as Boolean

Description
Set this property to true, to include types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: `MultipleOutputFiles` as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. 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.

OptionsDialogAction

See also

Property: `OptionsDialogAction` as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: `OutputFile` as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateWSDLDocumentation.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: `OutputFileDialogAction` as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction

should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors

3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: `OutputFormat` as `SPYSchemaDocumentationFormat`

Description

Defines the kind of documentation that will be generated: HTML (value=`0`), MS-Word (value=`1`), or RTF (value=`2`). The property gets initialized with the value used during the last call to `DocumentGenerateWSDLDocumentation`. The default for the first run is HTML.

Errors

3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Dialogs` (read-only)

Description

Access the parent of the object.

Errors

3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowBindingDiagram

See also

Property: `ShowBindingDiagram` as `Boolean`

Description

Set this property to `true`, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to `DocumentGenerateWSDLDocumentation`. The default for the first run is true.

Errors

3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowExtensibility

See also

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation(). The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowMessageParts

See also

Property: ShowMessageParts as Boolean

Description
Set this property to true, to show message parts of messges in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation(). The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPort

See also

Property: ShowPort as Boolean

Description
Set this property to true, to show service ports in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation(). The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPortTypeDiagram

See also

Property: ShowPortTypeDiagram as Boolean

Description
Set this property to true, to show port type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPortTypeOperations

See also

Property: ShowPortTypeOperations as Boolean

Description
Set this property to true, to show port type operations in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowServiceDiagram

See also

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation(). The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation(). The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowTypesDiagram

See also

Property: ShowTypesDiagram as Boolean

Description
Set this property to true, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation(). The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: ShowUsedBy as Boolean

Description
Set this property to true , to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation . The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SPSFile

See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

3.2.31 XBRLDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OptionsDialogAction
OutputFile
OutputFileDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
EmbedCSSInHTML
CreateDiagramsFolder

IncludeAll
IncludeOverview
IncludeNamespacePrefixes
IncludeGlobalElements
IncludeDefinitionLinkroles
IncludePresentationLinkroles
IncludeCalculationLinkroles

AllDetails
ShowDiagram
ShowSubstitutiongroup
ShowItemtype
ShowBalance
ShowPeriod
ShowAbstract
ShowNillable
ShowLabels
ShowReferences
ShowLinkbaseReferences

ShortQualifiedName
ShowImportedElements

Description
This object combines all options for XBRL document generation as they are available through user interface dialog boxes in Authentic Desktop. The document generation options are initialized with the values used during the last generation of XBRL documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4400 The object is no longer valid.

Application

See also

Property: Application as Application (read-only)

Description
Access the Authentic Desktop application object.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true , to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation . The default for the first run is false.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description

This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is PNG.

Errors

4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description

Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors

4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description

Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors

4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeAll

See also

Method: IncludeAll (i_bInclude as Boolean)

Description

Use this method to mark or unmark all include options.

Errors

4400 The object is no longer valid.

IncludeCalculationLinkroles

See also

Property: IncludeCalculationLinkroles as Boolean

Description
Set this property to true, to include calculation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeDefinitionLinkroles

See also

Property: IncludeDefinitionLinkroles as Boolean

Description
Set this property to true, to include definition linkroles in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeGlobalElements

See also

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeNamespacePrefixes

See also

Property: IncludeNamespacePrefixes as Boolean

Description
Set this property to true, to include namespace prefixes in the XBRL documentation. The property is initialized with the value used during the last call to

DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeOverview

See also

Property: `IncludeOverview` as Boolean

Description
Set this property to true, to include an overview in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludePresentationLinkroles

See also

Property: `IncludePresentationLinkroles` as Boolean

Description
Set this property to true, to include presentation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: `OptionsDialogAction` as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: `OutputFile` as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateXBRLDocumentation.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: `OutputFileDialogAction` as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: `OutputFormat` as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is HTML.

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as Dialogs (read-only)

Description
Access the parent of the object.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShortQualifiedName

See also

Property: ShortQualifiedName as Boolean

Description
Set this property to true, to use short qualified names in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowAbstract

See also

Property: ShowAbstract as Boolean

Description
Set this property to true, to show abstracts in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowBalance

See also

Property: ShowBalance as Boolean

Description
Set this property to true, to show balances in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowDiagram

See also

Property: ShowDiagram as Boolean

Description
Set this property to true, to show diagrams in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowImportedElements

See also

Property: ShowImportedElements as Boolean

Description
Set this property to true, to show imported elements in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowItemtype

See also

Property: ShowItemtype as Boolean

Description
Set this property to true, to show item types in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowLabels

See also

Property: ShowLabels as Boolean

Description
Set this property to true, to show labels in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowLinkbaseReferences

See also

Property: ShowLinkbaseReferences as Boolean

Description
Set this property to true, to show linkbase references in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowNillable

See also

Property: ShowNillable as Boolean

Description
Set this property to true, to show nillable properties in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowPeriod

See also

Property: ShowPeriod as Boolean

Description
Set this property to true, to show periods in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.

4401 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowReferences

See also

Property: ShowReferences as Boolean

Description
Set this property to true, to show references in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in Authentic Desktop. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowSubstitutiongroup

See also

Property: ShowSubstitutiongroup as Boolean

Description
Set this property to true, to show substitution groups in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

SPSFile

See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

3.2.32 XMLData

See also

Properties
Kind
Name
TextValue

HasChildren
MayHaveChildren
Parent

Methods
GetFirstChild
GetNextChild
GetCurrentChild

InsertChild
InsertChildAfter
InsertChildBefore
AppendChild

EraseAllChildren
EraseChild
EraseCurrentChild

IsSameNode

CountChildren
CountChildrenKind

GetChild
GetChildAttribute
GetChildElement
GetChildKind
GetNamespacePrefixForURI

HasChildrenKind
SetTextValueXMLEncoded

Description
The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind `spyXMLDataElement` with the Name property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view switch.

Note also:

- Setting a new text value for an XML element is possible if the element does not have non-text children. A text value can be set even if the element has attributes.
- When setting a new text value for an XML element which has more than one text child, the latter will be deleted and replaced by one new text child.
- When reading the text value of an XML element which has more than one text child, only the value of the first text child will be returned.

Objects of this class represent the different atomic parts of an XML document. See the enumeration type SPYXMLDataKind for the available part types. Each part knows its children, thus forming a XMLData tree with DocumentRootElement at its top. To get the top element of the document content - ignoring the XML header - use DocumentDataRoot. For examples on how to traverse the XMLData tree see Using XMLData to modify document structure and GetNextChild.

AppendChild

See also

Declaration: AppendChild (*pNewData* as XMLData)

Description

AppendChild appends pNewData as last child to the XMLData object. See also Using XMLData.

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
```

CountChildren

See also

Declaration: CountChildren as long

Description

CountChildren gets the number of children.

Available with TypeLibrary version 1.5

Errors

1500 The XMLData object is no longer valid.

CountChildrenKind

See also

Declaration: `CountChildrenKind` (*nKind* as SPYXMLDataKind) as long

Description
`CountChildrenKind` gets the number of children of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

EraseAllChildren

See also

Declaration: `EraseAllChildren`

Description
`EraseAllChildren` deletes all associated children of the `XMLData` object.

Errors
1500 The XMLData object is no longer valid.
1900 Document must not be modified

Example
The sample erases all elements of the active document.

```
Dim objCurrentParent As XMLData

Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

EraseChild

Method: `EraseChild` (Child as XMLData)

Description
Deletes the given child node.

Errors
1500 Invalid object.
1506 Invalid input xml
1510 Invalid parameter.

EraseCurrentChild

See also

Declaration: EraseCurrentChild

Description
EraseCurrentChild deletes the current XMLData child object. Before you call EraseCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild. After deleting the current child, EraseCurrentChild increments the internal iterator of the XMLData element. No error is returned when the last child gets erased and the iterator is moved past the end of the child list. The next call to EraseCurrentChild however, will return error 1503.

Errors

1500	The XMLData object is no longer valid.
1503	No iterator is initialized for this XMLData object, or the iterator points past the last child.
1900	Document must not be modified

Examples

```
// ---------------------------------------
// 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); }
}
```

GetChild

See also

Declaration: GetChild (*position* as long) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetChild() returns a reference to the child at the given index (zero-based).

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

GetChildAttribute

Method: `GetChildAttribute` (strName as string) child as XMLData object (`NULL` on error)

Description
Retrieves the attribute having the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

GetChildElement

Method: `GetChildElement` (strName as string, nIndex as long) child as XMLData object (`NULL` on error)

Description
Retrieves the Nth child element with the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

GetChildKind

See also

Declaration: `GetChildKind` (*`position`* as long, *`nKind`* as `SPYXMLDataKind`) as `XMLData`

Return Value
Returns an XML element as `XMLData` object.

Description
`GetChildKind()` returns a reference to a child of this kind at the given index (zero-based). The position parameter is relative to the number of children of the specified kind and not to all children of the object.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

GetCurrentChild

See also

Declaration: `GetCurrentChild` as `XMLData`

Return Value
Returns an XML element as `XMLData` object.

Description
`GetCurrentChild` gets the current child. Before you call `GetCurrentChild` you must initialize an internal iterator with `XMLData.GetFirstChild`.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

GetFirstChild

See also

Declaration: `GetFirstChild` (*`nKind`* as `SPYXMLDataKind`) as `XMLData`

Return Value
Returns an XML element as `XMLData` object.

Description
`GetFirstChild` initializes a new iterator and returns the first child. Set `nKind` = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use `GetCurrentChild`, `GetNextChild` or `EraseCurrentChild`.

Errors
1500 The XMLData object is no longer valid.
1501 Invalid XMLData kind was specified.
1504 Element has no children of specified kind.
1510 Invalid address for the return parameter was specified.

Example
See the example at `XMLData.GetNextChild`.

GetNamespacePrefixForURI

Method: `GetNamespacePrefixForURI` (strURI as string) strNS as string

Description
Returns the namespace prefix of the supplied URI.

Errors
1500 Invalid object.

1510 Invalid parameter.

GetNextChild

See also

Declaration: `GetNextChild` as `XMLData`

Return Value

Returns an XML element as `XMLData` object.

Description

`GetNextChild` steps to the next child of this element. Before you call `GetNextChild` you must initialize an internal iterator with `XMLData.GetFirstChild`.

Check for the last child of the element as shown in the sample below.

Errors

1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' VBA code snippet - iterate XMLData children
' ---------------------------------------------
On Error Resume Next
Set objParent = objSpy.ActiveDocument.RootElement

'get elements of all kinds
Set objCurrentChild = objParent.GetFirstChild(-1)

Do
 'do something useful with the child

 'step to next child
 Set objCurrentChild = objParent.GetNextChild
Loop Until (Err.Number - vbObjectError = 1503)
```

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// iterate through children of XMLData
// ---------------------------------------
try
{
   var objXMLData = ... // initialize somehow
   var objChild = objXMLData.GetFirstChild(-1);

   while (true)
   {
      // do something 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);
}
```

GetTextValueXMLDecoded

Method: `GetTextValueXMLDecoded()` as string

Description
Gets the decoded text value of the XML.

Errors
1500 Invalid object.
1510 Invalid parameter.

HasChildren

See also

Declaration: `HasChildren` as Boolean

Description
The property is true if the object is the parent of other `XMLData` objects. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

HasChildrenKind

See also

Declaration: `HasChildrenKind` (*nKind* as `SPYXMLDataKind`) as Boolean

Description
The method returns true if the object is the parent of other `XMLData` objects of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

InsertChild

See also

Declaration: `InsertChild` (*pNewData* as `XMLData`)

Description
`InsertChild` inserts the new child before the current child (see also `XMLData.GetFirstChild`, `XMLData.GetNextChild` to set the current child).

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

Examples
See `Using XMLData to modify document structure`.

InsertChildAfter

Method: `InsertChildBefore` (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

InsertChildBefore

Method: `InsertChildBefore` (Node as XMLData, NewData as XMLData)

Description
Inserts a new XML node (supplied with the second parameter) before the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

IsSameNode

See also

Declaration: `IsSameNode` (*`pNodeToCompare`* as `XMLData`) as Boolean

Description

Returns true if `pNodeToCompare` references the same node as the object itself.

Errors

1500 The XMLData object is no longer valid.
1506 Invalid address for the return parameter was specified.

Kind

See also

Declaration: `Kind` as `SPYXMLDataKind`

Description
Kind of this `XMLData` object. This property is read-only.

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

MayHaveChildren

See also

Declaration: `MayHaveChildren` as Boolean

Description
Indicates whether it is allowed to add children to this `XMLData` object.
This property is read-only.

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Name

See also

Declaration: `Name` as String

Description
Used to modify and to get the name of the `XMLData` object.

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Parent

See also

Declaration: `Parent` as `XMLData`

Return value
Parent as `XMLData` object. Nothing (or NULL) if there is no parent element.

Description
Parent of this element. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

SetTextValueXMLEncoded

Method: `SetTextValueXMLEncoded` (*`strVal`* as `String`)

Description
Sets the encoded text value of the XML.

Errors
1500 Invalid object.
1513 Modification not allowed.

TextValue

See also

Declaration: `TextValue` as String

Description
Used to modify and to get the text value of this `XMLData` object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Examples
See `Using XMLData to modify document structure` .

3.3 Interfaces (obsolete)

Interfaces contained in this book are obsolete. It is recommended to migrate your applications to the new interfaces. See the different properties and methods in this book for migration hints.

3.3.1 AuthenticEvent (obsolete)

Superseded by AuthenticView and AuthenticRange

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 interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties

altKey
altLeft
ctrlKey
ctrlLeft
shiftKey
shiftLeft

keyCode
repeat

button

clientX
clientY

dataTransfer

srcElement
fromElement

propertyName

cancelBubble
returnValue

type

Description

DocEditEvent interface.

altKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//       MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long
i_nKeyCode, SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

See also

Declaration: `altKey` as Boolean

Description
True if the right ALT key is pressed.

altLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//       MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyDown (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

See also

Declaration: `altLeft` as Boolean

Description
True if the left ALT key is pressed.

button (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditButtonDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.button == 1)
//         MsgBox ("left mouse button down detected");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyLeftButtonDownMask)
        MsgBox ("left mouse button down detected");
}
```

See also

Declaration: `button` as long

Description
Specifies which mouse button is pressed:

0	No button is pressed.
1	Left button is pressed.
2	Right button is pressed.
3	Left and right buttons are both pressed.
4	Middle button is pressed.
5	Left and middle buttons both are pressed.
6	Right and middle buttons are both pressed.
7	All three buttons are pressed.

cancelBubble (obsolete)

Superseded by the boolean return value of following event handler functions

AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

Returning *true* from an event handler function signals that the event has beend handled and normal event handling should be aborted.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//        // cancel key processing, swallow spaces :-)
//        Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long
i_nKeyCode, SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

See also

Declaration: `cancelBubble` as Boolean

Description
Set `cancelBubble` to TRUE if the default event handler should not be called.

clientX (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " +
Application.ActiveDocument.DocEditView.event.clientX +
//             "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyMouseMoveMask)
        MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

See also

Declaration: `clientX` as long

Description
X value of the current mouse position in client coordinates.

clientY (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " +
Application.ActiveDocument.DocEditView.event.clientX +
//             "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyMouseMoveMask)
        MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

See also

Declaration: `clientY` as long

Description
Y value of the current mouse position in client coordinates.

ctrlKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//       MsgBox ("control key is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyCtrlKeyMask)
        MsgBox ("control key is down");
}
```

See also

Declaration: `ctrlKey` as Boolean

Description
True if the right CTRL key is pressed.

ctrlLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//       MsgBox ("control key is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyCtrlKeyMask)
        MsgBox ("control key is down");
}
```

See also

Declaration: `ctrlLeft` as Boolean

Description
True if the left CTRL key is pressed.

dataTransfer (obsolete)

Superseded by parameters to
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDrop ()
// {
//     if ( Application. ActiveDocument. DocEditView. event. dataTransfer != 
null)
//         if (! 
Application. ActiveDocument. DocEditView. event. dataTransfer. ownDrag)
//         {
//            // cancel key processing, don' t drop foreign objects :-)
//            Application. ActiveDocument. DocEditView. event. cancelBubble = 
true;
//         }
// }
// use now:
function On_AuthenticView_BeforeDrop (long i_nXPos, long i_nYPos,
                                      IAuthenticRange *i_ipRange,
                                      IAuthenticDataTransfer *i_ipData)
{
    if ( i_ipRange != null)
        if (! i_ipRange. ownDrag)
            return true;  // cancel key processing, don' t drop foreign 
objects :-)

    return false;
}
```

See also

Declaration: `dataTransfer` as Variant

Description
Property dataTransfer.

fromElement (obsolete)

Not supported

See also

Declaration: `fromElement` as Variant (not supported)

Description
Currently no event sets this property.

keyCode (obsolete)

Superseded by a parameter to AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//        // cancel key processing, swallow spaces :-)
//        Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long
i_nKeyCode, SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

See also

Declaration: `keyCode` as long

Description
Keycode of the currently pressed key. This property is read-write.

propertyName (obsolete)

Not supported

See also

Declaration: `propertyName` as String (not supported)

Description
Currently no event sets this property.

repeat (obsolete)

Not supported

See also

Declaration: `repeat` as Boolean (not supported)

Description
True if the `onkeydown` event is repeated.

returnValue (obsolete)

No longer supported

See also

Declaration: `returnValue` as Variant

Description
Use `returnValue` to set a return value for your event handler.

shiftKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//       MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
    if (i_eMouseEvent & spyShiftKeyMask)
        MsgBox ("shift key is down");
}
```

See also

Declaration: `shiftKey` as Boolean

Description
True if the right SHIFT key is pressed.

shiftLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//       MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
    if (i_eMouseEvent & spyShiftKeyMask)
        MsgBox ("shift key is down");
}
```

See also

Declaration: `shiftLeft` as Boolean

Description
True if the left SHIFT key is pressed.

srcElement (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

With the new event handler function, a range object selecting this element is provided instead of the XMLData element currently below the mouse cursor.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     var objEvent = Application.ActiveDocument.DocEditView.event;
//     if (objEvent.srcElement != null)
//         MsgBox ("moving over " + objEvent.srcElement.Parent.Name);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if ((i_eMouseEvent & spyMouseMoveMask) &&
        (i_ipRange != null))
        MsgBox ("moving over " + i_ipRange.FirstXMLData.Parent.Name);
}
```

See also

Declaration: `srcElement` as Variant

Description
Element which fires the current event. This is usually an `XMLData` object.

type (obsolete)

Not supported

See also

Declaration: `type` as String (not supported)

Description
Currently no event sets this property.

3.3.2 AuthenticSelection (obsolete)

Superseded by AuthenticRange

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 interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties
Start
StartTextPosition
End
EndTextPosition

End (obsolete)

Superseded by AuthenticRange.LastXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData =
Application.ActiveDocument.DocEditView.CurrentSelection.End;
// use now:
var objXMLData =
Application.ActiveDocument.AuthenticView.Selection.LastXMLData;
```

See also

Declaration: `End` as `XMLData`

Description
XML element where the current selection ends.

EndTextPosition (obsolete)

Superseded by AuthenticRange.LastXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset =
Application.ActiveDocument.DocEditView.CurrentSelection.EndTextPosition;
// use now:
var nOffset =
Application.ActiveDocument.AuthenticView.Selection.LastXMLDataOffset;
```

See also

Declaration: `EndTextPosition` as long

Description
Position in `DocEditSelection.End.TextValue` where the selection ends.

Start (obsolete)

Superseded by AuthenticRange.FirstXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData =
Application.ActiveDocument.DocEditView.CurrentSelection.Start;
// use now:
var objXMLData =
Application.ActiveDocument.AuthenticView.Selection.FirstXMLData;
```

See also

Declaration: `Start` as `XMLData`

Description
XML element where the current selection starts.

StartTextPosition (obsolete)

Superseded by AuthenticRange.FirstXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset =
Application.ActiveDocument.DocEditView.CurrentSelection.StartTextPosition;
// use now:
var nOffset =
Application.ActiveDocument.AuthenticView.Selection.FirstXMLDataOffset;
```

See also

Declaration: `StartTextPosition` as long

Description
Position in `DocEditSelection.Start.TextValue` where the selection starts.

3.3.3 OldAuthentictView (obsolete)

Superseded by AuthenticView and AuthenticRange

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. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

LoadXML
SaveXML

EditClear
EditCopy
EditCut
EditPaste
EditRedo
EditSelectAll
EditUndo

RowAppend
RowDelete
RowDuplicate
RowInsert
RowMoveDown
RowMoveUp

ApplyTextState
IsTextStateApplied
IsTextStateEnabled

MarkUpView

SelectionSet
SelectionMoveTabOrder

GetNextVisible
GetPreviousVisible

GetAllowedElements

Properties

CurrentSelection

event

XMLRoot

IsEditClearEnabled
IsEditCopyEnabled
IsEditCutEnabled
IsEditPasteEnabled
IsEditRedoEnabled
IsEditUndoEnabled

IsRowAppendEnabled
IsRowDeleteEnabled
IsRowDuplicateEnabled
IsRowInsertEnabled
IsRowMoveDownEnabled
IsRowMoveUpEnabled

Description
Interface for Authentic View.

ApplyTextState (obsolete)

Superseded by AuthenticRange.PerformAction

Use spyAuthenticApply for the eAction parameter. The PerformAction method allows to apply text state attributes to any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.ApplyTextState ("bold");
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.PerformAction
(spyAuthenticApply, "bold"))
    MsgBox ("Error: can't set current selection to bold");
```

See also

Declaration: `ApplyTextState` (*`elementName`* as String)

Description
Applies or removes the text state defined by the parameter `elementName`. Common examples for the parameter elementName would be strong and italic.

In an XML document there are segments of data, which may contain sub-elements. For example consider the following HTML:

```
<b>fragment</b>
```

The HTML tag `<b>` will cause the word fragment to be bold. However, this only happens because the HTML parser knows that the tag `<b>` is bold. With XML there is much more flexibility. It is possible to define any XML tag to do anything you desire. The point is that it is possible to apply a Text state using XML. But the Text state that is applied must be part of the schema. For example in the `OrgChart.xml`, `OrgChart.sps`, `OrgChart.xsd` example the tag `<strong>` is the same as bold. And to apply bold the method `ApplyTextState()` is called. But like the row and edit operations it is necessary to test if it is possible to apply the text state.

See also `IsTextStateEnabled` and `IsTextStateApplied`.

CurrentSelection (obsolete)

Superseded by AuthenticView.Selection

The returned AuthenticRange object supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// var objDocEditSel =
Application.ActiveDocument.DocEditView.CurrentSelection;
// use now:
    var objRange = Application.ActiveDocument.AuthenticView.Selection;
```

See also

Declaration: `CurrentSelection` as `DocEditSelection`

Description
The property provides access to the current selection in the Authentic View.

EditClear (obsolete)

Superseded by AuthenticRange.Delete

The Delete method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditClear();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Delete())
    MsgBox ("Error: can't delete current selection");
```

See also

Declaration: `EditClear`

Description
Deletes the current selection.

EditCopy (obsolete)

Superseded by AuthenticRange.Copy

The Copy method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCopy();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Copy())
    MsgBox ("Error: can't copy current selection");
```

See also

Declaration: `EditCopy`

Description
Copies the current selection to the clipboard.

EditCut (obsolete)

Superseded by AuthenticRange.Cut

The Cut method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCut();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Cut())
    MsgBox ("Error: can't cut out current selection");
```

See also

Declaration: `EditCut`

Description
Cuts the current selection from the document and copies it to the clipboard.

EditPaste (obsolete)

Superseded by AuthenticRange.Paste

The Paste method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditPaste();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Paste())
    MsgBox ("Error: can't paste to current selection");
```

See also

Declaration: `EditPaste`

Description
Pastes the content from the clipboard into the document.

EditRedo (obsolete)

Superseded by AuthenticView.Redo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Redo())
    MsgBox ("Error: no redo step available");
```

See also

Declaration: `EditRedo`

Description
Redo the last undo step.

EditSelectAll (obsolete)

Superseded by AuthenticView.WholeDocument and AuthenticRange.Select

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditSelectAll();
// use now:
Application.ActiveDocument.AuthenticView.WholeDocument.Select();
```

See also

Declaration: `EditSelectAll`

Description
The method selects the complete document.

EditUndo (obsolete)

Superseded by AuthenticView.Undo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Undo())
    MsgBox ("Error: no undo step available");
```

See also

Declaration: `EditUndo`

Description
Undo the last action.

event (obsolete)

Superseded by parameters to AuthenticView events.

See also

Declaration: `event` as `DocEditEvent`

Description
The event property holds a `DocEditEvent` object which contains information about the current event.

GetAllowedElements (obsolete)

Superseded by AuthenticRange.CanPerformActionWith

AuthenticRange now supports all functionality of the 'elements' entry helper. Besides querying the elements that can be inserted, appended, etc., you can invoke the action as well. See AuthenticRange.PerformAction for more information.

```
// ----- javascript sample -----
// instead of:
// var arrElements = New Array();
// var objDocEditView = Application.ActiveDocument.DocEditView;
// var objStartElement = objDocEditView.CurrentSelection.Start;
// var objEndElement = objDocEditView.CurrentSelection.End;
// objDocEditView.GetAllowedElements(k_ActionInsertBefore, objStartElement,
objEndElement, arrElements);
// use now:
var arrElements = New Array();
Application.ActiveDocument.AuthenticView.Selection.CanPerformActionWith
(spyAuthenticInsertBefore, arrElements);
```

See also

Declaration: `GetAllowedElements` (*`nAction`* as `SpyAuthenticElementActions`, *`pStartElement`* as `XMLData`, *`pEndElement`* as `XMLData`, *`pElements`* as Variant)

Description
`GetAllowedElements()` returns the allowed elements for the various actions specified by `nAction`.

JavaScript example:

```
Function GetAllowed()
{
 var objView = Application.ActiveDocument.DocEditView;
```

```
 var arrElements = New Array(1);

 var objStart = objView.CurrentSelection.Start;
 var objEnd = objView.CurrentSelection.End;

 var strText;
 strText = "valid elements at current selection:\n\n";

 For(var i = 1;i <= 4;i++){
  objPlugIn.GetAllowedElements(i,objStart,objEnd,arrElements);
  strText = strText + ListArray(arrElements) + "------------------\n";
 }

 Return strText;
}

Function ListArray(arrIn)
{
 var strText = "";

 If(TypeOf(arrIn) == "object"){
  For(var i = 0;i <= (arrIn.length - 1);i++)
   strText = strText + arrIn[i] + "\n";
 }

 Return strText;
}
```

VBScript example:

```
Sub DisplayAllowed
 Dim objView
 Set objView = Application.ActiveDocument.DocEditView

 Dim arrElements()

 Dim objStart
 Dim objEnd
 Set objStart = objView.CurrentSelection.Start
 Set objEnd = objView.CurrentSelection.End

 Dim strText
 strText = "valid elements at current selection:" & chr(13) & chr(13)

 Dim i

 For i = 1 To 4
  objView.GetAllowedElements i,objStart,objEnd,arrElements
  strText = strText & ListArray(arrElements) & "---------------" & chr(13)
 Next

 msgbox strText
End Sub

Function ListArray(arrIn)
 Dim strText

 If IsArray(arrIn) Then
  Dim i

  For i = 0 To UBound(arrIn)
   strText = strText & arrIn(i) & chr(13)
  Next
 End If

 ListArray = strText
```

```
End Function
```

GetNextVisible (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the next XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData =
Application.ActiveDocument.DocEditView.GetNextVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0,
objXMLData, -1);
// use now:
var objRange = ...
try
    { objRange.SelectNext (spyAuthenticTag).Select(); }
catch (err)
{
    if ((err.number & 0xffff) == 2003)
        MsgBox ("end of document reached");
    else
        throw (err);
}
```

See also

Declaration: `GetNextVisible` (*pElement* as XMLData) as XMLData

Description
The method gets the next visible XML element in the document.

GetPreviousVisible (obsolete)

Superseded by AuthenticRange.SelectPrevious

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the previous XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData =
Application.ActiveDocument.DocEditView.GetPreviousVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0,
objXMLData, -1);
// use now:
var objRange = ...
try
    { objRange.SelectPrevious (spyAuthenticTag).Select(); }
catch (err)
{
    if ((err.number & 0xffff) == 2004)
        MsgBox ("begin of document reached");
    else
        throw (err);
}
```

See also

Declaration: `GetPreviousVisible` (*pElement* as XMLData) as XMLData

Description
The method gets the previous visible XML element in the document.

IsEditClearEnabled (obsolete)

Superseded by AuthenticRange.IsDeleteEnabled

The IsDeleteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditClearEnabled)
//    Application.ActiveDocument.DocEditView.EditClear();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsDeleteEnabled)
    objCurrSelection.Delete();
```

See also

Declaration: `IsEditClearEnabled` as Boolean

Description
True if `EditClear` is possible. See also `Editing operations`.

IsEditCopyEnabled (obsolete)

Superseded by AuthenticRange.IsCopyEnabled

The IsCopyEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCopyEnabled)
//    Application.ActiveDocument.DocEditView.EditCopy();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCopyEnabled)
    objCurrSelection.Copy();
```

See also

Declaration: `IsEditCopyEnabled` as Boolean

Description
True if copy to clipboard is possible. See also `EditCopy` and `Editing operations`.

IsEditCutEnabled (obsolete)

Superseded by AuthenticRange.IsCutEnabled

The IsCutEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCutEnabled)
//    Application.ActiveDocument.DocEditView.EditCut();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCutEnabled)
    objCurrSelection.Cut();
```

See also

Declaration: `IsEditCutEnabled` as Boolean

Description
True if `EditCut` is currently possible. See also `Editing operations`.

IsEditPasteEnabled (obsolete)

Superseded by AuthenticRange.IsPasteEnabled

The IsPasteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditPasteEnabled)
//    Application.ActiveDocument.DocEditView.EditPaste();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsPasteEnabled)
    objCurrSelection.Paste();
```

See also

Declaration: `IsEditPasteEnabled` as Boolean

Description
True if `EditPaste` is possible. See also `Editing operations`.

IsEditRedoEnabled (obsolete)

Superseded by AuthenticView.IsRedoEnabled

```
// ----- javascript sample -----
// instead of:
// if ( Application. ActiveDocument. DocEditView. IsEditRedoEnabled)
//    Application. ActiveDocument. DocEditView. EditRedo( );
// use now:
if ( Application. ActiveDocument. AuthenticView. IsRedoEnabled)
    Application. ActiveDocument. AuthenticView. Redo( );
```

See also

Declaration: `IsEditRedoEnabled` as Boolean

Description
True if `EditRedo` is currently possible. See also `Editing operations`.

IsEditUndoEnabled (obsolete)

Superseded by AuthenticView.IsUndoEnabled

```
// ----- javascript sample -----
// instead of:
// if ( Application. ActiveDocument. DocEditView. IsEditUndoEnabled)
//    Application. ActiveDocument. DocEditView. EditUndo( );
// use now:
if ( Application. ActiveDocument. AuthenticView. IsUndoEnabled)
    Application. ActiveDocument. AuthenticView. Undo( );
```

See also

Declaration: `IsEditUndoEnabled` as Boolean

Description
True if `EditUndo` is possible. See also `Editing operations`.

IsRowAppendEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowAppendEnabled)
//    Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.AppendRow();
```

See also

Declaration: `IsRowAppendEnabled` as Boolean

Description
True if `RowAppend` is possible. See also `Row operations`.

IsRowDeleteEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDeleteEnabled)
//    Application.ActiveDocument.DocEditView.Rowdelete();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.DeleteRow();
```

See also

Declaration: `IsRowDeleteEnabled` as Boolean

Description
True if `RowDelete` is possible. See also `Row operations`.

IsRowDuplicateEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDuplicateEnabled)
//    Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.DuplicateRow();
```

See also

Declaration: `IsRowDuplicateEnabled` as Boolean

Description
True if `RowDuplicate` is currently possible. See also `Row operations`.

IsRowInsertEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowInsertEnabled)
//    Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.InsertRow();
```

See also

Declaration: `IsRowInsertEnabled` as Boolean

Description
True if `RowInsert` is possible. See also `Row operations`.

IsRowMoveDownEnabled (obsolete)

Superseded by AuthenticRange.IsLastRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.OldAuthenticView.IsRowMoveDownEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsLastRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowDown();
```

See also

Declaration: `IsRowMoveDownEnabled` as Boolean

Description
True if `RowMoveDown` is currently possible. See also `Row operations`.

IsRowMoveUpEnabled (obsolete)

Superseded by AuthenticRange.IsFirstRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowMoveUpEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveUp();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsFirstRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowUp();
```

See also

Declaration: `IsRowMoveUpEnabled` as Boolean

Description
True if `RowMoveUp` is possible. See also `Row operations`.

IsTextStateApplied (obsolete)

Superseded by AuthenticRange.IsTextStateApplied

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateApplied ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsTextStateApplied (
"bold"))
    MsgBox ("bold on");
else
    MsgBox ("bold off");
```

See also

Declaration: `IsTextStateApplied` (*elementName* as String) as Boolean

Description
Checks to see if the it the text state has already been applied. Common examples for the parameter `elementName` would be strong and italic.

IsTextStateEnabled (obsolete)

Superseded by AuthenticRange.CanPerformAction

Use spyAuthenticApply for the eAction parameter. The CanPerformAction method allows to operate on any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateEnabled ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.CanPerformAction
(spyAuthenticApply, "bold"))
    ... // e.g. enable 'bold' button
```

See also

Declaration: `IsTextStateEnabled` (*i_strElementName* as String) as Boolean

Description
Checks to see if it is possible to apply a text state. Common examples for the parameter `elementName` would be strong and italic.

LoadXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property AsXMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.LoadXML (strDocAsXMLString);
// use now:
try
    { Application.ActiveDocument.AuthenticView.AsXMLString =
strDocAsXMLString; }
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

See also

Declaration: `LoadXML` (*`xmlString`* as String)

Description
Loads the current XML document with the XML string applied. The new content is displayed immediately.
The *xmlString* parameter must begin with the XML declaration, e.g.,
objPlugIn.LoadXML("<?xml version='1.0' encoding='UTF-8'?><root></root>");

MarkUpView (obsolete)

Superseded by AuthenticView.MarkupVisibility

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.MarkuUpView = 2;
// use now:
Application.ActiveDocument.AuthenticView.MarkupVisibility =
spyAuthenticMarkupLarge;
```

See also

Declaration: `MarkUpView` (*`kind`* as long)

Description
By default the document displayed is using HTML techniques. But sometimes it is desirable to show the editing tags. Using this method it is possible to display three different types of markup tags:

0	hide the markup tags
2	show the large markup tags

3 show the mixed markup tags.

RowAppend (obsolete)

Superseded by AuthenticRange.AppendRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.AppendRow())
    MsgBox ("Error: can't append row");
```

See also

Declaration: `RowAppend`

Description
Appends a row at the current position.

See also Row operations.

RowDelete (obsolete)

Superseded by AuthenticRange.DeleteRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDelete();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DeleteRow())
    MsgBox ("Error: can't delete row");
```

See also

Declaration: `RowDelete`

Description
Deletes the currently selected row(s).

See also Row operations.

RowDuplicate (obsolete)

Superseded by AuthenticRange.DuplicateRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DuplicateRow())
    MsgBox ("Error: can't duplicate row");
```

See also

Declaration: `RowDuplicate`

Description
The method duplicates the currently selected rows.

See also Row operations.

RowInsert (obsolete)

Superseded by AuthenticRange.InsertRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.InsertRow())
    MsgBox ("Error: can't insert row");
```

See also

Declaration: `RowInsert`

Description
Inserts a new row immediately above the current selection.

See also Row operations.

RowMoveDown (obsolete)

Superseded by AuthenticRange.MoveRowDown

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowDown())
    MsgBox ("Error: can't move row down");
```

See also

Declaration: `RowMoveDown`

Description
Moves the current row one position down.

See also Row operations.

RowMoveUp (obsolete)

Superseded by AuthenticRange.MoveRowUp

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowUp())
    MsgBox ("Error: can't move row up");
```

See also

Declaration: `RowMoveUp`

Description
Moves the current row one position up.

See also Row operations.

SaveXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property XMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// var strDocAsXMLString = Application.ActiveDocument.DocEditView.SaveXML();
// use now:
try
{
    var strDocAsXMLString =
Application.ActiveDocument.AuthenticView.AsXMLString;
    ... // do something here
}
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

See also

Declaration: `SaveXML` as String

Return Value
XML structure as string

Description
Saves the current XML data to a string that is returned to the caller.

SelectionMoveTabOrder (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the next paragraph is just one of them, and navigation is not necessarily bound to the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.SelectionMoveTabOrder(true, true);
// use now:
Application.ActiveDocument.AuthenticView.Selection.SelectNext
(spyAuthenticParagraph).Select();
// to append a row to a table use AuthenticRange.AppendRow
```

See also

Declaration: `SelectionMoveTabOrder` (*`bForward`* as Boolean, *`bTag`* as Boolean)

Description

`SelectionMoveTabOrder()` moves the current selection forwards or backwards.

If `bTag` is false and the current selection is at the last cell of a table a new line will be added.

SelectionSet (obsolete)

Superseded by AuthenticRange.FirstXMLData and related properties

AuthenticRange supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// if (! Application.ActiveDocument.DocEditView.SelectionSet(varXMLData1, 0,
varXMLData2, -1))
//    MsgBox ("Error: invalid data position");
// use now:
try
{
    var objSelection = Application.ActiveDocument.AuthenticView.Selection;
    objSelection.FirstXMLData = varXMLData1;
    objSelection.FirstXMLdataOffset = 0;
    objSelection.LastXMLData = varXMLData2;
    objSelection.LastXMLDataOffset = -1;
    objSelection.Select();
}
catch (err)
    { MsgBox ("Error: invalid data position"); }
// to select all text between varXMLData1 and varXMLdata2, inclusive
```

See also

Declaration: `SelectionSet` (*`pStartElement`* as `XMLData`, *`nStartPos`* as long, *`pEndElement`* as `XMLData`, *`nEndPos`* as long) as Boolean

Description

Use `SelectionSet()` to set a new selection in the Authentic View. Its possible to set `pEndElement` to null (nothing) if the selection should be just over one (`pStartElement`) XML element.

XMLRoot (obsolete)

Superseded by AuthenticView.XMLDataRoot

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.XMLRoot;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.XMLDataRoot;
```

See also

Declaration: `XMLRoot` as `XMLData`

Description
`XMLRoot` is the parent element of the currently displayed XML structure. Using the `XMLData` interface you have full access to the complete content of the file.

See also Using XMLData for more information.

3.4 Enumerations

This is a list of all enumerations used by the Authentic Desktop API. If your scripting environment does not support enumerations use the number-values instead.

3.4.1 ENUMApplicationStatus

Description
Enumeration to specify the current Application status.

Possible values:

eApplicationRunning	= 0
eApplicationAfterLicenseCheck	= 1
eApplicationBeforeLicenseCheck	= 2
eApplicationConcurrentLicenseCheckFailed	= 3
eApplicationProcessingCommandLine	= 4

3.4.2 SPYAttributeTypeDefinition

Description
Attribute type definition that can be selected for generation of Sample XML.
This type is used with the method GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:

spyMergedGlobal	= 0
spyDistinctGlobal	= 1
spyLocal	= 2

3.4.3 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange objects.

Possible values:

spyAuthenticInsertAt	= 0
spyAuthenticApply	= 1
spyAuthenticClearSurr	= 2
spyAuthenticAppend	= 3
spyAuthenticInsertBefore	= 4
spyAuthenticRemove	= 5

3.4.4 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange objects.

Possible values:

spyAuthenticDocumentBegin = 0
spyAuthenticDocumentEnd = 1
spyAuthenticRangeBegin = 2
spyAuthenticRangeEnd = 3

3.4.5 SPYAuthenticElementActions

Description
Actions that can be used with GetAllowedElements (superseded by `AuthenticRange.CanPerformActionWith`).

Possible values:

k_ActionInsertAt	= 0
k_ActionApply	= 1
k_ActionClearSurr	= 2
k_ActionAppend	= 3
k_ActionInsertBefore	= 4
k_ActionRemove	= 5

3.4.6 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange and AuthenticView objects.

Possible values:

spyAuthenticChar	= 0
spyAuthenticWord	= 1
spyAuthenticLine	= 3
spyAuthenticParagraph	= 4
spyAuthenticTag	= 6
spyAuthenticDocument	= 8
spyAuthenticTable	= 9
spyAuthenticTableRow	= 10
spyAuthenticTableColumn	= 11

3.4.7 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility.

Possible values:

spyAuthenticMarkupHidden	= 0
spyAuthenticMarkupSmall	= 1
spyAuthenticMarkupLarge	= 2
spyAuthenticMarkupMixed	= 3

3.4.8 SPYAuthenticToolbarButtonState

Description
Authentic toolbar button states are given by the following enumeration:

Possible values:

`authenticToolbarButtonDefault`	= 0
`authenticToolbarButtonEnabled`	= 1
`authenticToolbarButtonDisabled`	= 2

3.4.9 SPYDatabaseKind

Description
Values to select different kinds of databases for import. See DatabaseConnection.DatabaseKind for its use.

Possible values:

spyDB_Access	= 0
spyDB_SQLServer	= 1
spyDB_Oracle	= 2
spyDB_Sybase	= 3
spyDB_MySQL	= 4
spyDB_DB2	= 5
spyDB_Other	= 6
spyDB_Unspecified	= 7
spyDB_PostgreSQL	= 8
spyDB_iSeries	= 9

3.4.10 SPYDialogAction

Description
Values to simulate different interactions on dialogs. See Dialogs for all dialogs available.

Possible values:

spyDialogOK	= 0	//simulate click on OK button
spyDialogCancel	= 1	//simulate click on Cancel button
spyDialogUserInput	= 2	// show dialog and allow user interaction

3.4.11 SPYDOMType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

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.

3.4.12 SPYDTDSchemaFormat

Description
Enumeration to identify the different schema formats.

Possible values:

spyDTD	= 0
spyW3C	= 1

3.4.13 SPYEncodingByteOrder

Description
Enumeration values to specify encoding byte ordering for text import and export.

Possible values:

spyNONE	= 0
spyLITTLE_ENDIAN	= 1
spyBIG_ENDIAN	= 2

3.4.14 SPYExportNamespace

Description
Enumeration type to configure handling of namespace identifiers during export.

Possible values:

spyNoNamespace	= 0
spyReplaceColonWithUnderscore	= 1

3.4.15 SPYFindInFilesSearchLocation

Description
The different locations where a search can be performed. This type is used with the FindInFilesDlg dialog.

Possible values:

spyFindInFiles_Documents	= 0
spyFindInFiles_Project	= 1
spyFindInFiles_Folder	= 2

3.4.16 SPYFrequentElements

Description
Enumeration value to parameterize schema generation.

Possible values:

spyGlobalElements	= 0
spyGlobalComplexType	= 1

3.4.17 SPYImageKind

Description
Enumeration values to parameterize image type of the generated documentation. These values are used in SchemaDocumentationDialog.DiagramFormat and WSDLDocumentationDlg.DiagramFormat.

Possible values:

spyImageType_PNG	= 0
spyImageType_EMF	= 1

3.4.18 SPYImportColumnsType

Description
Enumeration to specify different Import columns types.

Possible values:

spyImportColumns_Element	= 0
spyImportColumns_Attribute	= 1

3.4.19 SPYKeyEvent

Description
Enumeration type to identify the different key events. These events correspond with the equally named windows messages.

Possible values:

spyKeyDown	= 0
spyKeyUp	= 1
spyKeyPressed	= 2

3.4.20 SPYKeyStatus

Description
Enumeration type to identify the key status.

Possible values:

spyLeftShiftKeyMask	= 1
spyRightShiftKeyMask	= 2
spyLeftCtrlKeyMask	= 4
spyRightCtrlKeyMask	= 8
spyLeftAltKeyMask	= 16
spyRightAltKeyMask	= 32

3.4.21 SPYLibType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyLibType_static	= 0
spyLibType_dll	= 1

3.4.22 SPYLoading

Description
Enumeration values to define loading behaviour of URL files.

Possible values:

spyUseCacheProxy	= 0
spyReload	= 1

3.4.23 SPYMouseEvent

Description
Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather then directly comparing with them.

Examples

```
' to check for ctrl-leftbutton-down in VB
If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
      ' react on ctrl-leftbutton-down
End If

' to check for double-click with any button in VBScript
If (((i_eMouseEvent And spyDoubleClickMask) <> 0) Then
      ' react on double-click
End If
```

Possible values:

spyNoButtonMask	= 0	
spyMouseMoveMask	= 1	
spyLeftButtonMask	= 2	
spyMiddleButtonMask	= 4	
spyRightButtonMask	= 8	
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

3.4.24 SPYNumberDateTimeFormat

Description
Enumeration value to configure database connections.

Possible values:

spySystemLocale	= 0
spySchemaCompatible	= 1

3.4.25 SPYProgrammingLanguage

Description
Enumeration values to select the programming language for code generation from schema definitions.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Possible values:

spyUndefinedLanguage	= -1
spyJava	= 0
spyCpp	= 1
spyCSharp	= 2

3.4.26 SPYProjectItemTypes

Description
Enumeration values to identify the different elements in project item lists. See SpyProjectItem.ItemType.

Possible values:

spyUnknownItem	= 0
spyFileItem	= 1
spyFolderItem	= 2
spyURLItem	= 3

3.4.27 SPYProjectType

Description
Enumeration values to parameterize generation of C# from schema definitions.

Possible values:

spyVisualStudioProject	= 0	Obsolete
spyVisualStudio2003Project	= 1	Obsolete
spyBorlandProject	= 2	Obsolete
spyMonoMakefile	= 3	
spyVisualStudio2005Project	= 4	For C++ code also
spyVisualStudio2008Project	= 5	For C++ code also
spyVisualStudio2010Project	= 6	For C++ code also

3.4.28 SPYSampleXMLGenerationOptimization

Description
Specify the elements that will be generated in the Sample XML.
This enumeration is used in GenerateSampleXMLDlg.

Possible values:

spySampleXMLGen_Optimized	= 0
spySampleXMLGen_NonMandatoryElements	= 1
spySampleXMLGen_Everything	= 2

3.4.29 SPYSampleXMLGenerationSchemaOrDTDAssignment

Description
Specifies what kind of reference to the schema/DTD should be added to the generated Sample XML.
This enumeration is used in GenerateSampleXMLDlg.

Possible values:

spySampleXMLGen_AssignRelatively	= 0
spySampleXMLGen_AssignAbsolutely	= 1
spySampleXMLGen_DoNotAssign	= 2

3.4.30 SPYSchemaDefKind

Description
Enumeration type to select schema diagram types.

Possible values:

spyKindElement	= 0
spyKindComplexType	= 1
spyKindSimpleType	= 2
spyKindGroup	= 3
spyKindModel	= 4
spyKindAny	= 5
spyKindAttr	= 6
spyKindAttrGroup	= 7
spyKindAttrAny	= 8
spyKindIdentityUnique	= 9
spyKindIdentityKey	= 10
spyKindIdentityKeyRef	= 11
spyKindIdentitySelector	= 12
spyKindIdentityField	= 13
spyKindNotation	= 14
spyKindInclude	= 15
spyKindImport	= 16
spyKindRedefine	= 17
spyKindFacet	= 18
spyKindSchema	= 19
spyKindCount	= 20

3.4.31 SPYSchemaDocumentationFormat

Description

Enumeration values to parameterize generation of schema documentation. These values are used in SchemaDocumentationDialog.OutputFormat and WSDLDocumentationDlg.OutputFormat.

Possible values:

spySchemaDoc_HTML	= 0
spySchemaDoc_MSWord	= 1
spySchemaDoc_RTF	= 2
spySchemaDoc_PDF	= 3

3.4.32 SPYSchemaExtensionType

Description
Enumeration to specify different Schema Extension types.

Possible values:

spySchemaExtension_None	= 0
spySchemaExtension_SQL_XML	= 1
spySchemaExtension_MS_SQL_Server	= 2
spySchemaExtension_Oracle	= 3

3.4.33 SPYSchemaFormat

Description
Enumeration to specify different Schema Format types.

Possible values:

spySchemaFormat_Hierarchical	= 0
spySchemaFormat_Flat	= 1

3.4.34 SPYTextDelimiters

Description
Enumeration values to specify text delimiters for text export.

Possible values:

spyTabulator	= 0
spySemicolon	= 1
spyComma	= 2
spySpace	= 3

3.4.35 SPYTextEnclosing

Description
Enumeration value to specify text enclosing characters for text import and export.

Possible values:

spyNoEnclosing	= 0
spySingleQuote	= 1
spyDoubleQuote	= 2

3.4.36 SPYTypeDetection

Description
Enumeration to select how type detection works during GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:

spyBestPossible	= 0
spyNumbersOnly	= 1
spyNoDetection	= 2

3.4.37 SPYURLTypes

Description
Enumeration to specify different URL types.

Possible values:

spyURLTypeAuto = -1
spyURLTypeXML = 0
spyURLTypeDTD = 1

3.4.38 SPYViewModes

Description
Enumeration values that define the different view modes for XML documents. The mode *spyViewAuthentic(4)* identifies the mode that was intermediately called DocEdit mode and is now called Authentic mode. The mode *spyViewWSDL* identifies a mode which is mapped to the schema view on the GUI but distinguished internally.

Possible values:

spyViewGrid	= 0	
spyViewText	= 1	
spyViewBrowser	= 2	
spyViewSchema	= 3	
spyViewContent	= 4	// obsolete
spyViewAuthentic	= 4	
spyViewWSDL	= 5	
spyViewZIP	= 6	
spyViewEditionInfo	= 7	
spyViewXBRL	= 8	

3.4.39 SPYVirtualKeyMask

Description
Enumeration type for the most frequently used key masks that identify the status of the virtual keys. Use these values as bitmasks rather then directly comparing with them. When necessary, you can create further masks by using the 'logical or' operator.

Examples

```
'   VBScript sample: check if ctrl-key is pressed
If ((i_nVirtualKeyStatus And 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
}
```

Possible values:

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

3.4.40 SPYXMLDataKind

Description
The different types of XMLData elements available for XML documents.

Possible values:

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

3.5 Application API for Java

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

The Application API in Java has an interface built up of Java classes, each of which corresponds to an object in the Application API. Developers can use these Java classes to interact with the COM API. These classes are listed below and described in subsequent sections. For a description of the Application API objects themselves, see the Application API documentation. Bear in mind that some API features are only available in scripting environments; these have therefore not been ported to Java.

Java classes

SpyApplication
  SpyProject
    SpyProjectItems
      SpyProjectItem
  SpyDocuments
    SpyDoc
      SpyAuthenticView
        SpyAuthenticRange
      SpyDocEditView
        SpyDocEditSelection
      SpyGridView
      SpyTextView
      SpyXMLData
  SpyDialogs
    SpyCodeGeneratorDlg
    SpyDTDSchemaGeneratorDlg
    SpyFileSelectionDlg
    SpyFindInFilesDlg
    SpyGenerateSampleXMLDlg
    SpySchemaDocumentationDlg
    SpyWSDL20DocumentationDlg
    SpyWSDLDocumentationDlg
    SpyXBRLDocumentationDlg
  SpyDatabaseConnection
  SpyElementList
  SpyElementListItem
  SpyExportSettings
  SpyFindInFilesResults
     SpyFindInFilesResult
        SpyFindInFilesMatch
  SpyTextImportExportSettings

Implementation of COM properties in Java

Properties in Java have been defined to include both a **`set`** and **`get`** method (`set` if it is allowed by the COM implementation). For example, the COM class `Document` contains the **`GridView`**

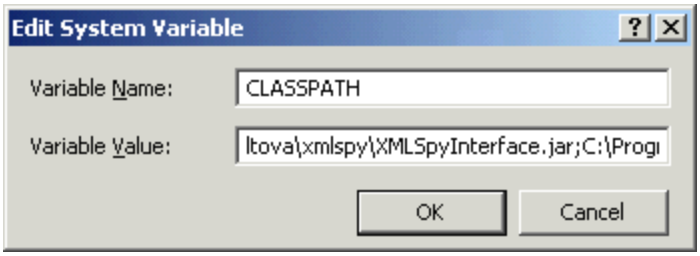
property. In Java the method is called `SpyDoc` and the property is defined as a **`GetGridView`** method.

If you encounter compiling problems, please check the following points:

- The `xmlspylib.dll` must be available in `..\windows\system32`.
- The `XMLSpyInterface.jar` file must be inserted in the ClassPath environment variable.

Setting the `ClassPath` variable in Windows XP

1. Click **Start | Settings | Control panel | System | Advanced | Environment Variables**. This opens the Environment Variables dialog box.
2. If a `ClassPath` entry already exists in the **System variables** group, select the `ClassPath` entry, and click the **Edit** button. Edit the path to: `"C:\Program Files\Altova\xmlspy\XMLSpyInterface.jar"`.



If a `ClassPath` entry does not exist in the System variables group, click the **New** button. The New System Variable dialog pops up. Enter `CLASSPATH` as the variable name, and `"C:\Program Files\Altova\xmlspy\XMLSpyInterface.jar"` as the `ClassPath` variable (alter the path to match your installation, if necessary).

3.5.1 Sample source code

The "`SpyDoc doc = app.GetDocuments().OpenFile(...)`" command parameter must be altered to suit your environment.

What the sample does:

- Starts a new XMLSpy instance
- Opens the Datasheet.xml file (alter the path here...)
- Switches to the Enhanced Grid view
- Appends a new child element called "NewChild" with the text value "NewValuE" element to the root element
- Checks if the document is valid and outputs a message to the Java console
- Quits and releases the XMLSpy application

```
import XMLSpyInterface.*;

public class TestSpyInterface
{
    public TestSpyInterface() {}

    public static void main(String[] args)
    {
        SpyApplication app = null;
        SpyDoc oDoc = null;
        SpyXMLData oData = null;
        SpyXMLData oNewChild = null;

        try
        {
            app = new SpyApplication();
            app.ShowApplication( true );

            oDoc = app.GetDocuments().OpenFile("C:\\FilePath\\OrgChart.xml",
            true );

            // OrgChart.xml is in the folder C:\Documents and
            Settings\<username>\My Documents\Altova\XMLSpy2012\. The filepath
            should be in
            // the form: C:\\Documents and
            Settings\\Username\\Folder\\Filename.xml

            if ( oDoc != null )
            {
                oDoc.SwitchViewMode(SPYViewModes.spyViewGrid);
                oData = oDoc.GetRootElement();
                oNewChild = oDoc.CreateChild(SPYXMLDataKind.spyXMLDataElement);

                oNewChild.SetName( "NewChild" );
                oNewChild.SetTextValue("newVaLuE");
                oData.AppendChild(oNewChild);

                if ( oDoc.IsValid() == false )
                {
                    // is to be expected after above insertion
                    System.out.println( "!!!!!! validation error: " +
                    oDoc.GetErrorString() );
                    System.out.println( "!!!!!! validation error: " +
                    oDoc.GetErrorPos() );
                    System.out.println( "!!!!!! validation error: " +
                    oDoc.GetBadData() );
                }
```

```
            }

            app.Quit();
        }
        finally
        {
            // Free any allocated resources by calling ReleaseInstance().
            if ( oNewChild != null )
                oNewChild.ReleaseInstance();

            if ( oData != null )
                oData.ReleaseInstance();

            if ( oDoc != null )
                oDoc.ReleaseInstance();

            if ( app != null )
                app.ReleaseInstance();
        }
    }
}
```

If you have difficulties compiling this sample, please try the following commands on the (**Start | Run | cmd**) command line. Please make sure you are currently in the folder that contains the sample java file.

compilation
javac -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface.java*

Execution
java -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface*

3.5.2 SpyApplication

```
public class SpyApplication
{
  public void ReleaseInstance();
  public void ShowApplication( boolean bShow );
  public void Quit();
  public void AddMacroMenuItem( String sMacro, String sDisplayText );
  public void ClearMacroMenu();
  public SpyDoc GetActiveDocument();
  public SpyProject GetCurrentProject();
  public SpyDocuments GetDocuments();
  public SpyElementList GetDatabaseImportElementList( SpyDatabaseConnection
  oImportSettings );
  public SpyDatabaseConnection GetDatabaseSettings();
  public SpyElementList GetDatabaseTables( SpyDatabaseConnection
  oImportSettings );
  public SpyExportSettings GetExportSettings();
  public SpyElementList GetTextImportElementList( SpyTextImportExportSettings
  oImportSettings );
  public SpyTextImportExportSettings GetTextImportExportSettings();
  public SpyDoc ImportFromDatabase( SpyDatabaseConnection oImportSettings,
  SpyElementList oElementList );
  public SpyDoc ImportFromSchema( SpyDatabaseConnection oImportSettings,
  String strTable, SpyDoc oSchemaDoc );
  public SpyDoc ImportFromText( SpyTextImportExportSettings oImportSettings,
  SpyElementList oElementList );
  public SpyDoc ImportFromWord( String sFile );
  public void NewProject( String sPath, boolean bDiscardCurrent );
  public void OpenProject( String sPath , boolean bDiscardCurrent, boolean
  bDialog );
  public long ShowForm( String sName );
  public void URLDelete( String sURL, String sUser, String sPassword );
  public void URLMakeDirectory( String sURL, String sUser, String sPassword );

  public int GetWarningNumber();
  public String GetWarningText();

  // since Version 2004R4
  public SpyApplication GetApplication();
  public SpyApplication GetParent();
  public SpyDialogs GetDialogs();
  public boolean GetVisible();
  public void SetVisible( boolean i_bVisibility );
  public long GetWindowHandle();

  public void ReloadSettings();
  public SpyFindInFilesResults FindInFiles( SpyFindInFilesDlg dlgSettings );
  public boolean ShowFindInFiles( SpyFindInFilesDlg dlgSettings );
  public void Selection( String sVal );

  public long Status();
  public int MajorVersion();
  public int MinorVersion();
  public String Edition();
  public boolean IsAPISupported();
  public long ServicePackVersion();
  public void CreateXMLSchemaFromDBStructure( SpyDatabaseConnection
oConnection, SpyElementList oTables );
}
```

3.5.3 SpyCodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

```
// since version 2004R4
public class SpyCodeGeneratorDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();
  public long GetProgrammingLanguage();
  public void SetProgrammingLanguage( long i_eVal );
  public String GetTemplateFileName();
  public void SetTemplateFileName( String i_strVal );
  public String GetOutputPath();
  public void SetOutputPath( String i_strVal );
  public long GetOutputPathDialogAction();
  public void SetOutputPathDialogAction( long i_eVal );
  public long GetPropertySheetDialogAction();
  public void SetPropertySheetDialogAction( long i_eVal );
  public long GetOutputResultDialogAction();
  public void SetOutputResultDialogAction( long i_eVal );
  public long GetCPPSettings_DOMType();
  public void SetCPPSettings_DOMType( long i_eVal );
  public long GetCPPSettings_LibraryType();
  public void SetCPPSettings_LibraryType( long i_eVal );
  public boolean GetCPPSettings_UseMFC();
  public void SetCPPSettings_UseMFC( boolean i_bVal );
  public long GetCSharpSettings_ProjectType();
  public void SetCSharpSettings_ProjectType( long i_eVal );
}
```

3.5.4 SpyDatabaseConnection

```
public class SpyDatabaseConnection
{
  public void ReleaseInstance();
  public String GetADOConnection();
  public void SetADOConnection( String sValue );
  public boolean GetAsAttributes();
  public void SetAsAttributes( boolean bValue );
  public boolean GetCreateMissingTables();
  public void SetCreateMissingTables( boolean bValue );
  public boolean GetCreateNew();
  public void SetCreateNew( boolean bValue );
  public boolean GetExcludeKeys();
  public void SetExcludeKeys( boolean bValue );
  public String GetFile();
  public void SetFile( String sValue );
  public boolean GetIncludeEmptyElements();
  public void SetIncludeEmptyElements( boolean bValue );
  public long GetNumberDateTimeFormat();
  public void SetNumberDateTimeFormat( long nValue );
  public String GetODBCConnection();
  public void SetODBCConnection( String sValue );
  public String GetSQLSelect();
  public void SetSQLSelect( String sValue );
  public long GetTextFieldLen();
  public void SetTextFieldLen( long nValue );

  // since version 2004R4
  public long GetDatabaseKind();
  public void SetDatabaseKind( long nValue );

  // since version 2008R2
  public boolean GetCommentIncluded();
  public void SetCommentIncluded( boolean bValue );
  public String GetNullReplacement();
  public void SetNullReplacement( String sValue );
  public String GetDatabaseSchema();
  public void SetDatabaseSchema( String sValue );

  // since version 2010r3
  public boolean GetPrimaryKeys()
  public void SetPrimaryKeys( boolean bValue )
  public boolean GetForeignKeys()
  public void SetForeignKeys( boolean bValue )
  public boolean GetUniqueKeys()
  public void SetUniqueKeys( boolean bValue )
  public long GetSchemaExtensionType()
  public void SetSchemaExtensionType( long nValue )
  public long GetSchemaFormat()
  public void SetSchemaFormat( long nValue )
  public long GetImportColumnsType()
  public void SetImportColumnsType( long nValue )
}
```

3.5.5 SpyDialogs

```
// Since version 2004R4
public class SpyDialogs
{
  public SpyApplication GetApplication();
  public SpyApplication GetParent();
  public SpyCodeGeneratorDlg GetCodeGeneratorDlg();
  public SpyFileSelectionDlg GetFileSelectionDlg();
  public SpySchemaDocumentationDlg GetSchemaDocumentationDlg();
  public SpyGenerateSampleXMLDlg GetGenerateSampleXMLDlg();
  public SpyDTDSchemaGeneratorDlg GetDTDSchemaGeneratorDlg();
  public SpyFindInFilesDlg GetFindInFilesDlg();
  public SpyWSDLDocumentationDlg GetWSDLDocumentationDlg();

  // Since version 2010
  public SpyWSDL20DocumentationDlg GetWSDL20DocumentationDlg();
  public SpyXBRLDocumentationDlg GetXBRLDocumentationDlg();
}
```

3.5.6 SpyDoc

```
public class SpyDoc
{
  public void ReleaseInstance();
  public void SetEncoding( String strEncoding );
  public void SetPathName( String strPath );
  public String GetPathName();
  public String GetTitle();
  public boolean IsModified();
  public void Save();
  public void Close( boolean bDiscardChanges );
  public void UpdateViews();
  public long GetCurrentViewMode();
  public boolean SwitchViewMode( long nMode );
  public SpyGridView GetGridView();
  public void SetActiveDocument();
  public void StartChanges();
  public void EndChanges();
  public void TransformXSL();
  public void AssignDTD( String sDTDFile, boolean bDialog );
  public void AssignSchema( String sSchemaFile, boolean bDialog );
  public void AssignXSL( String sXSLFile, boolean bDialog );
  public void ConvertDTDOrSchema( long nFormat, long nFrequentElements );
  public SpyXMLData CreateChild( long nKind );
  public void CreateSchemaDiagram( long nKind, String sName, String sFile );
  public SpyDocEditView GetDocEditView();
  public void ExportToDatabase( SpyXMLData oFromChild, SpyExportSettings
  oExportSettings, SpyDatabaseConnection oDatabaseConnection );
  public void ExportToText( SpyXMLData oFromChild, SpyExportSettings
  oExportSettings, SpyTextImportExportSettings oTextSettings );
  public void GenerateDTDOrSchema( long nFormat, int nValuesList, long
  nDetection, long nFrequentElements );
  public SpyElementList GetExportElementList( SpyXMLData oFromChild,
  SpyExportSettings oExportSettings );
  public SpyXMLData GetRootElement();
  public String SaveInString( SpyXMLData oData, boolean bMarked );
  public void SaveToURL( String sUrl, String sUser, String sPassword );
  public String GetErrorString(); // See IsValid() or IsWellFormed()
  public int GetErrorPos(); // See IsValid() or IsWellFormed()
  public SpyXMLData GetBadData(); // See IsValid() or IsWellFormed()
  public boolean IsValid();
  public boolean IsWellFormed( SpyXMLData oData, boolean bWithChildren );

  // Since version 2004R3
  public SpyAuthenticView GetAuthenticView()

  // Since version 2004R4
  public SpyApplication GetApplication();
  public SpyDocuments GetParent();
  public String GetFullName();
  public void SetFullName( String i_strName );
  public String GetName();
  public String GetPath();
  public boolean GetSaved();
  public void SaveAs( String i_strFileNameOrPath );
  public String GetEncoding();
  public SpyXMLData GetDataRoot();
  public void GenerateProgramCode( SpyCodeGeneratorDlg i_dlg );
  public void AssignXSLFO( String i_strFile, boolean i_bUseDialog );
  public void TransformXSLFO();
  public void GenerateSchemaDocumentation( SpySchemaDocumentationDlg i_dlg );
```

```
    public void ExecuteXQuery( String i_strXMLSourceFile );
    public void SetExternalIsValid( boolean bIsValid );
    public SpyDoc GenerateSampleXML( SpyGenerateSampleXMLDlg ipGenerateXMLDlg );
    public boolean UpdateXMLData();
    public String GetAsXMLString();
    public void SetAsXMLString( String newVal );
    public SpyDoc GenerateDTDOrSchemaEx( SpyDTDSchemaGeneratorDlg
    ipDTDSchemaGeneratorDlg );
    public SpyDoc ConvertDTDOrSchemaEx( long nFormat, long nFrequentElements,
    String sOutputPath, long nOutputPathDialogAction );
    public SpyTextView GetTextView();
    public String[] GetSuggestions();
    public void SetSuggestions( String[] aList );
    public void SetSelection( String sVal );

    // Since version 2009
    public void GenerateWSDLDocumentation( SpyWSDLDocumentationDlg
    ipWSDLDocumenationDlg );
    public void TransformXSLEx( long nDialogAction );

    // Since version 2010
    public void GenerateWSDL20Documentation( SpyWSDL20DocumentationDlg
    ipWSD20DocumenationDlg );
    public void GenerateXBRLDocumentation( SpyXBRLDocumentationDlg
    ipXBRLDocumentationDlg );
    public SpyDoc ConvertToWSDL20( String sFilePath, boolean bShowDialogs );

    // Since version 2010r3
    public String CreateDBStructureFromXMLSchema( SpyDatabaseConnection
    oConnection, SpyElementList oTables, boolean bDropTableWithExistingName );
    public SpyElementList GetDBStructureList( SpyDatabaseConnection oConnection
    );
}
```

3.5.7 SpyDocuments

```
public class SpyDocuments
{
  public void ReleaseInstance();
  public long Count();
  public SpyDoc GetItem( long nNo );
  public SpyDoc NewFile( String strFile, String strType );
  public SpyDoc NewFileFromText( String nSource, String strType );
  public SpyDoc OpenFile( String sPath, boolean bDialog );
  public SpyDoc OpenURL( String sUrl, long nURLType, long nLoading, String
  sUser, String sPassword );
  public SpyDoc OpenURLDialog( String sURL, long nURLType, long nLoading,
  String sUser, String sPassword );
  // Since version 2011r2
  public SpyDoc NewAuthenticFile( String strSPSPath, String strXMLPath );
  public SpyDoc OpenAuthenticFile( String strSPSPath, String strXMLPath );
}
```

3.5.8 SpyDTDSchemaGeneratorDlg

```
public class SpyDTDSchemaGeneratorDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public long GetDTDSchemaFormat();
  public void SetDTDSchemaFormat( long newVal );
  public short GetValueList();
  public void SetValueList( short newVal );
  public long GetTypeDetection();
  public void SetTypeDetection( long newVal );
  public long GetFrequentElements();
  public void SetFrequentElements( long newVal );
  public boolean GetMergeAllEqualNamed();
  public void SetMergeAllEqualNamed( boolean newVal );
  public boolean GetResolveEntities();
  public void SetResolveEntities( boolean newVal );
  public long GetAttributeTypeDefinition();
  public void SetAttributeTypeDefinition( long newVal );
  public boolean GetGlobalAttributes();
  public void SetGlobalAttributes( boolean newVal );
  public boolean GetOnlyStringEnums();
  public void SetOnlyStringEnums( boolean newVal );
  public long GetMaxEnumLength();
  public void SetMaxEnumLength( long newVal );
  public String GetOutputPath();
  public void SetOutputPath( String newVal );
  public long GetOutputPathDialogAction();
  public void SetOutputPathDialogAction( long newVal );
}
```

3.5.9 SpyElementList

```
public class SpyElementList
{
  public void ReleaseInstance();
  public long GetCount();
  public SpyElementListItem GetItem( long nIndex );
  public void RemoveElement( long nIndex );
}
```

3.5.10 SpyElementListItem

```
public class SpyElementListItem
{
  public void ReleaseInstance();
  public long GetElementKind();
  public void SetElementKind( long nKind );
  public long GetFieldCount();
  public String GetName();
  public long GetRecordCount();
}
```

3.5.11 SpyExportSettings

```
public class SpyExportSettings
{
  public void ReleaseInstance();
  public boolean GetCreateKeys();
  public void SetCreateKeys( boolean bValue );
  public SpyElementList GetElementList();
  public void SetElementList( SpyElementList obj );
  public boolean GetEntitiesToText ();
  public void SetEntitiesToText( boolean bValue );
  public boolean GetExportAllElements();
  public void SetExportAllElements( boolean bValue );
  public boolean GetFromAttributes();
  public void SetFromAttributes( boolean bValue );
  public boolean GetFromSingleSubElements();
  public void SetFromSingleSubElements( boolean bValue );
  public boolean GetFromTextValues();
  public void SetFromTextValues( boolean bValue );
  public boolean GetIndependentPrimaryKey();
  public void SetIndependentPrimaryKey( boolean bValue );
  public long GetNamespace();
  public void SetNamespace( long nValue );
  public int GetSubLevelLimit();
  public void SetSubLevelLimit( int nValue );
}
```

3.5.12 SpyFileSelectionDlg

```
// Since version 2004R4
public class SpyFileSelectionDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();
  public String GetFullName();
  public void SetFullName( String i_strName );
  public long GetDialogAction();
  public void SetDialogAction( long i_eAction );
}
```

3.5.13 SpyFindInFilesDlg

```
public class SpyFindInFilesDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public String GetFind();
  public void SetFind( String sNewVal );
  public boolean GetRegularExpression();
  public void SetRegularExpression( boolean bNewVal );
  public String GetReplace();
  public void SetReplace( String sNewVal );
  public boolean GetReplaceOnDisk();
  public void SetReplaceOnDisk( boolean bNewVal );
  public boolean GetDoReplace();
  public void SetDoReplace( boolean bNewVal );
  public boolean GetMatchWholeWord();
  public void SetMatchWholeWord( boolean bNewVal );
  public boolean GetMatchCase();
  public void SetMatchCase( boolean bNewVal );
  public long GetSearchLocation();
  public void SetSearchLocation( long nPosition );
  public String GetStartFolder();
  public void SetStartFolder( String sNewVal );
  public boolean GetIncludeSubfolders();
  public void SetIncludeSubfolders( boolean bNewVal );
  public boolean GetSearchInProjectFilesDoExternal();
  public void SetSearchInProjectFilesDoExternal( boolean bNewVal );
  public String GetFileExtension();
  public void SetFileExtension( String sNewVal );
  public boolean GetAdvancedXMLSearch();
  public void SetAdvancedXMLSearch( boolean bNewVal );
  public boolean GetXMLElementNames();
  public void SetXMLElementNames( boolean bNewVal );
  public boolean GetXMLElementContents();
  public void SetXMLElementContents( boolean bNewVal );
  public boolean GetXMLAttributeNames();
  public void SetXMLAttributeNames( boolean bNewVal );
  public boolean GetXMLAttributeContents();
  public void SetXMLAttributeContents( boolean bNewVal );
  public boolean GetXMLComments();
  public void SetXMLComments( boolean bNewVal );
  public boolean GetXMLCData();
  public void SetXMLCData( boolean bNewVal );
  public boolean GetXMLPI();
  public void SetXMLPI( boolean bNewVal );
  public boolean GetXMLRest();
  public void SetXMLRest( boolean bNewVal );
  public boolean GetShowResult();
  public void SetShowResult( boolean bNewVal );
}
```

3.5.14 SpyFindInFilesMatch

```
public class SpyFindInFilesMatch
{
  public void ReleaseInstance();
  public long Line();
  public long Position();
  public long Length();
  public String LineText();
  public boolean Replaced();
}
```

3.5.15 SpyFindInFilesResult

```
public class SpyFindInFilesResult
{
  public void ReleaseInstance();
  public long Count();
  public SpyFindInFilesMatch GetItem( long nNo );
  public String GetPath();
  public SpyDoc GetDocument();
}
```

3.5.16 SpyFindInFilesResults

```
public class SpyFindInFilesResults
{
  public void ReleaseInstance();
  public long Count();
  public SpyFindInFilesResult GetItem( long nNo );
}
```

3.5.17 SpyGenerateSampleXMLDlg

```
public class SpyGenerateSampleXMLDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public boolean GetNonMandatoryAttributes();
  public void SetNonMandatoryAttributes( boolean newVal );
  public boolean GetNonMandatoryElements();
  public void SetNonMandatoryElements( boolean newVal );
  public boolean GetTakeFirstChoice();
  public void SetTakeFirstChoice( boolean newVal );
  public long GetRepeatCount();
  public void SetRepeatCount( long newVal );
  public boolean GetFillWithSampleData();
  public void SetFillWithSampleData( boolean newVal );
  public boolean GetFillElementsWithSampleData();
  public void SetFillElementsWithSampleData( boolean newVal );
  public boolean GetFillAttributesWithSampleData();
  public void SetFillAttributesWithSampleData( boolean newVal );
  public boolean GetContentOfNillableElementsIsNonMandatory();
  public void SetContentOfNillableElementsIsNonMandatory( boolean newVal );
  public boolean GetTryToUseNonAbstractTypes();
  public void SetTryToUseNonAbstractTypes( boolean newVal );
  public long GetOptimization();
  public void SetOptimization( long newVal );
  public long GetSchemaOrDTDAssignment();
  public void SetSchemaOrDTDAssignment( long newVal );
  public String GetLocalNameOfRootElement();
  public void SetLocalNameOfRootElement( String newVal );
  public String GetNamespaceURIOfRootElement();
  public void SetNamespaceURIOfRootElement( String newVal );
  public long GetOptionsDialogAction();
  public void SetOptionsDialogAction( long newVal );
}
```

3.5.18 SpyGridView

```
public class SpyGridView
{
  public void ReleaseInstance();
  public SpyXMLData GetCurrentFocus();
  public void Deselect( SpyXMLData oData );
  public boolean GetIsVisible();
  public void Select( SpyXMLData oData );
  public void SetFocus( SpyXMLData oData );
}
```

3.5.19 SpyProject

```
public class SpyProject
{
  public void ReleaseInstance();
  public void CloseProject( boolean bDiscardChanges, boolean bCloseFiles,
  boolean bDialog );
  public String GetProjectFile();
  public void SetProjectFile( String sFile );
  public SpyProjectItems GetRootItems();
  public void SaveProject();
  public void SaveProjectAs( String sPath, boolean bDialog );
}
```

3.5.20 SpyProjectItem

```
public class SpyProjectItem
{
  public void ReleaseInstance();
  public SpyProjectItems GetChildItems();
  public String GetFileExtensions();
  public void SetFileExtensions( String sExtensions );
  public long GetItemType();
  public String GetName();
  public SpyDoc Open();
  public SpyProjectItem GetParentItem();
  public String GetPath();
  public String GetValidateWith();
  public void SetValidateWith( String sVal );
  public String GetXMLForXSLTransformation();
  public void SetXMLForXSLTransformation( String sVal );
  public String GetXSLForXMLTransformation();
  public void SetXSLForXMLTransformation( String sVal );
  public String GetXSLTransformationFileExtension();
  public void SetXSLTransformationFileExtension( String sVal );
  public String GetXSLTransformationFolder();
  public void SetXSLTransformationFolder( String sVal );
}
```

3.5.21 SpyProjectItems

```
public class SpyProjectItems
{
  public void ReleaseInstance();
  public void AddFile( String sPath );
  public void AddFolder( String sName );
  public void AddURL( String sURL, long nURLType, String sUser, String
  sPassword, boolean bSave );
  public long Count();
  public SpyProjectItem GetItem( long nNumber );
  public void RemoveItem( SpyProjectItem oItemToRemove );
}
```

3.5.22 SpySchemaDocumentationDlg

```
// Since version 2004R4
public class SpySchemaDocumentationDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();

  public String GetOutputFile();
  public void SetOutputFile( String i_strVal );
  public long GetOutputFormat();
  public void SetOutputFormat( long i_eVal );

  public boolean GetShowResult();
  public void SetShowResult( boolean i_bVal );
  public long GetOptionsDialogAction();
  public void SetOptionsDialogAction( long i_eVal );
  public long GetOutputFileDialogAction();
  public void SetOutputFileDialogAction( long i_eVal );
  public boolean GetShowProgressBar();
  public void SetShowProgressBar( boolean i_bVal );

  public void IncludeAll( boolean i_bInclude );
  public boolean GetIncludeIndex();
  public void SetIncludeIndex( boolean i_bVal );
  public boolean GetIncludeGlobalElements();
  public void SetIncludeGlobalElements( boolean i_bVal );
  public boolean GetIncludeLocalElements();
  public void SetIncludeLocalElements( boolean i_bVal );
  public boolean GetIncludeGroups();
  public void SetIncludeGroups( boolean i_bVal );
  public boolean GetIncludeComplexTypes();
  public void SetIncludeComplexTypes( boolean i_bVal );
  public boolean GetIncludeSimpleTypes();
  public void SetIncludeSimpleTypes( boolean i_bVal );
  public boolean GetIncludeAttributeGroups();
  public void SetIncludeAttributeGroups( boolean i_bVal );
  public boolean GetIncludeRedefines();
  public void SetIncludeRedefines( boolean i_bVal );

  public void AllDetails( boolean i_bDetailsOn );
  public boolean GetShowDiagram();
  public void SetShowDiagram( boolean i_bVal );
  public boolean GetShowNamespace();
  public void SetShowNamespace( boolean i_bVal );
  public boolean GetShowType();
  public void SetShowType( boolean i_bVal );
  public boolean GetShowChildren();
  public void SetShowChildren( boolean i_bVal );
  public boolean GetShowUsedBy();
  public void SetShowUsedBy( boolean i_bVal );
  public boolean GetShowProperties();
  public void SetShowProperties( boolean i_bVal );
  public boolean GetShowSingleFacets();
  public void SetShowSingleFacets( boolean i_bVal );
  public boolean GetShowPatterns();
  public void SetShowPatterns( boolean i_bVal );
  public boolean GetShowEnumerations();
  public void SetShowEnumerations( boolean i_bVal );
  public boolean GetShowAttributes();
  public void SetShowAttributes( boolean i_bVal );
```

```
public boolean GetShowIdentityConstraints();
public void SetShowIdentityConstraints( boolean i_bVal );
public boolean GetShowAnnotations();
public void SetShowAnnotations( boolean i_bVal );
public boolean GetShowSourceCode();
public void SetShowSourceCode( boolean i_bVal );

// Since version 2009
public boolean GetEmbedDiagrams();
public void SetEmbedDiagrams( boolean i_bVal );
public long GetDiagramFormat();
public void SetDiagramFormat( long i_nVal );
public boolean GetIncludeGlobalAttributes();
public void SetIncludeGlobalAttributes( boolean i_bVal );
public boolean GetIncludeLocalAttributes();
public void SetIncludeLocalAttributes( boolean i_bVal );
public boolean GetIncludeReferencedSchemas();
public void SetIncludeReferencedSchemas( boolean i_bVal );
public boolean GetMultipleOutputFiles();
public void SetMultipleOutputFiles( boolean i_bVal );

// Since version 2010
public boolean GetEmbedCSSInHTML();
public void SetEmbedCSSInHTML( boolean i_bVal );
public boolean GetCreateDiagramsFolder();
public void SetCreateDiagramsFolder( boolean i_bVal );

// Since version 2010r3
public boolean GetGenerateRelativeLinks();
public void SetGenerateRelativeLinks( boolean i_bVal );

// Since version 2011r2
public boolean GetUseFixedDesign();
public void SetUseFixedDesign( boolean i_bVal );
public String GetSPSFile();
public void SetSPSFile( String i_strVal );
}
```

3.5.23 SpyTextImportExportSettings

```
public class SpyTextImportExportSettings
{
  public void ReleaseInstance();
  public String GetDestinationFolder();
  public void SetDestinationFolder( String sVal );
  public long GetEnclosingCharacter();
  public void SetEnclosingCharacter( long nEnclosing );
  public String GetEncoding();
  public void SetEncoding( String sVal );
  public long GetEncodingByteOrder();
  public void SetEncodingByteOrder( long nByteOrder );
  public long GetFieldDelimiter();
  public void SetFieldDelimiter( long nDelimiter );
  public String GetFileExtension ();
  public void SetFileExtension( String sVal );
  public boolean GetHeaderRow();
  public void SetHeaderRow( boolean bVal );
  public String GetImportFile();
  public void SetImportFile( String sVal );
}
```

3.5.24 SpyTextView

```
public class SpyTextView
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDoc GetParent();
  public long LineFromPosition( long nCharPos );
  public long PositionFromLine( long nLine );
  public long LineLength( long nLine );
  public String GetSelText();
  public void SetSelText( String sText );
  public String GetRangeText( long nPosFrom, long nPosTill );
  public void ReplaceText( long nPosFrom, long nPosTill, String sText );
  public void MoveCaret( long nDiff );
  public void GoToLineChar( long nLine, long nChar );
  public void SelectText( long nPosFrom, long nPosTill );
  public long GetSelectionStart();
  public void SetSelectionStart( long nNewVal );
  public long GetSelectionEnd();
  public void SetSelectionEnd( long nNewVal );
  public String GetText();
  public void SetText( String sText );
  public long LineCount();
  public long Length();
}
```

3.5.25 SpyWSDL20DocumentationDlg

```
// Since version 2010
public class SpyWSDL20DocumentationDlg
{
	public void ReleaseInstance();
	public SpyApplication GetApplication();

	public long GetOptionsDialogAction();
	public void SetOptionsDialogAction( long nNewVal );

	public long GetOutputFileDialogAction();
	public void SetOutputFileDialogAction( long nNewVal );

	public boolean GetShowProgressBar();
	public void SetShowProgressBar( boolean bNewVal );

	public String GetOutputFile();
	public void SetOutputFile( String sNewVal );

	public long GetOutputFormat();
	public void SetOutputFormat( long nNewVal );

	public boolean GetMultipleOutputFiles();
	public void SetMultipleOutputFiles( boolean bNewVal );

	public boolean GetEmbedCSSInHTML();
	public void SetEmbedCSSInHTML( boolean bNewVal );

	public long GetDiagramFormat();
	public void SetDiagramFormat( long nNewVal );

	public boolean GetEmbedDiagrams();
	public void SetEmbedDiagrams( boolean bNewVal );

	public boolean GetCreateDiagramsFolder();
	public void SetCreateDiagramsFolder( boolean bNewVal );

	public boolean GetShowResult();
	public void SetShowResult( boolean bNewVal );

	public void IncludeAll( boolean bNewVal );
	public void AllDetails( boolean bNewVal );

	public boolean GetIncludeOverview();
	public void SetIncludeOverview( boolean bNewVal );

	public boolean GetIncludeService();
	public void SetIncludeService( boolean bNewVal );

	public boolean GetIncludeBinding();
	public void SetIncludeBinding( boolean bNewVal );

	public boolean GetIncludeInterface();
	public void SetIncludeInterface( boolean bNewVal );

	public boolean GetIncludeTypes();
```

```
        public void SetIncludeTypes( boolean bNewVal );

        public boolean GetIncludeImportedWSDLFiles();
        public void SetIncludeImportedWSDLFiles( boolean bNewVal );

        public boolean GetShowServiceDiagram();
        public void SetShowServiceDiagram( boolean bNewVal );

        public boolean GetShowBindingDiagram();
        public void SetShowBindingDiagram( boolean bNewVal );

        public boolean GetShowInterfaceDiagram();
        public void SetShowInterfaceDiagram( boolean bNewVal );

        public boolean GetShowTypesDiagram();
        public void SetShowTypesDiagram( boolean bNewVal );

        public boolean GetShowEndpoint();
        public void SetShowEndpoint( boolean bNewVal );

        public boolean GetShowSourceCode();
        public void SetShowSourceCode( boolean bNewVal );

        public boolean GetShowExtensibility();
        public void SetShowExtensibility( boolean bNewVal );

        public boolean GetShowUsedBy();
        public void SetShowUsedBy( boolean bNewVal );

        public boolean GetShowOperation();
        public void SetShowOperation( boolean bNewVal );

        public boolean GetShowFault();
        public void SetShowFault( boolean bNewVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign();
        public void SetUseFixedDesign( boolean i_bVal );

        public String GetSPSFile();
        public void SetSPSFile( String i_strVal );

    }
```

3.5.26 SpyWSDLDocumentationDlg

```
// Since version 2008r2sp1
public class SpyWSDLDocumentationDlg
{
	public void ReleaseInstance();
	public SpyApplication GetApplication();

	public String GetOutputFile();
	public void SetOutputFile( String sNewVal );

	public long GetOutputFileDialogAction();
	public void SetOutputFileDialogAction( long nNewVal );

	public long GetOptionsDialogAction();
	public void SetOptionsDialogAction( long nNewVal );

	public boolean GetShowProgressBar();
	public void SetShowProgressBar( boolean bNewVal );

	public boolean GetShowResult();
	public void SetShowResult( boolean bNewVal );

	public long GetOutputFormat();
	public void SetOutputFormat( long nNewVal );

	public boolean GetEmbedDiagrams();
	public void SetEmbedDiagrams( boolean bNewVal );

	public long GetDiagramFormat();
	public void SetDiagramFormat( long nNewVal );

	public boolean GetMultipleOutputFiles();
	public void SetMultipleOutputFiles( boolean bNewVal );

	public void IncludeAll( boolean bNewVal );

	public boolean GetIncludeBinding();
	public void SetIncludeBinding( boolean bNewVal );

	public boolean GetIncludeImportedWSDLFiles();
	public void SetIncludeImportedWSDLFiles( boolean bNewVal );

	public boolean GetIncludeMessages();
	public void SetIncludeMessages( boolean bNewVal );

	public boolean GetIncludeOverview();
	public void SetIncludeOverview( boolean bNewVal );

	public boolean GetIncludePortType();
	public void SetIncludePortType( boolean bNewVal );

	public boolean GetIncludeService();
	public void SetIncludeService( boolean bNewVal );

	public boolean GetIncludeTypes();
	public void SetIncludeTypes( boolean bNewVal );

	public void AllDetails( boolean bNewVal );

	public boolean GetShowBindingDiagram();
	public void SetShowBindingDiagram( boolean bNewVal );
```

```
        public boolean GetShowExtensibility();
        public void SetShowExtensibility( boolean bNewVal );

        public boolean GetShowMessageParts();
        public void SetShowMessageParts( boolean bNewVal );

        public boolean GetShowPort();
        public void SetShowPort( boolean bNewVal );

        public boolean GetShowPortTypeDiagram();
        public void SetShowPortTypeDiagram( boolean bNewVal );

        public boolean GetShowPortTypeOperations();
        public void SetShowPortTypeOperations( boolean bNewVal );

        public boolean GetShowServiceDiagram();
        public void SetShowServiceDiagram( boolean bNewVal );

        public boolean GetShowSourceCode();
        public void SetShowSourceCode( boolean bNewVal );

        public boolean GetShowTypesDiagram();
        public void SetShowTypesDiagram( boolean bNewVal );

        public boolean GetShowUsedBy();
        public void SetShowUsedBy( boolean bNewVal );

        // Since version 2010
        public boolean GetEmbedCSSInHTML();
        public void SetEmbedCSSInHTML( boolean i_bVal );

        public boolean GetCreateDiagramsFolder();
        public void SetCreateDiagramsFolder( boolean i_bVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign();
        public void SetUseFixedDesign( boolean i_bVal );

        public String GetSPSFile();
        public void SetSPSFile( String i_strVal );

    }
```

3.5.27 SpyXBRLDocumentationDlg

```
// Since version 2010
public class SpyXBRLDocumentationDlg
{
	public void ReleaseInstance();
	public SpyApplication GetApplication();

	public long GetOptionsDialogAction();
	public void SetOptionsDialogAction( long nNewVal );

	public long GetOutputDialogAction();
	public void SetOutputDialogAction( long nNewVal );

	public boolean GetShowProgressBar();
	public void SetShowProgressBar( boolean bNewVal );

	public String GetOutputFile();
	public void SetOutputFile( String sNewVal );

	public long GetOutputFormat();
	public void SetOutputFormat( long nNewVal );

	public boolean GetEmbedCSSInHTML();
	public void SetEmbedCSSInHTML( boolean bNewVal );

	public long GetDiagramFormat();
	public void SetDiagramFormat( long nNewVal );

	public boolean GetEmbedDiagrams();
	public void SetEmbedDiagrams( boolean bNewVal );

	public boolean GetCreateDiagramsFolder();
	public void SetCreateDiagramsFolder( boolean bNewVal );

	public boolean GetShowResult();
	public void SetShowResult( boolean bNewVal );

	public void IncludeAll( boolean bNewVal );
	public void AllDetails( boolean bNewVal );

	public boolean GetIncludeOverview();
	public void SetIncludeOverview( boolean bNewVal );

	public boolean GetIncludeNamespacePrefixes();
	public void SetIncludeNamespacePrefixes( boolean bNewVal );

	public boolean GetIncludeGlobalElements();
	public void SetIncludeGlobalElements( boolean bNewVal );

	public boolean GetIncludeDefinitionLinkroles();
	public void SetIncludeDefinitionLinkroles( boolean bNewVal );

	public boolean GetIncludePresentationLinkroles();
	public void SetIncludePresentationLinkroles( boolean bNewVal );

	public boolean GetIncludeCalculationLinkroles();
	public void SetIncludeCalculationLinkroles( boolean bNewVal );
```

```
        public boolean GetShowDiagram();
        public void SetShowDiagram( boolean bNewVal );

        public boolean GetShowSubstitutiongroup();
        public void SetShowSubstitutiongroup( boolean bNewVal );

        public boolean GetShowItemtype();
        public void SetShowItemtype( boolean bNewVal );

        public boolean GetShowBalance();
        public void SetShowBalance( boolean bNewVal );

        public boolean GetShowPeriod();
        public void SetShowPeriod( boolean bNewVal );

        public boolean GetShowAbstract();
        public void SetShowAbstract( boolean bNewVal );

        public boolean GetShowNillable();
        public void SetShowNillable( boolean bNewVal );

        public boolean GetShowLabels();
        public void SetShowLabels( boolean bNewVal );

        public boolean GetShowReferences();
        public void SetShowReferences( boolean bNewVal );

        public boolean GetShowLinkbaseReferences();
        public void SetShowLinkbaseReferences( boolean bNewVal );

        public boolean GetShortQualifiedName();
        public void SetShortQualifiedName( boolean bNewVal );

        public boolean GetShowImportedElements();
        public void SetShowImportedElements( boolean bNewVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign();
        public void SetUseFixedDesign( boolean i_bVal );

        public String GetSPSFile();
        public void SetSPSFile( String i_strVal );
};
```

3.5.28 SpyXMLData

```
public class SpyXMLData
{
  public void ReleaseInstance();
  public void AppendChild( SpyXMLData oNewData );
  public void EraseAllChildren();
  public void EraseCurrentChild();
  public SpyXMLData GetCurrentChild();
  public SpyXMLData GetFirstChild( long nKind );
  public SpyXMLData GetNextChild();
  public boolean GetHasChildren();
  public void InsertChild( SpyXMLData oNewData );
  public boolean IsSameNode( SpyXMLData oToComp);
  public long GetKind();
  public boolean GetMayHaveChildren();
  public String GetName();
  public void SetName( String sValue );
  public SpyXMLData GetParent();
  public String GetTextValue();
  public void SetTextValue( String sValue );
}
```

3.5.29 Authentic

SpyAuthenticRange

```
// Since version 2004R3
public class SpyAuthenticRange
{
    public void ReleaseInstance();
    public SpyApplication GetApplication();
    public SpyAuthenticView GetParent();
    public SpyAuthenticRange GotoNext( long eKind );
    public SpyAuthenticRange GotoPrevious( long eKind );
    public void Select();
    public long GetFirstTextPosition();
    public void SetFirstTextPosition( long nTextPosition );
    public long GetLastTextPosition();
    public void SetLastTextPosition( long nTextPosition );
    public String GetText();
    public void SetText( String strText );
    public boolean PerformAction( long eAction, String strElementName );
    public boolean CanPerformAction( long eAction, String strElementName );
    public String[] CanPerformActionWith( long eAction );
    public SpyAuthenticRange GoTo( long eKind, long nCount, long nFrom );
    public SpyAuthenticRange SelectNext( long eKind );
    public SpyAuthenticRange SelectPrevious( long eKind );
    public SpyAuthenticRange MoveBegin( long eKind, long nCount );
    public SpyAuthenticRange MoveEnd( long eKind, long nCount );
    public SpyAuthenticRange ExpandTo( long eKind );
    public SpyAuthenticRange CollapsToBegin();
    public SpyAuthenticRange CollapsToEnd();
    public SpyAuthenticRange GotoNextCursorPosition();
    public SpyAuthenticRange GotoPreviousCursorPosition();
    public boolean IsEmpty();
    public boolean IsEqual( SpyAuthenticRange ipCmp );
    public SpyAuthenticRange Clone();
    public SpyAuthenticRange SetFromRange( SpyAuthenticRange ipSrc );
    public boolean Delete();
    public boolean Cut();
    public boolean Copy();
    public boolean Paste();
    public SpyXMLData GetFirstXMLData();
    public void SetFirstXMLData( SpyXMLData objXMLDataPtr );
    public long GetFirstXMLDataOffset();
    public void SetFirstXMLDataOffset( long nOffset );
    public SpyXMLData GetLastXMLData();
    public void SetLastXMLData( SpyXMLData objXMLDataPtr );
    public long GetLastXMLDataOffset();
    public void SetLastXMLDataOffset( long nOffset );
    public String[] GetElementHierarchy();
    public String[] GetElementAttributeNames( String strElementName );
    public boolean HasElementAttribute( String strElementName, String
strAttributeName );
    public String GetElementAttributeValue( String strElementName, String
strAttributeName );
    public void SetElementAttributeValue( String strElementName, String
strAttributeName, String strNewValue );
    public String[] GetEntityNames();
    public void InsertEntity( String strEntityName );
    public boolean IsInDynamicTable();
    public boolean AppendRow();
    public boolean InsertRow();
```

```
    public boolean DuplicateRow();
    public boolean DeleteRow();
    public boolean MoveRowUp();
    public boolean MoveRowDown();

    // Since version 2004R4
    public boolean IsCopyEnabled();
    public boolean IsCutEnabled();
    public boolean IsPasteEnabled();
    public boolean IsDeleteEnabled();
    public boolean IsTextStateApplied( String i_strElementName );
    public boolean IsFirstRow();
    public boolean IsLastRow();
}
```

SpyAuthenticView

```
// Since version 2004R3
public class SpyAuthenticView
{
    public void ReleaseInstance();
    public SpyApplication GetApplication();
    public SpyDoc GetParent();
    public SpyAuthenticRange GetSelection();
    public void SetSelection( SpyAuthenticRange obj );
    public SpyAuthenticRange GetDocumentBegin();
    public SpyAuthenticRange GetDocumentEnd();
    public SpyAuthenticRange GetWholeDocument();
    public long GetMarkupVisibility();
    public void SetMarkupVisibility( long eSpyAuthenticMarkupVisibility );
    public SpyAuthenticRange GoTo( long eKind, long nCount, long nFrom );
    public void Print( boolean bWithPreview, boolean bPromptUser );
    public boolean Undo();
    public boolean Redo();
    public void UpdateXMLInstanceEntities();

    // Since version 2004R4
    public String GetAsXMLString();
    public void SetAsXMLString( String i_strXML );
    public SpyXMLData GetXMLDataRoot();
    public boolean IsUndoEnabled();
    public boolean IsRedoEnabled();
}
```

SpyDocEditSelection

```
public class SpyDocEditSelection
{
  public void ReleaseInstance();
  public SpyXMLData GetEnd();
  public long GetEndTextPosition();
  public SpyXMLData GetStart();
  public long GetStartTextPosition();
}
```

SpyDocEditView

```
public class SpyDocEditView
{
  public void ReleaseInstance();
  public void ApplyTextState( String sElementName );
  public SpyDocEditSelection GetCurrentSelection();
  public void EditClear();
  public void EditCopy();
  public void EditCut();
  public void EditPaste();
  public void EditRedo();
  public void EditSelectAll();
  public void EditUndo();
  public SpyXMLData GetNextVisible( SpyXMLData oElement );
  public SpyXMLData GetPreviousVisible( SpyXMLData oElement );
  public boolean GetIsEditClearEnabled();
  public boolean GetIsEditCopyEnabled();
  public boolean GetIsEditCutEnabled();
  public boolean GetIsEditPasteEnabled();
  public boolean GetIsEditRedoEnabled();
  public boolean GetIsEditUndoEnabled();
  public boolean GetIsRowAppendEnabled();
  public boolean GetIsRowDeleteEnabled();
  public boolean GetIsRowDuplicateEnabled();
  public boolean GetIsRowInsertEnabled();
  public boolean GetIsRowMoveDownEnabled();
  public boolean GetIsRowMoveUpEnabled();
  public boolean IsTextStateApplied( String sElementName );
  public boolean IsTextStateEnabled( String sElementName );
  public void LoadXML( String sXML );
  public void MarkUpView( long nKind );
  public void RowAppend();
  public void RowDelete();
  public void RowDuplicate();
  public void RowInsert();
  public void RowMoveDown();
  public void RowMoveUp();
  public String SaveXML();
  public void SelectionMoveTabOrder( boolean bForward, boolean bTag );
  public boolean SelectionSet( SpyXMLData oStart, long nStartPos, SpyXMLData
  oEndElement, long nEndPos );
  public SpyXMLData GetXMLRoot();
  public String[] GetAllowedElements( long nAction, SpyXMLData oStartPtr,
  SpyXMLData oEndPtr );
}
```

3.5.30 Predefined constants

This section lists all classes that define the predefined constants used by the Java interface.

SPYApplicationStatus

```
public class SPYApplicationStatus
{
   public final static long spyApplicationStatus_Running         = 0;
   public final static long                                      = 1;
   spyApplicationStatus_AfterLicenseCheck
   public final static long                                      = 2;
   spyApplicationStatus_BeforeLicenseCheck
   public final static long                                      = 3;
   spyApplicationStatus_ConcurrentLicenseCheckFailed
   public final static long                                      = 4;
   spyApplicationStatus_ProcessingCommandLine
}
```

SPYAttributeTypeDefinition

```
public class SPYAttributeTypeDefinition
{
   public final static long          = 0;
   spyMergedGlobal
   public final static long          = 1;
   spyDistinctGlobal
   public final static long spyLocal = 2;
}
```

SPYAuthenticActions

```
public class SPYAuthenticActions
{
   public final static long                  = 0;
   spyAuthenticInsertAt
   public final static long                  = 1;
   spyAuthenticApply
   public final static long                  = 2;
   spyAuthenticClearSurr
   public final static long                  = 3;
   spyAuthenticAppend
   public final static long                  = 4;
   spyAuthenticInsertBefore
   public final static long                  = 5;
   spyAuthenticRemove
}
```

SPYAuthenticDocumentPosition

```
public class SPYAuthenticDocumentPosition
{
```

```
    public final static long                  = 0;
    spyAuthenticDocumentBegin
    public final static long                  = 1;
    spyAuthenticDocumentEnd
    public final static long                  = 2;
    spyAuthenticRangeBegin
    public final static long                  = 3;
    spyAuthenticRangeEnd
}
```

SPYAuthenticElementKind

```
public class SPYAuthenticElementKind
{
   public final static long          = 0;
   spyAuthenticChar
   public final static long          = 1;
   spyAuthenticWord
   public final static long          = 3;
   spyAuthenticLine
   public final static long          = 4;
   spyAuthenticParagraph
   public final static long          = 6;
   spyAuthenticTag
   public final static long          = 8;
   spyAuthenticDocument
   public final static long          = 9;
   spyAuthenticTable
   public final static long          = 10;
   spyAuthenticTableRow
   public final static long          = 11;
   spyAuthenticTableColumn
}
```

SPYAuthenticMarkupVisibility

```
public class SPYAuthenticMarkupVisibility
{
    public final static long                  = 0;
    spyAuthenticMarkupHidden
    public final static long                  = 1;
    spyAuthenticMarkupSmall
    public final static long                  = 2;
    spyAuthenticMarkupLarge
    public final static long                  = 3;
    spyAuthenticMarkupMixed
}
```

SPYDatabaseKind

```
public class SPYLoading
{
```

```
    public final static long           = 0;
    spyDB_Access
    public final static long           = 1;
    spyDB SQLServer
    public final static long           = 2;
    spyDB_Oracle
    public final static long           = 3;
    spyDB_Sybase
    public final static long           = 4;
    spyDB MySQL
    public final static long spyDB_DB2 = 5;
    public final static long           = 6;
    spyDB Other
    public final static long           = 7;
    spyDB_Unspecified
}
```

SPYDialogAction

```
public class SPYDialogAction
{
    public final static long           = 0;
    spyDialogOK
    public final static long           = 1;
    spyDialogCancel
    public final static long           = 2;
    spyDialogUserInput
}
```

SPYDOMType

```
public class SPYDOMType
{
    public final static long          = 0;
    spyDOMType_msxml4
    public final static long          = 1;
    spyDOMType xerces
}
```

SPYDTDSchemaFormat

```
public class SPYDTDSchemaFormat
{
    public final static long spyDTD     = 0;
    public final static long spyDCD     = 1;
    public final static long spyXMLData = 2;
    public final static long spyBizTalk = 3;
    public final static long spyW3C     = 4;
}
```

SPYEncodingByteOrder

```
public class SPYEncodingByteOrder
{
    public final static long spyNONE      = 0;
    public final static long              = 1;
    spyLITTLE ENDIAN
    public final static long              = 2;
    spyBIG_ENDIAN
}
```

SPYExportNamespace

```
public class SPYExportNamespace
{
    public final static long spyNoNamespace                = 0;
    public final static long spyReplaceColonWithUnderscore = 1;
}
```

SPYFindInFilesSearchLocation

```
public class SPYFindInFilesSearchLocation
{
   public final static long              = 0;
   spyFindInFiles Documents
   public final static long              = 1;
   spyFindInFiles_Project
   public final static long              = 2;
   spyFindInFiles_Folder
}
```

SPYFrequentElements

```
public class SPYFrequentElements
{
   public final static long          = 0;
   spyGlobalElements
   public final static long          = 1;
   spyGlobalComplexType
}
```

SPYImageKind

```
public class SPYImageKind
{
   public final static long           = 0;
   spyImageType PNG
   public final static long           = 1;
   spyImageType_EMF
}
```

SPYImportColumnsType

Enter topic text here.

SPYLibType

```
public class SPYLibType
{
   public final static long          = 0;
   spyLibType_static
   public final static long          = 1;
   spyLibType_dll
}
```

SPYLoading

```
public class SPYLoading
{
   public final static long          = 0;
   spyUseCacheProxy
   public final static long          = 1;
   spyReload
}
```

SPYNumberDateTimeFormat

```
public class  SPYNumberDateTimeFormat
{
   public final static long           = 0;
   spySystemLocale
   public final static long           = 1;
   spySchemaCompatible
}
```

SPYProgrammingLanguage

```
public class SPYLoading
{
    public final static long spyUndefinedLanguage = -1;
    public final static long spyJava              = 0;
    public final static long spyCpp               = 1;
    public final static long spyCSharp            = 2;
}
```

SPYProjectItemTypes

```
public class SPYProjectItemTypes
{
   public final static long       = 0;
   spyUnknownItem
   public final static long       = 1;
   spyFileItem
```

```
    public final static long        = 2;
    spyFolderItem
    public final static long        = 3;
    spyURLItem
 }
```

SPYProjectType

```
 public class SPYProjectType
 {
    public final static long             = 0;
    spyVisualStudioProject
    public final static long             = 1;
    spyVisualStudio2003Project
    public final static long             = 2;
    spyBorlandProject
    public final static long             = 3;
    spyMonoMakefile
 }
```

SPYSampleXMLGenerationOptimization

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long                      = 0;
   spySampleXMLGen_Optimized
   public final static long                      = 1;
   spySampleXMLGen NonMandatoryElements
   public final static long                      = 2;
   spySampleXMLGen_Everything
}
```

SPYSampleXMLGenerationSchemaOrDTDAssignment

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long                      = 0;
   spySampleXMLGen AssignRelatively
   public final static long                      = 1;
   spySampleXMLGen_AssignAbsolutely
   public final static long                      = 2;
   spySampleXMLGen_DoNotAssign
}
```

SPYSchemaDefKind

```
 public class SPYSchemaDefKind
 {
    public final static long spyKindElement  = 0;
    public final static long                 = 1;
    spyKindComplexType
    public final static long                 = 2;
    spyKindSimpleType
    public final static long spyKindGroup    = 3;
```

```
   public final static long spyKindModel    = 4;
   public final static long spyKindAny      = 5;
   public final static long spyKindAttr     = 6;
   public final static long                 = 7;
   spyKindAttrGroup
   public final static long spyKindAttrAny  = 8;
   public final static long                 = 9;
   spyKindIdentityUnique
   public final static long                 = 10;
   spyKindIdentityKey
   public final static long                 = 11;
   spyKindIdentityKeyRef
   public final static long                 = 12;
   spyKindIdentitySelector
   public final static long                 = 13;
   spyKindIdentityField
   public final static long spyKindNotation = 14;
   public final static long spyKindInclude  = 15;
   public final static long spyKindImport   = 16;
   public final static long spyKindRedefine = 17;
   public final static long spyKindFacet    = 18;
   public final static long spyKindSchema   = 19;
   public final static long spyKindCount    = 20;
}
```

SPYSchemaDocumentationFormat

```
public class SPYSchemaDocumentationFormat
{
   public final static long           = 0;
   spySchemaDoc HTML
   public final static long           = 1;
   spySchemaDoc_MSWord
   public final static long           = 2;
          spySchemaDoc RTF
   public final static long           = 3;
          spySchemaDoc_PDF
}
```

SPYSchemaExtensionType

```
public class SPYSchemaExtensionType
{
   public final static long                    = 0;
   spySchemaExtension_None
   public final static long                    = 1;
   spySchemaExtension SQL XML
   public final static long                    = 2;
   spySchemaExtension_MS_SQL_Server
   public final static long                    = 3;
   spySchemaExtension Oracle
}
```

SPYSchemaFormat

```
public class SPYSchemaFormat
{
   public final static long                  = 0;
   spySchemaFormat_Hierarchical
   public final static long                  = 1;
   spySchemaFormat Flat
}
```

SPYTextDelimiters

```
public class SPYTextDelimiters
{
    public final static long           = 0;
    spyTabulator
    public final static long           = 1;
    spySemicolon
    public final static long spyComma  = 2;
    public final static long spySpace  = 3;
}
```

SPYTextEnclosing

```
public class SPYTextEnclosing
{
   public final static long       = 0;
   spyNoEnclosing
   public final static long       = 1;
   spySingleQuote
   public final static long       = 2;
   spyDoubleQuote
}
```

SPYTypeDetection

```
public class SPYTypeDetection
{
   public final static long          = 0;
   spyBestPossible
   public final static long          = 1;
   spyNumbersOnly
   public final static long          = 2;
   spyNoDetection
}
```

SPYURLTypes

```
public class SPYURLTypes
{
   public final static long        = (-1);
   spyURLTypeAuto
```

```
    public final static long         = 0;
    spyURLTypeXML
    public final static long         = 1;
    spyURLTypeDTD
}
```

SpyViewModes

```
public class SPYViewModes
{
    public final static long            = 0;
    spyViewGrid
    public final static long            = 1;
    spyViewText
    public final static long            = 2;
    spyViewBrowser
    public final static long            = 3;
    spyViewSchema
    public final static long            = 4;
    spyViewContent
    public final static long            = 4;
    spyViewAuthentic
    public final static long            = 5;
    spyViewWSDL
    public final static long spyViewZIP = 6;
    public final static long            = 7;
    spyViewEditionInfo
}
```

SPYWhitespaceComparison

```
public class SPYWhitespaceComparison
{
    public final static long           = 0;
    spyCompareAsIs
    public final static long           = 1;
    spyCompareNormalized
    public final static long           = 2;
    spyStripAll
}
```

SPYXMLDataKind

```
public class SPYXMLDataKind
{
   public final static long            = 0;
   spyXMLDataXMLDocStruct
   public final static long            = 1;
   spyXMLDataXMLEntityDocStruct
   public final static long            = 2;
   spyXMLDataDTDDocStruct
   public final static long            = 3;
   spyXMLDataXML
   public final static long            = 4;
   spyXMLDataElement
```

```
    public final static long            = 5;
    spyXMLDataAttr
    public final static long            = 6;
    spyXMLDataText
    public final static long            = 7;
    spyXMLDataCData
    public final static long            = 8;
    spyXMLDataComment
    public final static long            = 9;
    spyXMLDataPI
    public final static long            = 10;
    spyXMLDataDefDoctype
    public final static long            = 11;
    spyXMLDataDefExternalID
    public final static long            = 12;
    spyXMLDataDefElement
    public final static long            = 13;
    spyXMLDataDefAttlist
    public final static long            = 14;
    spyXMLDataDefEntity
    public final static long            = 15;
    spyXMLDataDefNotation
    public final static long            = 16;
    spyXMLDataKindsCount
}
```

4 ActiveX Integration

AuthenticDesktopControl is a control that provides a means of integration of the Authentic Desktop user interface and the functionality described in this section into most kinds of applications. ActiveX technology was chosen so as to allow integration using any of a wide variety of languages; this enables C++, C#, VisualBasic, or HTML to be used for integration. ActiveX components officially only work with Microsoft Internet Explorer. All components are full OLE Controls, which makes integration as simple as possible. Two different levels of integration are provided, thus enabling the integration to be adapted to a wide range of needs.

To integrate Authentic Desktop you must install the Authentic Desktop Integration Package. Ensure that you install Authentic Desktop first, and then the Authentic Desktop Integration Package.

For a successful integration you have to consider the following main design factors:

- What technology or programming language can the hosting application use to integrate the AuthenticDesktopControl?
- Should the integrated UI look exactly like Authentic Desktop with all its menus, toolbars, and windows, or will a subset of these elements—like allowing only one document and a restricted set of commands—be more effective?
- How deep will the integration be? Should the Authentic Desktop user interface be used as is? Are user interface extensions and/or restrictions required? Can some frequently used tasks be automated?

The sections, Integration at the Application Level and Integration at Document Level, both of which have examples in various programming languages, will help you to make the right decisions quickly. The section, Object Reference, describes all COM objects that can be used for integration, together with their properties and methods.

For automation tasks, the Authentic Desktop Automation Interface is accessible from the AuthenticDesktopControl as well.

For information about how to integrate Authentic Desktop into Microsoft Visual Studio see the section, Authentic Desktop in Visual Studio.

4.1 Integration at Application Level

Integration at application level is simple and straightforward. It allows you to embed the complete interface of Authentic Desktop into a window of your application. Since you get the whole user interface of Authentic Desktop, you get 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`. Its property `IntegrationLevel` defaults to application-level. You may use `Appearance` and `BorderStyle` to configure the appearance of the control's wrapper window. Do not instantiate or access `AuthenticDesktopControlDocument` or `AuthenticDesktopControlPlaceHolder` ActiveX controls when integrating at application-level.

If you have any initialization to do or if you want to automate some behaviour of Authentic Desktop, use the properties, methods, and events described for `AuthenticDesktopControl`. Consider using `AuthenticDesktopControl.Application` for more complex access to Authentic Desktop functionality.

In this section is an example (Example: HTML) showing how the Authentic Desktop application can be embedded in an HTML page. For usage with other programming languages, or more sophisticated access, see the Examples of integration at document-level.

4.1.1 Example: HTML

This example shows a simple integration of the Authentic Desktop control at application-level into a HTML page. The integration is described in the following sections:

- Instantiate a AuthenticDesktopControl in HTML code.
- Implement buttons to load documents and automate code-generation tasks.
- Define actions for some application events.

The code for this example is available at the following location in your Authentic Desktop installation:
`Examples\ActiveX\HTML\AuthenticActiveX_ApplicationLevel.htm`

Note: This example works only in Internet Explorer.

Instantiate the Control

The HTML `Object` tag is used to create an instance of the AuthenticDesktopControl. The `Classid` is that of AuthenticDesktopControl. Width and height specify the window size. No additional parameters are necessary, since application-level is the default.

Add Button to Open Default Document

As a simple example of how to automate some tasks, we add a button to the page:

```
<input type="button" value="Open Marketing Expenses"
onclick="BtnOpenMEFile()">
```

When clicked, a predefined document will be opened in the AuthenticDesktopControl. We use a method to locate the file relative to the AuthenticDesktopControl so the example can run on different installations.

Connect to Custom Events

The example implements two event callbacks for AuthenticDesktopControl custom events to show the principle:

```
<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentOpened" of AuthenticDesktopControl object  -->
<SCRIPT LANGUAGE="javascript">
       function objAuthenticDesktopControl::OnDocumentOpened( objDocument )
       {
       // alert("Document '" + objDocument.Name + "' opened!");
       }
</SCRIPT>

<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentClosed" of AuthenticDesktopControl object  -->
<SCRIPT LANGUAGE="javascript">
       function objAuthenticDesktopControl::OnDocumentClosed( objDocument )
       {
       // alert("Document '" + objDocument.Name + "' closed!");
       }
</SCRIPT>
```

4.2 Integration at Document Level

Integration at document level gives you freedom over instantiation and placement of the following parts of the Authentic Desktop user interface:

If necessary, a replacement for the menus and toolbars of Authentic Desktop must be provided by your application.

You will need to instantiate and access multiple ActiveX controls, depending on which user interface parts you want to re-use. All these controls are contained in the AuthenticDesktopControl OCX.

- Use AuthenticDesktopControl to set the integration level and access application wide functionality.
- Use AuthenticDesktopControlDocument to create any number of editor windows. It may be sufficient to create only one window and re-use it, depending on your needs.
- Optionally Use AuthenticDesktopControlPlaceholder to embed Authentic Desktop entry helper windows, validator output or other windows mentioned above.
- Access run-time information about commands, menus, and toolbars available in AuthenticDesktopControl to seamlessly integrate these commands into your application's menus and toolbars. See Query Authentic Desktop Commands for more information.

If you want to automate some behaviour of Authentic Desktop use the properties, methods, and events described for the AuthenticDesktopControl, AuthenticDesktopControlDocument and AuthenticDesktopControlPlaceHolder. Consider using AuthenticDesktopControl.Application, AuthenticDesktopControlDocument.Document and AuthenticDesktopControlPlaceHolder.Project for more complex access to Authentic Desktop functionality. However, to open a document always use AuthenticDesktopControlDocument.Open or AuthenticDesktopControlDocument.New on the appropriate document control. To open a project always use AuthenticDesktopControlPlaceHolder.OpenProject on a placeholder control embedding a Authentic Desktop project window.

See Examples on how to instantiate and access the necessary controls in different programming environments.

4.2.1 Use AuthenticDesktopControl

To integrate at document level, instantiate a AuthenticDesktopControl first. Set the property IntegrationLevel to IActiveXIntegrationOnDocumentLevel(=1) . Set the window size of the embedding window to 0x0 to hide any user interface behind the control. You may use Appearance and BorderStyle to configure the appearance of the control's wrapper window.

Avoid using the method Open since this might lead to unexpected results. Use the corresponding open methods of AuthenticDesktopControlDocument and AuthenticDesktopControlPlaceHolder, instead.

See Query Authentic Desktop Commands for a description of how to integrate Authentic Desktop commands into your application. Send commands to Authentic Desktop via the method Exec. Query if a command is currently enabled or disabled using the method QueryStatus.

4.2.2 Use AuthenticDesktopControlDocument

An instance of the `AuthenticDesktopControlDocument` ActiveX control allows you to embed one Authentic Desktop document editing window into your application. You can use any number of instances you need.

Use the method `Open` to load any other existing file.

The control does not support a read-only mode. The value of the property `ReadOnly` is ignored.

Use `Path` and `Save` or methods and properties accessible via the property `Document` to access document functionality.

4.2.3 Use AuthenticDesktopControlPlaceHolder

Instances of AuthenticDesktopControlPlaceHolder ActiveX controls allow you to selectively embed the additional helper windows of Authentic Desktop into your application. The property `PlaceholderWindowID` selects the Authentic Desktop helper window to be embedded. Use only one AuthenticDesktopControlPlaceHolder for each window identifier. See for valid window identifiers.

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.

4.2.4 Query Authentic Desktop Commands

When integrating at document-level, no menu or toolbar from Authentic Desktop is available to your application. Instead, you can query all the commands and the structure of the application menu at runtime. Professional applications will need to integrate this menu in a sophisticated manner into their own menu structure. Your installation of Authentic Desktop even provides you with command label images used within Authentic Desktop. See the folder `Examples\ActiveX\Images` of your Authentic Desktop installation for icons in GIF format. The file names correspond to the `labels` of commands.

4.2.5 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 `Examples\ActiveX` of your Authentic Desktop installation.

C#

The C# example shows how to integrate the AuthenticDesktopControl in a common desktop application created with C# using Visual Studio 2008.
Please note that the example application is already complete. There is no need to change anything if you want to run and see it working. The following steps describe what general actions and considerations must be taken in order to create a project such as this.

Introduction

Adding the Authentic Desktop components to the Toolbox
Before you take a look at the sample project please add the assemblies to the .NET IDE Toolbox. The Authentic Desktop Installer will have already installed the assemblies in the .NET Global Assembly Cache (GAC). If you open the Toolbox dialog under **Tools | Add/Remove Toolbox Items** the controls will appear as `AxAuthenticDesktopControl`, `AxAuthenticDesktopControlDocument` and `AxAuthenticDesktopControlPlaceholder` on the .NET Framework Components tab. Check all to make them available to the IDE.

Placing the AuthenticDesktopControl

It is necessary to have one AuthenticDesktopControl instance to set the integration level and to manage the Document and Placeholder controls of the Authentic Desktop library. The control is accessible via the General section of the Toolbox helper window in the IDE. To add it you need to select the component in the Toolbox window and drag a rectangle wherever you want to have it in the destination window. If you have an application which does not open a window on startup you can use a simple invisible Form with the control on it which is created manually in the code.

The example project adds this instance to the main MdiContainer MDIMain. If you open MDIMain in the Design View from the Solution Explorer you will see a light blue rectangle at the top-left side in the client area of the Frame window. Selecting this rectangle will show you the properties of the AuthenticDesktopControl. It is important to set the `IntegrationLevel` property to **`ICActiveXIntegrationOnDocumentLevel`** in order to turn on the Document and Placeholder support of the Authentic Desktop library.

HTML

This example shows an integration of the Authentic Desktop control at document-level into a HTML page. The following topics are covered:

- Instantiate a AuthenticDesktopControl ActiveX control object in HTML code
- Instantiate a AuthenticDesktopControlDocument ActiveX control to allow editing a Authentic Desktop file
- Instantiate one AuthenticDesktopControlPlaceHolder for a AuthenticDesktopControl project window
- Instantiate one AuthenticDesktopControlPlaceHolder to alternatively host one of the Authentic Desktop helper windows

- Create a simple customer toolbar for some heavy-used Authentic Desktop commands
- Add some more buttons that use the COM automation interface of Authentic Desktop
- Use event handlers to update command buttons

This example is available in its entirety in the file `Authentic Desktop ActiveX_ApplicationLevel.htm` within the `C:\Program Files\Altova\Authentic Desktop2012\Examples\ActiveX\HTML\` folder of your Authentic Desktop installation.

Note: This example works only in Internet Explorer.

Instantiate the AuthenticDesktopControl

AuthenticDesktopControlThe HTML `OBJECT` tag is used to create an instance of the AuthenticDesktopControl. The Classid is that of AuthenticDesktopControl. Width and height are set to 0 since we use this control as manager control without use for its user interface. The integration level is specified as a parameter within the `OBJECT` tag.

```
<OBJECT id=""
        width="0"
        height="0"
        VIEWASTEXT>
   <PARAM NAME="IntegrationLevel" VALUE="1">
</OBJECT>
```

Create Editor Window

The HTML `OBJECT` tag is used to embed an editing window.

```
<OBJECT id="objDoc1"
        width="600"
        height="500"
        VIEWASTEXT>
</OBJECT>
```

Create Project Window

The HTML `OBJECT` tag is used to create a AuthenticDesktopControlPlaceHolder window. The first additional custom parameter defines the placeholder to show the Authentic Desktop project window. The second parameter loads one of the example projects delivered with your Authentic Desktop installation (located in the `<yourusername>/MyDocuments` folder).

```
<OBJECT id="objProjectWindow"
        width="200"
        height="200"
        VIEWASTEXT>
</OBJECT>
```

Create Placeholder for Helper Windows

The HTML `OBJECT` tag is used to instantiate a AuthenticDesktopControlPlaceHolder ActiveX control that can host the different Authentic Desktop helper windows. Initially, no helper window is shown. See the example file.

```
<OBJECT id="objEHWindow"
        width="200"
        height="200"
        VIEWASTEXT>
   <PARAM name="PlaceholderWindowID" value="">
</OBJECT>
```

Three buttons allow us to switch the actual window that will be shown. The JavaScript execute on-button-click sets the property `PlaceHolderWindowID` to the corresponding value defined in

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
//
--------------------------------------------------------------------------------
// specify which of the windows shall be shown in the placeholder control.
function BtnHelperWindow(i_ePlaceholderWindowID)
{
   objEHWindow.PlaceholderWindowID = i_ePlaceholderWindowID;
}
</SCRIPT>
```

4.3 Command Table for Authentic Desktop

Tables in this section list the names and identifiers of all commands that are available within Authentic Desktop. Every sub-section lists the commands from the corresponding top-level menu of Authentic Desktop. The left-most column shows the command's menu text to make it easier for you to identify the functionality behind the command. The last sub-section is a collection of those commands that are not accessible via the main menu.

See Query Authentic Desktop Commands on how to query the current resource structure and command availability.

Use the command identifiers with AuthenticDesktopControl.QueryStatus or AuthenticDesktopControlDocument.QueryStatus to check the current status of a command. Use AuthenticDesktopControl.Exec or AuthenticDesktopControlDocument.Exec to execute a command.

4.3.1 File Menu

File menu commands:

New...	ID_FILE_NEW	57600
Open...	ID_FILE_OPEN	57601
Open Global Resource...	IDC_OPEN_GLOBALRESOURCE	34112
Reload	IDC_FILE_RELOAD	34065
Encoding...	IDC_ENCODING	34061
Close	ID_FILE_CLOSE	57602
Close All	IDC_CLOSE_ALL	34050
Save	ID_FILE_SAVE	57603
Save As...	ID_FILE_SAVE_AS	57604
Save to URL...	ID_FILE_SAVE_TO_URL	34209
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

4.3.2 Edit Menu

Edit menu commands:

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

4.3.3 Project Menu

Project menu commands:

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
Source Control/Open Project...	IDC_SCC_OPEN_PROJECT	34140
Source Control/Enable Source Code Control	IDC_SCC_ENABLE	34137
Source Control/Get latest version...	IDC_SCC_GET	34138
Source Control/Check Out...	IDC_SCC_CHECK_OUT	34135
Source Control/Check In...	IDC_SCC_CHECK_IN	34134
Source Control/Undo Check Out...	IDC_SCC_UNDO_CHECK_OUT	34145
Source Control/Add to Source Control	IDC_SCC_ADD	34133
Source Control/Remove from Source Control	IDC_SCC_REMOVE	34143
Source Control/Show History...	IDC_SCC_HISTORY	34139
Source Control/Show Differences...	IDC_SCC_DIFF	34136
Source Control/Properties...	IDC_SCC_PROPERTIES	34141
Source Control/Refresh Status...	IDC_SCC_REFRESH	34142
Source Control/Run Native Interface...	IDC_SCC_RUN	34144
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
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
Properties...	IDC_ICPROJECTGUI_PROJECT_PROPERTIES	37223
Recent Project	IDC_ICPROJECTGUI_RECENT	37224

4.3.4 XML Menu

XML menu commands:

Check well-formedness	IDC_CHECK_WELL_FORM	34049
Validate XML	IDC_VALIDATE	32954

4.3.5 XSL/XQuery Menu

XSL/XQuery menu commands:

XSL Transformation	IDC_TRANSFORM_XSL	33006
XSL:FO Transformation	IDC_TRANSFORM_XSLFO	33007
XSL Parameters/XQuery Variables...	IDC_TRANSFORM_XSL_PARAMS	33008

4.3.6 Authentic Menu

Authentic menu commands:

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
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_MIX	32857
Append row	IDC_ROW_APPEND	32806
Insert row	IDC_ROW_INSERT	32809
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

4.3.7 View Menu

View menu commands:

WSDL Design view	IDC_VIEW_WSDL	34117
XBRL Taxonomy view	IDC_VIEW_XBRL	34118
Authentic view	IDC_VIEW_CONTENT	34177
Browser view	IDC_VIEW_BROWSER	34176
Text View Settings	IDC_TEXTVIEW_SETTINGS	34119

4.3.8 Browser Menu

Browser menu commands:

Back	IDC_BROWSER_BACK	34039
Forward	IDC_BROWSER_FORWARD	34045
Stop	IDC_BROWSER_STOP	34047
Refresh	IDC_BROWSER_REFRESH	34046
Fonts/Largest	IDC_BROWSER_FONT_LARGEST	34041
Fonts/Larger	IDC_BROWSER_FONT_LARGE	34040
Fonts/Medium	IDC_BROWSER_FONT_MEDIUM	34042
Fonts/Smaller	IDC_BROWSER_FONT_SMALL	34043
Fonts/Smallest	IDC_BROWSER_FONT_SMALLEST	34044
Separate window	IDC_BROWSER_USE_OWN_FRAME	34048

4.3.9 Tools Menu

Tools menu command:

Spelling...	IDC_SPELL_CHECK	34154
Spelling options...	IDC_SPELL_OPTIONS	34155
Switch to Scripting environment...	ID_WINDOW_SWITCHTOVBA	34231
Show macros...	ID_WINDOW_VBAMACROS	34232
Project/Assign Scripts to Project	ID_SCRIPTINGPRJ_ASSIGN	34214
Project/Unassign Scripts from Project	ID_SCRIPTINGPRJ_UNASSIGN	34215
Project/Project Scripts active	ID_SCRIPTINGPRJ_ACTIVE	34213
Global Resources	IDC_GLOBALRESOURCES	37401
Active Configuration/<plugin not loaded>	IDC_GLOBALRESOURCES_SUBMENUENTRY1	37408
Customize...	IDC_CUSTOMIZE	34055
Options...	IDC_SETTINGS	33300
<placeholder>	ID_SCRIPTING_MACROITEMS	34249

4.3.10 Window Menu

Window menu commands:

Cascade	ID_WINDOW_CASCADE	57650
Tile horizontally	ID_WINDOW_TILE_HORZ	57651
Tile vertically	ID_WINDOW_TILE_VERT	57652
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

4.3.11 Help Menu

Help menu commands:

Table of Contents...	IDC_HELP_CONTENTS	34076
Index...	IDC_HELP_INDEX	34077
Search...	IDC_HELP_SEARCH	34079
Keyboard Map...	IDC_HELP_KEYMAPDLG	34078
Software Activation...	IDC_ACTIVATION	34005
Order Form...	IDC_OPEN_ORDER_PAGE	34094
Registration...	IDC_REGISTRATION	34131
Check for Updates...	IDC_CHECK_FOR_UPDATES	34275
Support Center...	IDC_OPEN_SUPPORT_PAGE	34096
FAQ on the Web...	IDC_SHOW_FAQ	34153
Download Components and Free Tools	IDC_OPEN_COMPONENTS_PAGE	34093
Authentic Desktop on the Internet..	IDC_OPEN_AUTHENTIC_HOME	34098
Authentic Desktop Training...	ID_HELP_AUTHENTICTRAINING	34210
About Authentic Desktop	ID_APP_ABOUT	57664

4.3.12 Misc Menu

Miscellaneous menu and context menu commands:

Database	IDC_ADDRESOURCE_DATABASE	37405
File	IDC_ADDRESOURCE_FILE	37403
Folder	IDC_ADDRESOURCE_FOLDER	37404
Attribute	IDC_ADD_CHILD_ATTRIBUTE	33402
CData	IDC_ADD_CHILD_CDATA	33403
Comment	IDC_ADD_CHILD_COMMENT	33404
ATTLIST	IDC_ADD_CHILD_DEF_ATTLIST	33405
DOCTYPE	IDC_ADD_CHILD_DEF_DOCTYPE	33406
ELEMENT	IDC_ADD_CHILD_DEF_ELEMENT	33407
ENTITY	IDC_ADD_CHILD_DEF_ENTITY	33408
ExternalID	IDC_ADD_CHILD_DEF_EXTERNAL_ID	33409
NOTATION	IDC_ADD_CHILD_DEF_NOTATION	33410
Processing Instruction	IDC_ADD_CHILD_PI	33411
Element	IDC_ADD_CHILD_STRUCT	33412
Text	IDC_ADD_CHILD_TEXT	33413
XInclude...	IDC_ADD_CHILD_XINCLUDE	34027
XML	IDC_ADD_CHILD_XML	33414
Attribute	IDC_APPEND_ATTRIBUTE	33415
CData	IDC_APPEND_CDATA	33416
Comment	IDC_APPEND_COMMENT	33417
ATTLIST	IDC_APPEND_DEF_ATTLIST	33418
DOCTYPE	IDC_APPEND_DEF_DOCTYPE	33419
ELEMENT	IDC_APPEND_DEF_ELEMENT	33420
ENTITY	IDC_APPEND_DEF_ENTITY	33421
ExternalID	IDC_APPEND_DEF_EXTERNAL_ID	33422
NOTATION	IDC_APPEND_DEF_NOTATION	33423
Processing Instruction	IDC_APPEND_PI	33424
Element	IDC_APPEND_STRUCT	33425
Text	IDC_APPEND_TEXT	33426
XInclude...	IDC_APPEND_XINCLUDE	34026
XML	IDC_APPEND_XML	33427
Assign DTD...	IDC_ASSIGN_DTD	34032

Assign sample XML file...	IDC_ASSIGN_SAMPLE_XML	33000
Assign Schema...	IDC_ASSIGN_SCHEMA	34033
Assign a StyleVision Stylesheet...	IDC_ASSIGN_SPS	34034
Assign XSL...	IDC_ASSIGN_XSL	33001
Assign XSL:FO...	IDC_ASSIGN_XSLFO	33002
Collapse unselected	IDC_COLLAPSE_UNSELECTED	33428
Convert all to Global Resources	IDC_CONVERT_ALL_DATASOURCES_TO_GLOBAL_RESOURCES	36687
Convert to Global Resource	IDC_CONVERT_DATASOURCE_TO_GLOBAL_RESOURCES	36684
Map to other DTD/Schema or DB in MapForce	IDC_DTD_OPENIN_MAPFORCE	34056
Design HTML/PDF Output in StyleVision	IDC_DTD_OPENIN_STYLEVISION	34057
Edit Global Resource...	IDC_EDIT_GLOBAL_RESOURCES	36686
Insert file path...	IDC_EDIT_INSERT_PATH_STRING	34013
Insert XInclude...	IDC_EDIT_INSERT_XINCLUDE_STRING	34017
Enclose in Element	IDC_ENCLOSE_IN_ELEMENT	33446
Check In...	IDC_FILE_CHECK_IN	32951
Check Out...	IDC_FILE_CHECK_OUT	32952
Undo Check Out...	IDC_FILE_UNDO_CHECK_OUT	32953
Find in files...	IDC_FIND_IN_FILES	34000

Flush memory cache	IDC_FLUSH_CACHED_FILES	34066
Generate Program Code...	IDC_GENERATE_CODE_FROM_SCHEMA	34067
Browse...	IDC_GLOBALRESOURCESUI_CHOOSEFILEASFILE	37420
Choose another Global Resource...	IDC_GLOBALRESOURCESUI_CHOOSEFILEASGR	37419
Add configuration	IDC_GLOBALRESOURCESUI_DETAILS_ADDCONFIG	37423
Add a configuration copy	IDC_GLOBALRESOURCESUI_DETAILS_ADDCONFIGCOPY	37424
Delete configuration	IDC_GLOBALRESOURCESUI_DETAILS_DELETECONFIG	37425
Active Global Resource Configuration	IDC_GLOBALRESOURCES_ACTIVECONFIG	37400
Delete	IDC_GLOBALRESOURCES_MAINDLG_DELETERESOURCE	37422
Go to Next Bookmark	IDC_GOTONEXTBOOKMARK	34070
Go to Previous Bookmark	IDC_GOTOPREVBOOKMARK	34071
Go to Definition	IDC_GOTO_DEFINITION	33447
Go to DTD	IDC_GOTO_DTD	34072
Go to File	IDC_GOTO_FILE	33448

Go to line/char	IDC_GOTO_LINE	34073
Go to Schema	IDC_GOTO_SCHEMA	34074
Go to XSL	IDC_GOTO_XSL	33004
Display as Table	IDC_GRID_VIEW_AS_TABLE	34075
Edit a SELECT Statement...	IDC_ICDBWNDS_EDIT_LOCALVIEW	36676
Remove SELECT statement	IDC_ICDBWNDS_REMOVE_LOCALVIEW	36680
Add a SELECT Statement...	IDC_ICDBWND_CREATE_LOCALVIEW	36677
Generate and add a SELECT Statement...	IDC_ICDBWND_GENEATE_SQL_FOR_LOCALVIEW	36678
Remove from Online Browser	IDC_ICDBWND_REMOVE_FROM_BROWSERVIEW	36683
Add Active and Related Files	IDC_ICPROJECTGUI_ADD_ACTIVE_AND_RELATED_FILES	37217
Add Active File	IDC_ICPROJECTGUI_ADD_ACTIVE_FILE	37216

Add External Folder...	IDC_ICPROJECTGUI_ADD_EXT_FOLDER	37219
Add External Web Folder...	IDC_ICPROJECTGUI_ADD_EXT_URL_FOLDER	37220
Add Files...	IDC_ICPROJECTGUI_ADD_FILES	37213
Add Project Folder...	IDC_ICPROJECTGUI_ADD_FOLDER	37218
Add Global Resource...	IDC_ICPROJECTGUI_ADD_GLOBAL_RESOURCE	37238
Add URL...	IDC_ICPROJECTGUI_ADD_URL	37214
Properties...	IDC_ICPROJECTGUI_PROPERTIES	37222
Refresh	IDC_ICPROJECTGUI_REFRESH_EXT_FOLDER	37221
Include another DTD...	IDC_INCLUDE_DTD	34084
Move left	IDC_MOVE_LEFT	34091
Move right	IDC_MOVE_RIGHT	34092
Namespace prefix...	IDC_NAMESPACE	33462
Optimal widths	IDC_OPTIMAL_WIDTHS	34099
0 or More Matches	IDC_RECM_0MATCH	33901
1 or More Matches	IDC_RECM_1MATCH	33902
Any Character	IDC_RECM_ANYCHAR	33903
Character in Range	IDC_RECM_CHARINRANGE	33904
Character Not in Range	IDC_RECM_CHARNOTINRANGE	33905
End of Line	IDC_RECM_ENDLINE	33906
End of Word	IDC_RECM_ENDWORD	33907
Beginning of Line	IDC_RECM_STARTLINE	33908

Beginning of Word	IDC_RECM_STARTWORD	33909
Tagged Expression	IDC_RECM_TAGGEDEXP	33910
Collapse -	IDC_SEL_COLLAPSE	34151
Expand +	IDC_SEL_EXPAND	34152
Expand fully	IDC_SEL_EXPAND_ALL	33463

Append Row	IDC_TABLE_APPEND_ROW	34157
Insert Row	IDC_TABLE_INSERT_ROW	34158
Ascending Sort	IDC_TABLE_SORT_ASC	33464
Descending Sort	IDC_TABLE_SORT_DESC	33465
Insert/Remove Bookmark	IDC_TOGGLE_BOOKMARK	34162
Comment In/Out	IDC_TOGGLE_XML_COMMENT	34029
Update Entry Helpers	IDC_UPDATE_ELEMENT_CHOICE	34173
Messages window	IDC_VALIDATOR_OUTPUT	34175
Word Wrap	IDC_WORD_WRAP	34181
Add Active File to Project	ID_ADDACTIVEFILETOPROJECT	36549
Add Files to Project ...	ID_ADDFILESTOPROJECT	36550
Create Folder	ID_CREATEFOLDER	36555
Disconnect	ID_DISCONNECT	36559
Disconnect from all Data Sources	ID_DISCONNECTFROMALLDATASOURCES	36560
Edit Data	ID_EDITRESULTDATA	36646
Rename	ID_EDIT_RENAME	36563
Open Project	ID_FILE_LOAD_PROJECT	36565
Adds a configuration	ID_GLOBALRESOURCES_ADDCONFIG	37429
Adds a configuration as copy of the currently selected configuration	ID_GLOBALRESOURCES_ADDCONFIGCOPY	37430
Deletes a configuration	ID_GLOBALRESOURCES_DELCONFIG	37431
Show referenced table	ID_ICDBWNDS_FGNKEY_GOTO_REFERENCE	36567
View in XMLSpy	ID_ICDBWNDS_VIEW_XMLSCHEMA_IN_XMLSPY	36500
Add a New Data Source	ID_ICDBWND_ADDANEWDATASOURCE	36568
Add to Design Editor	ID_ICDBWND_BROWSER_ADD_TO_DESIGNVIEW	36569
Add to/Remove from Favorites	ID_ICDBWND_BROWSER_ADD_TO_FAVOURITES	36570
Refresh	ID_ICDBWND_BROWSER_REFRESH_ROOT	36571

All	ID_ICDBWND_BROWSER_SEARCH_MODE_ALL	36575
From current DataSource	ID_ICDBWND_BROWSER_SEARCH_MODE_DATASOURCE	36576

From focused item	ID_ICDBWND_BROWSER_SEARCH_MODE_FOCUSED_ITEM	36577
Show in new Design Editor	ID_ICDBWND_BROWSER_SHOW_IN_DESIGNVIEW	36578
Check	ID_ICDBWND_CHECK	36579
Check Children	ID_ICDBWND_CHECK_ALL	36580
Clear	ID_ICDBWND_CLEAR_ROWCOUNT	36538
Children	ID_ICDBWND_COLLAPSE_CHILDREN	36581
Siblings	ID_ICDBWND_COLLAPSE_SIBLING	36582
Execute SQL	ID_ICDBWND_EXECUTE	36583
Children	ID_ICDBWND_EXPAND_CHILDREN	36584
Siblings	ID_ICDBWND_EXPAND_SIBLINGS	36585
Export database data	ID_ICDBWND_EXPORT	36586
Add	ID_ICDBWND_FILEDSN_ADD	36588
Delete	ID_ICDBWND_FILEDSN_DELETE	36589
Contains	ID_ICDBWND_FILTER_CONTAINS	36591
Does not contain	ID_ICDBWND_FILTER_DOES_NOT_CONTAIN	36592
Ends with	ID_ICDBWND_FILTER_ENDS_WITH	36593
Equals	ID_ICDBWND_FILTER_EQUALS	36594
No Filter	ID_ICDBWND_FILTER_INCACTIVE	36596
Starts with	ID_ICDBWND_FILTER_STARTS_WITH	36597
Connect	ID_ICDBWND_MENU_DATASOURCE_CONNECT	36605
Disconnect	ID_ICDBWND_MENU_DATASOURCE_DISCONNECT	36606
Get Tables	ID_ICDBWND_MENU_DATASOURCE_GETTABLES	36607
Preview	ID_ICDBWND_PREVIEWITEM	36608
Remove all favorites	ID_ICDBWND_REMOVE_ALL_FAVORITES	36609
Toggle	ID_ICDBWND_TOGGLE	36610
Uncheck	ID_ICDBWND_UNCHECK	36611

Uncheck Children	ID_ICDBWND_UNCHECK_ALL	36612
Show/Update	ID_ICDBWND_UPDATE_ROWCOUNT	36537
Open	ID_OPEN	36619
Add file extension	ID_POPUP_ADDEXTENSION	35003
Add search path	ID_POPUP_ADDSEARCHPATH	35004
Delete file extension	ID_POPUP_DELETEEXTENSION	35005
Delete search path	ID_POPUP_DELETESEARCHPATH	35006
Edit file extension	ID_POPUP_EDITEXTENSION	35007

Edit search path	ID_POPUP_EDITSEARCHPATH	35008
Reload search paths	ID_POPUP_RELOAD_SEARCH_PATHS	35009
Reset search paths	ID_POPUP_RESET_SEARCH_PATHS	33810
Start Debugger/Go	ID_PROCESS_XSL	34212
New Project	ID_PROJECT_NEWPROJECT	36620
Append	ID_RELDISP_APPEND_LAYER	33201
Delete	ID_RELDISP_DELETE_LAYER	33202
Focus next item on active layer	ID_RELDISP_FOCUS_NEXT_ITEM_ON_LAYER	33227
Focus previous item on active layer	ID_RELDISP_FOCUS_PREV_ITEM_ON_LAYER	33228
Selected Layer	ID_RELDISP_HIDE_LAYER	33203
Others	ID_RELDISP_HIDE_LAYER_OTHER	33204
Insert	ID_RELDISP_INSERT_LAYER	33205
Selected Layer	ID_RELDISP_LOCK_LAYER	33206
Others	ID_RELDISP_LOCK_LAYER_OTHER	33207
Move Back	ID_RELDISP_MOVE_LAYER_ITEMS_BACK	33208
Move Front	ID_RELDISP_MOVE_LAYER_ITEMS_FRONT	33209
Selected Layer	ID_RELDISP_PALE_LAYER	33210
Others	ID_RELDISP_PALE_LAYER_OTHER	33211
Selected Layer	ID_RELDISP_RECOLOR_LAYER	33212
Others	ID_RELDISP_RECOLOR_LAYER_OTHER	33213
Rename	ID_RELDISP_RENAME_LAYER	33214
Reset all layer states	ID_RELDISP_RESET_LAYER_STATES	33231
Select	ID_RELDISP_SELECT_ITEMS	33215
Select Others	ID_RELDISP_SELECT_OTHER_ITEMS	33216
Selected Layer	ID_RELDISP_SHOW_LAYER	33217
Show layer item count	ID_RELDISP_SHOW_LAYER_ITEM_COUNT	33229
Others	ID_RELDISP_SHOW_LAYER_OTHER	33218
Selected Layer	ID_RELDISP_TOGGLE_COLORING	33219
Others	ID_RELDISP_TOGGLE_COLORING_OTHER	33220
Selected Layer	ID_RELDISP_TOGGLE_LAYER_VISIBILITY	33221
Others	ID_RELDISP_TOGGLE_LAYER_VISIBILITY_OTHER	33222
Selected Layer	ID_RELDISP_TOGGLE_LOCK	33223
Others	ID_RELDISP_TOGGLE_LOCK_OTHER	33224
Selected Layer	ID_RELDISP_UNLOCK_LAYER	33225

Others	ID_RELDISP_UNLOCK_LAYER_OTHER	33226
Remove all Data Sources	ID_REMOVEALLDATASOURCES	36621
Remove	ID_REMOVE_DATASOURCE	36622
Remove from Favorites	ID_REMOVE_FAVORITE_ITEM	36623
Remove	ID_REMOVE_FROM_PROJECT	36624
All rows	ID_RETRIEVE_ALLROWS	36625
First n rows	ID_RETRIEVE_FIRSTROWS	36626
Save Project	ID_SAVEPROJECT	36627
Save Project As...	ID_SAVEPROJECTAS	36628

4.4 Accessing AuthenticDesktopAPI

The focus of this documentation is the ActiveX controls and interfaces required to integrate the Authentic Desktop user interface into your application. To allow you to automate or control the functionality of the integrated components, the following properties give you access to the Authentic Desktop automation interface (AuthenticDesktopAPI):

AuthenticDesktopControl.Application
AuthenticDesktopControlDocument.Document
AuthenticDesktopControlPlaceHolder.Project

Some restrictions apply to the usage of the Authentic Desktop automation interface when integrating AuthenticDesktopControl at document-level. See Integration at document level for details.

4.5 Object Reference

Objects:
Authentic DesktopCommand
Authentic DesktopCommands
AuthenticDesktopControl
AuthenticDesktopControlDocument
AuthenticDesktopControlPlaceHolder

To give access to standard Authentic Desktop functionality, objects of the **Authentic Desktop automation interface** can be accessed as well. See AuthenticDesktopControl.Application, AuthenticDesktopControlDocument.Document and AuthenticDesktopControlPlaceHolder.Project for more information.

4.5.1 Authentic DesktopCommand

Properties:
ID
Label
IsSeparator
ToolTip
StatusText
Accelerator
SubCommands

Description:
Each Command object can be one of three possible types:

- **Command**: ID is set to a value greater 0 and Label is set to the command name. IsSeparator is false and the SubCommands collection is empty.
- **Separator**: IsSeparator is true. ID is 0 and Label is not set. The SubCommands collection is empty.
- (**Sub**) **Menu**: The SubCommands collection contains Command objects and Label is the name of the menu. ID is set to 0 and IsSeparator is false.

Accelerator

Property: Label as string

Description:
For command objects that are children of the ALL_COMMANDS collection, this is the accelerator key defined for the command. If the command has no accelerator key assigned, this property returns the empty string.

The string representation of the accelerator key has the following format:

```
[ALT+][CTRL+][SHIFT+]key
```

Where key is converted using the Windows Platform SDK function GetKeyNameText.

ID

Property: ID as long

Description:
ID is 0 for separators and menus.
For commands, this is the ID which can be used with Exec and QueryStatus.

IsSeparator

Property: IsSeparator as boolean

Description:
True if the command is a separator.

Label

Property: `Label` as string

Description:
Label is empty for separators.
For command objects that are children of the ALL_COMMANDS collection, this is a unique name. Command icons are stored in files with this name. See Query Commands for more information.

For command objects that are children of menus, the label property holds the command's menu text.
For sub-menus, this property holds the menu text.

StatusText

Property: `Label` as string

Description:
For command objects that are children of the ALL_COMMANDS collection, this is the text shown in the status bar when the command is selected.

SubCommands

Property: `SubCommands` as `Commands`

Description:
The `SubCommands` collection holds any sub-commands if this command is actually a menu or submenu.

ToolTip

Property: `ToolTip` as string

Description:
For command objects that are children of the ALL_COMMANDS collection, this is the text shown as tool-tip.

4.5.2 Authentic DesktopCommands

Properties:
Count
Item

Description:
Collection of Command objects to get access to command labels and IDs of the AuthenticDesktopControl. Those commands can be executed with the Exec method and their status can be queried with QueryStatus.

Count

Property: `Count` as long

Description:
Number of Command objects on this level of the collection.

Item

Property: `Item` (`n` as long) as Command

Description:
Gets the command with the index `n` in this collection. Index is 1-based.

4.5.3 AuthenticDesktopControl

Properties:
IntegrationLevel
Appearance
Application
BorderStyle
CommandsList
CommandsStructure (deprecated)
EnableUserPrompts
MainMenu
Toolbars

Methods:
Open
Exec
QueryStatus

Events:
OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow
OnFileChangedAlert
OnContextChanged
OnDocumentOpened
OnValidationWindowUpdated

This object is a complete ActiveX control and should only be visible if the Authentic Desktop library is used in the Application Level mode.

Properties

The following properties are defined:

IntegrationLevel
EnableUserPrompts
Appearance
BorderStyle

Command related properties:
CommandsList
MainMenu
Toolbars
CommandsStructure (deprecated)

Access to AuthenticDesktopAPI:
Application

Appearance

Property: `Appearance` as short

Dispatch Id: *-520*

Description:
A value not equal to `0` displays a client edge around the control. Default value is `0`.

Application

Property: `Application` as Application

Dispatch Id: *1*

Description:
The `Application` property gives access to the `Application` object of the complete Authentic Desktop automation server API. The property is read-only.

BorderStyle

Property: `BorderStyle` as short

Dispatch Id: *-504*

Description:
A value of `1` displays the control with a thin border. Default value is `0`.

CommandsList

Property: `CommandList` as `Commands` (read-only)

Dispatch Id: *1004*

Description:
This property returns a flat list of all commands defined available with AuthenticDesktopControl.

EnableUserPrompts

Property: `EnableUserPrompts` as boolean

Dispatch Id: *1006*

Description:
Setting this property to *false*, disables user prompts in the control. The default value is *true*.

IntegrationLevel

Property: `IntegrationLevel` as `ICActiveXIntegrationLevel`

Dispatch Id: *1000*

Description:
The `IntegrationLevel` property determines the operation mode of the control. See also Integration at the application level and Integration at document level for more information.

Note: It is important to set this property immediately after the creation of the `AuthenticDesktopControl` object.

MainMenu

Property: `MainMenu` as `Command`(read-only)

Dispatch Id: *1003*

Description:
This property gives access to the description of the AuthenticDesktopControl main menu.

Toolbars

Property: `Toolbars` as `Commands`(read-only)

Dispatch Id: *1005*

Description:
This property returns a list of all toolbar descriptions that describe all toolbars available with AuthenticDesktopControl.

Methods

The following methods are defined:

`Open`
`Exec`
`QueryStatus`

Exec

Method: `Exec` (`nCmdID` as long) as boolean

Dispatch Id: *6*

Description:
`Exec` calls the `Authentic Desktop` command with the ID `nCmdID`. If the command can be executed, the method returns `true`. See also `CommandsStructure` to get a list of all available commands and `QueryStatus` to retrieve the status of any command.

Open

Method: `Open` (`strFilePath` as string) as boolean

Dispatch Id: *5*

Description:
The result of the method depends on the extension passed in the argument `strFilePath`. If the file extension is `.sps`, a new document is opened. If the file extension is `.svp`, the corresponding project is opened. If a different file extension is passed into the method, the control tries to load the file as a new component into the active document.

Do not use this method to load documents or projects when using the control in document-level

integration mode. Instead, use AuthenticDesktopControlDocument.Open and AuthenticDesktopControlPlaceHolder.OpenProject.

QueryStatus

Method: QueryStatus (nCmdID as long) as long

Dispatch Id: *7*

Description:
QueryStatus returns the enabled/disabled and checked/unchecked status of the command specified by nCmdID. The status is returned as a bit mask.

Bit	Value	Name	Meaning
0	1	Supported	Set if the command is supported.
1	2	Enabled	Set if the command is enabled (can be executed).
2	4	Checked	Set if the command is checked.

This means that if QueryStatus returns 0 the command ID is not recognized as a valid Authentic Desktop command. If QueryStatus returns a value of 1 or 5, the command is disabled.

Events

The AuthenticDesktopControl ActiveX control provides the following connection point events:

OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow
OnFileChangedAlert
OnContextChanged

OnDocumentOpened
OnValidationWindowUpdated

OnCloseEditingWindow

Event: OnCloseEditingWindow (i_strFilePath as String) as boolean

Dispatch Id: *1002*

Description:
This event is triggered when Authentic Desktop needs to close an already open document. As an answer to this event, clients should close the editor window associated with *i_strFilePath*. Returning *true* from this event indicates that the client has closed the document. Clients can return *false* if no specific handling is required and AuthenticDesktopControl should try to close the editor and destroy the associated document control.

OnContextChanged

Event: OnContextChanged (i_strContextName as String, i_bActive as bool) as bool

Dispatch Id: *1004*

Description:
This event is not used in Authentic Desktop

OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` as Document)

Dispatch Id: *1*

Description:
This event is triggered whenever a document is opened. The argument `objDocument` is a `Document` object from the Authentic Desktop automation interface and can be used to query for more details about the document, or perform additional operations. When integrating on document-level, it is often better to use the event `AuthenticDesktopControlDocument. OnDocumentOpened` instead.

OnFileChangedAlert

Event: `OnFileChangedAlert` (`i_strFilePath` as String) as bool

Dispatch Id: *1001*

Description:
This event is triggered when a file loaded with AuthenticDesktopControl, is changed on the harddisk by another application. Clients should return true, if they handled the event, or false, if Authentic Desktop should handle it in its customary way, i.e. prompting the user for reload.

OnLicenseProblem

Event: `OnLicenseProblem` (`i_strLicenseProblemText` as String)

Dispatch Id: *1005*

Description:
This event is triggered when AuthenticDesktopControl detects that no valid license is available for this control. In case of restricted user licenses this can happen some time after the control has been initialized. Integrators should use this event to disable access to this control's functionality. After returning from this event, the control will block access to its functionality (e.g. show empty windows in its controls and return errors on requests).

OnOpenedOrFocused

Event: `OnOpenedOrFocused` (`i_strFilePath` as String, `i_bOpenWithThisControl` as bool)

Dispatch Id: *1000*

Description:
When integrating at application level, this event informs clients that a document has been opened, or made active by Authentic Desktop.

When integrating at document level, this event instructs the client to open the file `i_strFilePath` in a document window. If the file is already open, the corresponding document window should be made the active window.

if `i_bOpenWithThisControl` is `true`, the document must be opened with AuthenticDesktopControl, since internal access is required. Otherwise, the file can be opened with different editors.

OnToolWindowUpdated

Event: `OnToolWindowUpdated( pToolWnd` as long `)`

Dispatch Id: *1006*

Description:
This event is triggered when the tool window is updated.

OnUpdateCmdUI

Event: `OnUpdateCmdUI ()`

Dispatch Id: *1003*

Description:
Called frequently to give integrators a good opportunity to check status of Authentic Desktop commands using `AuthenticDesktopControl.QueryStatus`. Do not perform long operations in this callback.

OnValidationWindowUpdated

Event: `OnValidationWindowUpdated` ()

Dispatch Id: *3*

Description:
This event is triggered whenever the validation output window, is updated with new information.

4.5.4 AuthenticDesktopControlDocument

Properties:
Appearance
BorderStyle
Document
IsModified
Path
ReadOnly

Methods:
Exec
New
Open
QueryStatus
Reload
Save
SaveAs

Events:
OnDocumentOpened
OnDocumentClosed
OnModifiedFlagChanged
OnContextChanged
OnFileChangedAlert
OnActivate

If the AuthenticDesktopControl is integrated in the Document Level mode each document is displayed in an own object of type `AuthenticDesktopControlDocument`. The `AuthenticDesktopControlDocument` contains only one document at the time but can be reused to display different files one after another.

This object is a complete ActiveX control.

Properties

The following properties are defined:

ReadOnly
IsModified
Path
Appearance
BorderStyle

Access to AuthenticDesktopAPI:
Document

Appearance

Property: `Appearance` as short

Dispatch Id: *-520*

Description:
A value not equal to `0` displays a client edge around the document control. Default value is `0`.

BorderStyle

Property: `BorderStyle` as short

Dispatch Id: *-504*

Description:
A value of `1` displays the control with a thin border. Default value is `0`.

Document

Property: `Document` as Document

Dispatch Id: *1*

Description:
The `Document` property gives access to the `Document` object of the Authentic Desktop automation server API. This interface provides additional functionalities which can be used with the document loaded in the control. The property is read-only.

IsModified

Property: `IsModified` as boolean (read-only)

Dispatch Id: *1006*

Description:
`IsModified` is *true* if the document content has changed since the last open, reload or save operation. It is *false*, otherwise.

Path

Property: `Path` as string

Dispatch Id: *1005*

Description:
Sets or gets the full path name of the document loaded into the control.

ReadOnly

Property: `ReadOnly` as boolean

Dispatch Id: *1007*

Description:
Using this property you can turn on and off the read-only mode of the document. If `ReadOnly` is `true` it is not possible to do any modifications.

Methods

The following methods are defined:

Document handling:
New
Open
Reload
Save
SaveAs

Command Handling:
Exec
QueryStatus

Exec

Method: `Exec` (`nCmdID` as long) as boolean

Dispatch Id: *8*

Description:
`Exec` calls the Authentic Desktop command with the ID `nCmdID`. If the command can be executed, the method returns `true`. The client should call the `Exec` method of the document control if there is currently an active document available in the application.

See also `CommandsStructure` to get a list of all available commands and `QueryStatus` to retrieve the status of any command.

New

Method: `New` () as boolean

Dispatch Id: *1000*

Description:
This method initializes a new document inside the control..

Open

Method: `Open` (`strFileName` as string) as boolean

Dispatch Id: *1001*

Description:
`Open` loads the file `strFileName` as the new document into the control.

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.

Reload

Method: `Reload` () as boolean

Dispatch Id: *1002*

Description:
`Reload` updates the document content from the file system.

Save

Method: `Save` () as boolean

Dispatch Id: *1003*

Description:
`Save` saves the current document at the location `Path`.

SaveAs

Method: `SaveAs` (`strFileName` as string) as boolean

Dispatch Id: *1004*

Description:
`SaveAs` sets `Path` to *strFileName* and then saves the document to this location.

Events

The AuthenticDesktopControlDocument ActiveX control provides following connection point events:

`OnDocumentOpened`
`OnDocumentClosed`
`OnModifiedFlagChanged`

OnContextChanged
OnFileChangedAlert
OnActivate
OnSetEditorTitle

OnActivate

Event: `OnActivate` ()

Dispatch Id: *1005*

Description:
This event is triggered when the document control is activated, has the focus, and is ready for user input.

OnContextChanged

Event: `OnContextChanged` (`i_strContextName` as String, `i_bActive` as bool) as bool

Dispatch Id: *1004*

Description: None

OnDocumentClosed

Event: `OnDocumentClosed` (`objDocument` as Document)

Dispatch Id: *1001*

Description:
This event is triggered whenever the document loaded into this control is closed. The argument `objDocument` is a `Document` object from the Authentic Desktop automation interface and should be used with care.

OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` as Document)

Dispatch Id: *1000*

Description:
This event is triggered whenever a document is opened in this control. The argument `objDocument` is a `Document` object from the Authentic Desktop automation interface, and can be used to query for more details about the document, or perform additional operations.

OnDocumentSaveAs

Event: `OnContextDocumentSaveAs` (`i_strFileName` as String)

Dispatch Id: *1007*

Description:
This event is triggered when this document gets internally saved under a new name.

OnFileChangedAlert

Event: `OnFileChangedAlert` () as bool

Dispatch Id: *1003*

Description:
This event is triggered when the file loaded into this document control, is changed on the harddisk by another application. Clients should return true, if they handled the event, or false, if Authentic Desktop should handle it in its customary way, i.e. prompting the user for reload.

OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`i_bIsModified` as boolean)

Dispatch Id: *1002*

Description:
This event gets triggered whenever the document changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the document contents differs from the original content, and *false*, otherwise.

OnSetEditorTitle

Event: OnSetEditorTitle ()

Dispatch Id: *1006*

Description:
This event is being raised when the contained document is being internally renamed.

4.5.5 AuthenticDesktopControlPlaceHolder

Properties available for all kinds of placeholder windows:
PlaceholderWindowID

Properties for project placeholder window:
Project

Methods for project placeholder window:
OpenProject
CloseProject

The AuthenticDesktopControlPlaceHolder control is used to show the additional Authentic Desktop windows like Overview, Library or Project window. It is used like any other ActiveX control and can be placed anywhere in the client application.

Properties

The following properties are defined:

PlaceholderWindowID

Access to AuthenticDesktopAPI:
Project

Label

Property: Label as String (read-only)

Dispatch Id: *1001*

Description:
This property gives access to the title of the placeholder. The property is read-only.

PlaceholderWindowID

Property: PlaceholderWindowID as

Dispatch Id: *1*

Description:
Using this property the object knows 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 enumeration. The control changes its state immediately and shows the new Authentic Desktop window.

Project

Property: Project as Project (read-only)

Dispatch Id: *2*

Description:

The Project property gives access to the Project object of the Authentic Desktop automation server API. This interface provides additional functionalities which can be used with the project loaded into the control. The property will return a valid project interface only if the placeholder window has PlaceholderWindowID with a value of Authentic DesktopXProjectWindow (=3). The property is read-only.

Methods

The following method is defined:

OpenProject
CloseProject

OpenProject

Method: OpenProject (strFileName as string) as boolean

Dispatch Id: *3*

Description:
OpenProject loads the file strFileName as the new project into the control. The method will fail if the placeholder window has a PlaceholderWindowID different to XMLSpyXProjectWindow (=3).

CloseProject

Method: CloseProject ()

Dispatch Id: *4*

Description:
CloseProject closes the project loaded the control. The method will fail if the placeholder window has a PlaceholderWindowID different to Authentic DesktopXProjectWindow (=3).

Events

The AuthenticDesktopControlPlaceholder ActiveX control provides following connection point events:

OnModifiedFlagChanged

OnModifiedFlagChanged

Event: OnModifiedFlagChanged (i_bIsModified as boolean)

Dispatch Id: *1*

Description:
This event gets triggered only for placeholder controls with a PlaceholderWindowID of Authentic DesktopXProjectWindow (=3). The event is fired whenever the project

content changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the project contents differs from the original content, and *false*, otherwise.

OnSetLabel

Event: `OnSetLabel( i_strNewLabel` as string)

Dispatch Id: *1000*

Description:
Raised when the title of the placeholder window is changed.

4.5.6 Enumerations

The following enumerations are defined:

ICActiveXIntegrationLevel
AuthenticDesktopControlPlaceholderWindow

ICActiveXIntegrationLevel

Possible values for the IntegrationLevel property of the AuthenticDesktopControl.

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

AuthenticDesktopControlPlaceholderWindow

This enumeration contains the list of the supported additional Authentic Desktop windows.

```
AuthenticDesktopControlNoToolWnd                    =  -1
AuthenticDesktopControlEntryHelperTopToolWnd        =   0
AuthenticDesktopControlEntryHelperMiddleToolWnd     =   1
AuthenticDesktopControlEntryHelperBottomToolWnd     =   2
AuthenticDesktopControlValidatorOutputToolWnd       =   3
AuthenticDesktopControlProjectWindowToolWnd         =   4
AuthenticDesktopControlXSLTDebuggerContextToolWnd   =   5
AuthenticDesktopControlXSLTDebuggerCallstackToolWnd =   6
AuthenticDesktopControlXSLTDebuggerVariableToolWnd  =   7
AuthenticDesktopControlXSLTDebuggerWatchToolWnd     =   8
AuthenticDesktopControlXSLTDebuggerTemplateToolWnd  =   9
AuthenticDesktopControlXSLTDebuggerInfoToolWnd      =  10
AuthenticDesktopControlXSLTDebuggerMessageToolWnd   =  11
AuthenticDesktopControlXSLTDebuggerTraceToolWnd     =  12
AuthenticDesktopControlSOAPDebuggerToolWnd          =  13
AuthenticDesktopControlXPathProfilerListToolWnd     =  14
AuthenticDesktopControlXPathProfilerTreeToolWnd     =  15
AuthenticDesktopControlXPathDialogToolWnd           =  16
AuthenticDesktopControlDBQueryManagerToolWnd        =  17
AuthenticDesktopControlInfoToolWnd                  =  18
AuthenticDesktopControlXSLOutlineToolWnd            =  19
AuthenticDesktopControlSchemaFindToolWnd            =  20
```

Chapter 4

Appendices

Appendices

These appendices contain technical information about Authentic Desktop and important licensing information. Each appendix contains sub-sections as given below:

Technical Data

- OS and memory requirements
- Altova XML Parser
- Altova XSLT and XQuery Engines
- Unicode support
- Internet usage

License Information

- Electronic software distribution
- Intellectual property rights and copyright
- End User License Agreement

1 Technical Data

This section contains useful background information on the technical aspects of your software. It is organized into the following sections:

1.1 OS and Memory Requirements

Operating System
Altova software applications are:

- 32-bit Windows applications for Windows XP, Windows Server 2003 and 2008, Windows Vista, and Windows 7, or
- 64-bit Windows applications for Windows Vista and Windows 7

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. The memory requirement increases with the size of the document.

Memory requirements are also influenced by the unlimited Undo history. When repeatedly cutting and pasting large selections in large documents, available memory can rapidly be depleted.

1.2 Altova XML Parser

When opening any XML document, the application uses its built-in validating parser (the Altova XML Parser) to check for well-formedness, validate the document against a schema (if specified), and build trees and Infosets. The Altova XML Parser 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 Altova XML Parser implements the Final Recommendation of the W3C's XML Schema specification. New developments recommended by the W3C's XML Schema Working Group are continuously being incorporated in the Altova Parser, so that Altova products give you a state-of-the-art development environment.

1.3 Altova XSLT and XQuery Engines

Altova products use the Altova XSLT 1.0 Engine, Altova XSLT 2.0 Engine, and Altova XQuery 1.0 Engines. Documentation about implementation-specific behavior for each engine is in the section Engine Information, in Appendix 1 of the product documentation, should that engine be used in the product.

These three engines are also available in the AltovaXML package, which can be downloaded from the Altova website free of charge. Documentation for using the engines is available with the AltovaXML package.

1.4 Unicode Support

Unicode is the 16-bit character-set (extendable to 32-bit) defined by the Unicode Consortium. It provides a unique number for every character,

- no matter what the platform,
- no matter what the program,
- no matter what the language.

Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. No single encoding could contain enough characters: for example, the European Union alone requires several different encodings to cover all its languages. Even for a single language like English, no single encoding was adequate for all the letters, punctuation, and technical symbols in common use.

These encoding systems used to conflict with one another. That is, two encodings used the same number for two different characters, or different numbers for the same character. Any given computer (especially servers) needs to support many different encodings; yet whenever data is passed between different encodings or platforms, that data always runs the risk of corruption.

Unicode is changing all that!
Unicode provides a unique number for every character, no matter what the platform, no matter what the program, and no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystems, Microsoft, Oracle, SAP, Sun, Base and many others.

Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, etc., and is the official way to implement ISO/IEC 10646. It is supported in many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends.

Incorporating Unicode into client-server or multi-tiered applications and web sites offers significant cost savings over the use of legacy character sets. Unicode enables a single software product or a single web site to be targeted across multiple platforms, languages and countries without re-engineering. It allows data to be transported through many different systems without corruption.

1.4.1 Windows XP

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. Consequently you may encounter XML documents that contain "unprintable" characters, because the font you have selected does not contain the required glyphs. Therefore it can sometimes be very useful to have a font that covers the entire Unicode range - especially when editing XML documents from all over the world.

The most universal font we have encountered is a typeface called Arial Unicode MS that has been created by Agfa Monotype for Microsoft. This font contains over 50,000 glyphs and covers the entire set of characters specified by the Unicode 2.1 standard. It needs 23MB and is included with Microsoft Office 2000.

We highly recommend that you install this font on your system and use it with the application if you are often editing documents in different writing systems. This font is not installed with the "Typical" setting of the Microsoft Office setup program, but you can choose the Custom Setup option to install this font.

In the `/Examples` folder in your application folder you will also find a new XHTML file called `Unicode-UTF8.html` that contains the sentence "When the world wants to talk, it speaks Unicode" in many different languages ("Wenn die Welt miteinander spricht, spricht sie Unicode") and writing-systems (世界的に話すなら、Unicode です。) - this line has been adopted from the 10th Unicode conference in 1997 and is a beautiful illustration of the importance of Unicode for the XML standard. Opening this file will give you a quick impression on what is possible with Unicode and what writing systems are supported by the fonts available on your PC installation.

1.4.2 Right-to-Left Writing Systems

Please note that even under Windows NT 4.0 any text from a right-to-left writing-system (such as Hebrew or Arabic) is not rendered correctly except in those countries that actually use right-to-left writing-systems. This is due to the fact that only the Hebrew and Arabic versions of Windows NT contains support for rendering and editing right-to-left text on the operating system layer.

1.5 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.
- If you use the URL mode of the Open dialog box to open a document directly from a URL (**File | Open | Switch to URL**), that document is retrieved through a http (port 80) connection. (*This functionality is available in XMLSpy and Authentic Desktop.*)
- If you open an XML document that refers to an XML Schema or DTD and the document is specified through a URL, it is also retrieved through a http (port 80) connection once you validate the XML document. This may also happen automatically upon opening a document if you have instructed the application to automatically validate files upon opening in the File tab of the Options dialog (**Tools | Options**). (*This functionality is available in XMLSpy and Authentic Desktop.*)
- 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 this manual and the Altova Software License Agreement.

2 License Information

This section contains:

- Information about the distribution of this software product
- Information about the intellectual property rights related to this software product
- The End User License Agreement governing the use of this software product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

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 before making a purchasing decision.
- 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 a comprehensive integrated onscreen help system. The latest version of the user manual is available at http://www.altova.com/support_help.html (i) in HTML format for online browsing, and (ii) in 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 this 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 have to purchase an End User License Agreement, which is delivered in the form of a key-code that you enter into the Registration dialog to unlock the product. You can register and purchase your license at the online shop on the Altova website.

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 only distribute the Setup programs, provided that they are not modified in any way. Any person that 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.

For further details, please refer to the End User License Agreement at the end of this section.

2.2 Intellectual Property Rights

The Altova Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

Altova software contains certain Third Party Software that is also protected by intellectual property laws, including without limitation applicable copyright laws as described in detail at http://www.altova.com/legal_3rdparty.html.

All other names or trademarks are the property of their respective owners.

2.3 End User License Agreement

Altova End-User License Agreement for Authentic
THIS IS A LEGAL DOCUMENT -- RETAIN FOR YOUR RECORDS
ALTOVA® END-USER LICENSE AGREEMENT
For Authentic® Enterprise Software Editions
And Authentic® Community Software Editions

Licensor:
Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Austria

Important - Read Carefully. Notice to User:

This Altova End User License Agreement for Authentic® ("AEULA") governs your right to (i) use the Authentic Desktop Enterprise Edition software, (ii) use, reproduce and distribute the Authentic Browser-Plugin Enterprise Edition software, (iii) use the Authentic Desktop Community Edition software and (iv) use, reproduce and distribute the Authentic Browser-Plugin Community Edition software (each or collectively hereinafter referenced, as "Authentic Software"). Your license rights depend on the specific software edition that you have licensed as some editions have different rights and restrictions applicable to them as set forth in detail below. This Authentic EULA is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software and any accompanying documentation, including, without limitation, printed materials, 'online' files, or electronic documentation ("Documentation"). By clicking the "I accept" and "Next" buttons below, or by installing, or otherwise using the Authentic Software, you agree to be bound by the terms of this AEULA as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use the Authentic Software, and you must destroy any downloaded copies of the Authentic Software in your possession or control. Please go to our Web site at http://www.altova.com/authenticeula to download and print a copy of this Authentic EULA for your files and http://www.altova.com/privacy to review the Privacy Policy.

1. Authentic Desktop Enterprise Edition ("ADEE") Software Terms and Conditions

(a) License Grant. Upon your acceptance of this AEULA, Altova grants you a non-exclusive, non-transferable (except as provided below), limited license, without the right to grant sublicenses, to install and use a copy of ADEE software on one compatible personal computer or workstation up to the Permitted Number of computers. You may not distribute or reproduce the ADEE software other than as expressly permitted. Subject to the limitations set forth in Section 1(a)(i) you may install and use a copy of this software on more than one of your compatible personal computers or workstations if you have purchased a Named-User license. Subject to the limitations set forth in Sections 1(a)(iii) and 1(a)(iv), users may use the software concurrently on a network. The Permitted Number of computers and/or users and the type of license, e.g. Installed, Named-User, and Concurrent-User, shall be determined and specified at such time as you elect to purchase the software. Installed user licenses are intended to be fixed and not concurrent. In other words, you cannot uninstall for one user in order to reinstall that license to a different user and then uninstall and reinstall back to the original user. Users should be static. Notwithstanding the foregoing, permanent switchovers are acceptable (i.e., an employee has left the company, machine is retired). During the evaluation period, hereinafter defined, only a single user may install and use the ADEE software on one (1) personal

computer or workstation. You may install one (1) copy of the ADEE software on a computer file server within your internal network solely for the purpose of downloading and installing this ADEE software onto other computers within your internal network up to the Permitted Number of computers in a commercial environment only. No other network use is permitted, including without limitation using the ADEE software either directly or through commands, data or instructions from or to a computer not part of your internal network, for Internet or Web-hosting services or by any user not licensed to use this copy of ADEE software through a valid license from Altova, except as set forth in Sections 1(a)(iii) and (iv) below Altova makes no warranties or representations about the performance of Altova software in a terminal server environment and the foregoing are expressly excluded from the limited warranty in Section 3(c) hereof and technical support is not available with respect to issues arising from use in such an environment.

(i) If you have licensed the "Named-User" version of the ADEE software, you may install the software on up to five (5) compatible personal computers or workstations of which you are the primary user thereby allowing you to switch from one computer to the other as necessary provided that only one (1) instance of the ADEE software will be used by you as the Named-User at any given time. If you have purchased multiple Named-User licenses, each individual Named-User will receive a separate license key code.

(ii) If you have licensed a "Concurrent-User" version of the ADEE software, you may install the ADEE software on any compatible computers in a commercial environment only, up to ten (10) times the Permitted Number of users, provided that only the Permitted Number of users actually use the software at the same time and further provided that the computers on which the ADEE software is installed are on the same physical computer network. The Permitted Number of concurrent users shall be delineated at such time as you elect to purchase the ADEE licenses. Each separate physical network or office location requires its own set of separate Concurrent User Licenses for those wishing to use the Concurrent-User versions of the software in more than one location or on more than one network, all subject to the above Permitted Number limitations and based on the number of users using or needing access to the software. If a computer is not on the same physical network, then a locally installed user license or a license dedicated to concurrent use in a virtual environment is required.

(iii) If you have licensed a "Concurrent-User" versions of ADEE software, you may install a copy of the ADEE Software on a terminal server (Microsoft Terminal Server, Citrix Metaframe, etc.), application virtualization server (Microsoft App-V, Citrix XenApp, VMWare ThinApp, etc.) or virtual machine environment within your internal network for the sole and exclusive purpose of permitting individual users within your organization to access and use the Software through a terminal server, application virtualization session, or virtual machine environment from another computer provided that the total number of users that access or use the ADEE Software concurrently at any given point in time on such network, virtual machine or terminal server does not exceed the Permitted Number; and provided that the total number of users authorized to use the ADEE Software through the terminal server, application virtualization session, or virtual machine environment does not exceed six (6) times the Permitted Number of users. Accordingly, the limitations set forth in Section 1(a)(ii) regarding the number of installations and the requirement that the usage be on the same physical network shall not apply to terminal server, application virtualization session, or virtual machine environments. In a virtual environment, you must deploy a means of preventing users from exceeding the Permitted Number of concurrent users. Altova makes no warranties or representations about the performance of Altova software in a terminal server, application virtualization session, or virtual machine environment and the foregoing are expressly excluded from the limited warranty in Section 3 hereof and technical support is not available with respect to issues arising from use in such environments.

(iv) You may make one (1) backup and one (1) archival copy of the ADEE software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the ADEE software. You may not

transfer the rights to a backup or archival copy unless you transfer all rights in the ADEE software as provided in this AEULA. You, as the primary user of the computer on which the ADEE software is installed, may also install the ADEE software on one of your home computers for your use. A copy of the ADEE software may be installed on home computers up to a total of the number of Permitted Users provided that the ADEE software will not be used at the same time on a home computer as the ADEE software is being used on the primary computer. If you are using a Concurrent-User version of the ADEE software for home use, then you may install the software on any compatible computers equal to the number of Permitted Users only.

(b) Key Codes. Prior to your purchase and as part of the registration for the thirty (30) -day evaluation period, as applicable, you will receive an evaluation key code. You will receive a purchase key code when you elect to purchase the ADEE software licenses from either Altova GmbH or an authorized reseller. The purchase key code will enable you to activate the software beyond the initial evaluation period. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova.

(c) Limited Transfer Rights. You may transfer all your rights to use the ADEE software to another person or legal entity provided that: (i) you also transfer each of this AEULA, the ADEE software and all other software or hardware bundled or pre-installed with the ADEE software, including all copies, updates and prior versions, and all copies of font software converted into other formats, to such person or entity; (ii) you retain no copies, including backups and copies stored on a computer; (iii) the receiving party secures a personalized key code from Altova; and (iv) the receiving party accepts the terms and conditions of this AEULA and any other terms and conditions upon which you legally purchased a license to the ADEE software.

(d) Applicable AEULA Terms. The terms and conditions set forth in Sections 1, 3 and 7 apply to the ADEE software.

2. Authentic_Browser-Plugin_Enterprise_Edition_("ABEE")_Software_Terms_and Conditions

(a) License Grant and Term. Upon your acceptance of this AEULA, Altova grants you a non-exclusive, non-transferable limited license, without the right to grant sublicenses, to install and use ABEE software on a per server basis for a twelve (12) month term, commencing on the date of your license purchase and expiring on the date that is twelve (12) months thereafter (the "ABEE License Term"). Altova also grants you a non-exclusive, non-transferable, limited worldwide license, without the right to grant sublicenses, to use software to develop web pages, web applications, or applications that include ABEE software, to reproduce the ABEE software on your website or server and to distribute the ABEE software from your website or server over a computer network, but only in its executable object code form, and only to end users for the limited purpose of enabling them to view, share, and/or edit XML files during the ABEE License Term. If you wish to continue to use, and/or reproduce and/or distribute the ABEE software after the expiration of its license term, you must purchase a new Authentic Browser-Plugin Enterprise Edition. If you have purchased an ABEE software license then under the terms of the AEULA, support and maintenance (or SMP as further detailed below) for the software is included as part of the license purchase and you will be entitled to receive the benefits set forth below during the ABEE License Term which is coterminous with the Support Period. Unlike other Altova software products, you **cannot** renew SMP for the ABEE software and at the expiration of the ABEE License Term and Support Period, you must purchase a new ABEE software license if you wish to continue to use, reproduce or distribute the ABEE software.

(b) Key Codes. Prior to your purchase you may request a thirty (30) day evaluation key code, which will be sent to you. **You will receive a purchase key code when you elect to purchase the ABEE software licenses from either Altova GmbH or an authorized reseller. The purchase key code will enable you to use the ABEE software during the ABEE License Term. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova.**

(c) ABEE Software Specific Restrictions. In addition to the restrictions and obligations provided in other sections of this AEULA that are applicable to the ABEE software, your limited license to distribute the ABEE software set forth above, is further subject to all of the following restrictions; (i) ABEE software may only be licensed but may not be sold, and (ii) you must use, reproduce or distribute the ABEE software provided by Altova **AS IS** and may not impair, alter or remove Altova's AEULA, (which will appear in the installation process and which an end user must accept in order to be able to install or operate the software or any other files).

(d) Applicable AEULA Terms. The terms and conditions set forth in Sections 2, 3 and 7 apply to the ABEE software.

3. Authentic_Enterprise_Editions_(ADEE_and_ABEE)_Software_Additional_Terms_and Conditions

The terms set forth in Section 3 are applicable to the ADEE and ABEE software licenses and are in addition to the specific terms applicable to those software licenses.

(a) Upgrades and Updates. If the software that you have licensed is an upgrade or an update, then the latest update or upgrade that you download and install replaces all or part of the ADEE or ABEE software previously licensed The update or upgrade and the associated license keys, as applicable, does not constitute the granting of a second license to the software in that you may not use the upgrade or updated copy in addition to the copy of the software that it is replacing and whose license has terminated.

(b) Support and Maintenance. Altova offers "Support & Maintenance Package(s)" ("SMP") for the ADEE and ABEE software product editions that you have licensed. The Support Period, hereinafter defined, covered by such SMP shall be delineated at such time as you elect to purchase a SMP. In the case of your ABEE software license, twelve months of SMP is included that is coterminous with the ABEE License Term. Your rights with respect to support and maintenance as well as your upgrade eligibility depend on your decision to purchase SMP and the level of SMP that you have purchased:

(i) If you have not purchased SMP, you will receive the software AS IS and will not receive any maintenance releases or updates. However; Altova, at its option and in its sole discretion on a case by case basis, may decide to offer maintenance releases to you as a courtesy, but these maintenance releases will not include any new features in excess of the feature set at the time of your purchase of the Software. In addition, Altova will provide free technical support to you for thirty (30) days after the date of your purchase (the "Support Period" for the purposes of this paragraph 3(b)), and Altova, in its sole discretion on a case by case basis, may also provide free courtesy technical support during your thirty (30) -day evaluation period. Technical support is provided via a Web-based support form only, and there is no guaranteed response time.

(ii) If you have purchased SMP then, solely for the duration of its delineated Support Period, **you are eligible to receive the version of the ADEE or ABEE software edition** that you have licensed and all maintenance releases and updates for that edition that are released during your Support Period. For the duration of your SMP's Support Period, you will also be eligible to receive upgrades to the comparable edition of the next version of the ADEE or ABEE software that succeeds the software edition that you have licensed for applicable upgrades released during your Support Period. The specific upgrade edition that you are eligible to receive based on your Support Period is further detailed in the SMP that you have purchased. Software that is introduced as a separate product is not included in SMP. Maintenance releases, updates and upgrades may or may not include additional features. In addition, Altova will provide Priority Technical Support to you for the duration of the Support Period. Priority Technical Support is provided via a Web-based support form only and Altova will make commercially reasonable efforts to respond via e-mail to all requests within forty-eight

(48) hours during Altova's business hours (MO-FR, 8am UTC – 10pm UTC, Austrian and US holidays excluded) and to make reasonable efforts to provide work-arounds to errors reported in the software.

(iii) During the Support Period you may also report any software problem or error to Altova. If Altova determines that a reported reproducible material error in the software exists and significantly impairs the usability and utility of the ADEE or ABEE software, Altova agrees to use reasonable commercial efforts to correct or provide a usable work-around solution in an upcoming maintenance release or update, which is made available at certain times at Altova's sole discretion. If Altova, in its discretion, requests written verification of an error or malfunction discovered by you or requests supporting example files that exhibit the software problem, you shall promptly provide such verification or files, by email, telecopy, or overnight mail, setting forth in reasonable detail the respects in which the ADEE or ABEE software fails to perform. You shall use reasonable efforts to cooperate in diagnosis or study of errors. Altova may include error corrections in maintenance releases, updates, or new major releases of the software. Altova is not obligated to fix errors that are immaterial. Immaterial errors are those that do not significantly impact use of the software as determined by Altova in its sole discretion. Whether or not you have purchased the Support & Maintenance Package, technical support only covers issues or questions resulting directly out of the operation of the ADEE or ABEE software and Altova will not provide you with generic consultation, assistance, or advice under any circumstances.

(iv) Updating the ADEE or ABEE software may require the updating of software not covered by this AEULA before installation. Updates of the operating system and application software not specifically covered by this AEULA are your responsibility and will not be provided by Altova under this AEULA. Altova's obligations under this Section are contingent upon your proper use of the ADEE or ABEE software and your compliance with the terms and conditions of this AEULA at all times. Altova shall be under no obligation to provide the above technical support if, in Altova's opinion, the ADEE or ABEE software has failed due to the following conditions: (a) damage caused by the relocation of the software to another location or CPU; (b) alterations, modifications or attempts to change the software without Altova's written approval; (c) causes external to the software, such as natural disasters, the failure or fluctuation of electrical power, or computer equipment failure; (d) your failure to maintain the software at Altova's specified release level; or (e) use of the software with other software without Altova's prior written approval. It will be your sole responsibility to: (a) comply with all Altova-specified operating and troubleshooting procedures and then notify Altova immediately of ADEE or ABEE software malfunction and provide Altova with complete information thereof; (b) provide for the security of your confidential information; and (c) establish and maintain backup systems and procedures necessary to reconstruct lost or altered files, data or programs.

(c) Limited Warranty. Altova warrants to the person or entity that first purchases a license for use of the ADEE or ABEE software pursuant to the terms of this AEULA that (i) the software will perform substantially in accordance with any accompanying documentation for a period of ninety (90) days from the date of receipt, and (ii) any support services provided by Altova shall be substantially as described in Section 3(b) of this AEULA. Some states and jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. To the extent allowed by applicable law, implied warranties on the ADEE or ABEE software, if any, are limited to ninety (90) days. Altova's and its suppliers' entire liability and your exclusive remedy shall be, at Altova's option, either (i) return of the price paid, if any, or (ii) repair or replacement of the ADEE or ABEE software that does not meet Altova's Limited Warranty and which is returned to Altova with a copy of your receipt. This Limited Warranty is void if failure of the ADEE or ABEE software has resulted from accident, abuse, misapplication, abnormal use, Trojan horse, virus, or any other malicious external code. Any replacement ADEE or ABEE software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. This limited warranty does not apply to Evaluation Software. THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA'S OR ITS SUPPLIERS' BREACH OF WARRANTY.

ALTOVA AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, ALTOVA AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, INFORMATIONAL CONTENT OR ACCURACY, QUIET ENJOYMENT, TITLE AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

(d) Limitation of Liability and Infringement Claims. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW EVEN IF A REMEDY FAILS ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE ADEE or ABEE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, ALTOVA'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS AEULA SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE ADEE OR ABEE SOFTWARE PRODUCT. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions, Altova's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this AEULA between Altova and you. Altova will indemnify and hold you harmless and will defend or settle any claim, suit or proceeding brought against you by a third party that is based upon a claim that the content contained in the ADEE or ABEE software infringes a copyright or violates an intellectual or proprietary right protected by United States or European Union law ("Claim"), but only to the extent the Claim arises directly out of the use of the Software and subject to the limitations set forth in this Section 3(d) of this Agreement except as otherwise expressly provided. You must notify Altova in writing of any Claim within ten (10) business days after you first receive notice of the Claim, and you shall provide to Altova at no cost such assistance and cooperation as Altova may reasonably request from time to time in connection with the defense of the Claim. Altova shall have sole control over any Claim (including, without limitation, the selection of counsel and the right to settle on your behalf on any terms Altova deems desirable in the sole exercise of its discretion). You may, at your sole cost, retain separate counsel and participate in the defense or settlement negotiations. Altova shall pay actual damages, costs, and attorney fees awarded against you (or payable by you pursuant to a settlement agreement) in connection with a Claim to the extent such direct damages and costs are not reimbursed to you by insurance or a third party, to an aggregate maximum equal to the purchase price of the ADEE or ABEE software. If the ADEE or ABEE software or its use becomes the subject of a Claim or its use is enjoined, or if in the opinion of Altova's legal counsel the software is likely to become the subject of a Claim, Altova shall attempt to resolve the Claim by using commercially reasonable efforts to modify the ADEE or ABEE software or obtain a license to continue using the ADEE or ABEE software. If in the opinion of Altova's legal counsel the Claim, the injunction or potential Claim cannot be resolved through reasonable modification or licensing, Altova, at its own election, may terminate

this AEULA without penalty, and will refund to you on a pro rata basis any fees paid in advance by you to Altova. THE FOREGOING CONSTITUTES ALTOVA'S SOLE AND EXCLUSIVE LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT. This indemnity does not apply to infringements that would not be such, except for customer-supplied elements.

(e) Evaluation Software. This section applies to all evaluation copies of the ADEE or ABEE software ("Evaluation Software") and continues in effect until you purchase a license. THE EVALUATION SOFTWARE IS PROVIDED TO YOU "AS-IS" WITH NO WARRANTIES FOR USE OR PERFORMANCE, AND ALTOVA DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, WHETHER EXPRESS OR IMPLIED. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR PRE-RELEASE AND/OR EVALUATION SOFTWARE, BUT IT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL. If the Evaluation Software has a time-out feature, then the software will cease operation after the conclusion of the designated evaluation period. Access to any files created with the Evaluation Software is entirely at your risk.

4. Authentic Desktop Community Edition ("ADCE") Software Terms and Conditions

(a) License Grant and Keycode. Upon your acceptance of this AEULA, Altova grants you a non-exclusive, non-transferable limited license, without the right to grant sublicenses, to install and use a copy of ADCE software on your compatible personal computer or workstation for the purpose of viewing, distributing, sharing, and editing of XML files solely in connection with STYLEVISION® Power Stylesheets as further provided herein. You will receive a key code that will enable you to activate or operate the ADCE software. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova. You may make one (1) backup and one (1) archival copy of the ADCE software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the ADCE. You may install one (1) copy of such Setup Program for ADCE software on a computer file server within your internal network for the sole and exclusive purpose of installing the ADCE software to an unlimited number of client computers on your internal network. No other server or network use of the ADCE software is permitted, including but not limited to using the ADCE software (i) either directly or through commands, data or instructions from or to another computer or (ii) for internal network, internet or web hosting services.

(b) Distribution. Upon your acceptance of this AEULA as part of your use of the ADCE software, and subject to your ongoing compliance with its terms and conditions, Altova hereby grants ADCE software users a non-exclusive, non-transferable, limited license, without the right to grant sublicenses, to reproduce the Setup Program for the ADCE software and distribute the Setup Program for the ADCE software in executable form to end users in the manner hereinafter provided. You may distribute the Setup Program for the ADCE software to any third party electronically or via download from the website or on physical media such as CD-ROMS or diskettes as part of or in conjunction with products that you have developed.

(c) ADCE Software Specific Restrictions. In addition to the restrictions and obligations provided in other sections of this AEULA, your license to distribute the Setup Program for the ACDE software is further subject to all of the following restrictions: (i) ACDE software shall only be licensed and not sold, (ii) you may not make the ACDE software available as a stand-alone product and if distributed as part of a product bundle you may charge for the product bundle provided that you license such product bundle at the same or lower fee at which you license any reasonably equivalent product bundle which does not include the ACDE software, (iii) you must use the Setup Program for ACDE provided by Altova AS IS and may not impair, alter or remove Altova's AEULA, (which will appear in the installation process and which an end user must accept in order to be able to install or operate the ACDE software) or any other files, and (iv) you may not combine the ACDE software with your product in such a way that your product

modifies or generates Stylevision Power Stylesheets.

(d) Applicable AEULA Terms. The terms and conditions set forth in Sections 4, 6 and 7 apply to the ADCE software.

5. Authentic Browser-Plugin Community Edition ("ABCE") Software Terms and Conditions

(a) License Grant and Distribution. Upon your acceptance of this AEULA, Altova grants you a non-exclusive, non-transferable limited license, without the right to grant sublicenses, to use and develop web pages, web applications, or applications that include the ABCE software, to reproduce the ABCE software and to distribute the ABCE software in executable form in the manner hereinafter provided to end users for the purpose of viewing, sharing and editing XML files solely in connection with StyleVision® Power Stylesheets as further provided herein. You may install the ABCE software on a web server within your network for the purpose of downloading and installing the ABCE software (to an unlimited number of client computers on your internal network). You may distribute the ABCE software to any third party electronically or via download from the website or on physical media such as CD-ROMS or diskettes as part of or in conjunction with products that you have developed.

(b) ABCE Software Specific Restrictions. In addition to the restrictions and obligations provided in other sections of this AEULA, your license to distribute ABCE software is further subject to all of the following restrictions: (i) the ABCE software shall only be licensed and not sold; (ii) you may not make the ABCE software available as a stand-alone product and if distributed as part of a product bundle you may charge for the product bundle provided that you license such product bundle at the same or lower fee at which you license any reasonably equivalent product bundle which does not include the ABCE software; (iii) you must use the ABCE software provided by Altova AS IS and may not impair, alter or remove Altova's AEULA (which will appear in the installation process and which an end user must accept in order to be able to install or operate the ABCE software) or any other files; (iv) other Altova products cannot be distributed under this AEULA; and (v) you may not combine the ABCE software with your product in such a way that your product modifies or generates Stylevision Power Stylesheet(s).

(c) Applicable AEULA Terms. The terms and conditions set forth in Sections 5, 6 and 7 apply to the ABCE software.

6. Authentic Community Editions (ADCE and ABCE) Software Additional Terms and Conditions

The terms set forth in Section 6 are applicable to the ADCE and ABCE software licenses and are in addition to the specific terms applicable to those software licenses.

(a) Use Limitation. The ADCE and ABCE software are licensed and distributed by Altova for viewing, distributing, sharing, and editing of XML files solely in connection with StyleVision® Power Stylesheets, defined as .sps files that are template files developed by Altova or its customers using Altova's StyleVision® product. You are not authorized to integrate or use the ADCE or ABCE software with (i) any StyleVision® Power Stylesheet(s) not developed in accordance with the Altova Software License Agreement available at http://www.altova.com/eula or (ii) other software or enhancement that uses Inter Application Communication (IAC) to programmatically interface with ADCE or ABCE software for the purpose of enabling additional functionality normally not available in ADCE or ABCE software or providing functionality that competes with other Altova products.

(b) Warranty Disclaimer. THE ADCE OR ABCE SOFTWARE IS PROVIDED TO YOU FREE OF CHARGE, AND ON AN "**AS-IS**" BASIS. ALTOVA PROVIDES NO WARRANTIES FOR THE ADCE OR ABCE SOFTWARE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES AND REPRESENTATIONS,

WHETHER EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, SATISFACTORY QUALITY, INFORMATIONAL CONTENT, OR ACCURACY, QUIET ENJOYMENT, TITLE, AND NON- INFRINGEMENT. ALTOVA DOES NOT WARRANT THAT THE ADCE OR ABCE SOFTWARE IS ERROR-FREE OR WILL OPERATE WITHOUT INTERRUPTION. IF MANDATORILY APPLICABLE LAW REQUIRES ANY WARRANTIES WITH RESPECT TO THE ADCE OR ABCE SOFTWARE, ALL SUCH WARRANTIES ARE LIMITED IN DURATION TO THIRTY (30) DAYS FROM THE DATE OF INSTALLATION OR BEGIN OF USE, WHATEVER IS THE EARLIER. SOME STATES OR JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER LEGAL RIGHTS THAT VARY FROM STATE TO STATE OR FROM JURISDICTION TO JURISDICTION. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR THE ACCURACY AND ADEQUACY OF THE ADCE OR ABCE SOFTWARE FOR YOUR INTENDED USE AND YOU WILL INDEMNIFY AND HOLD HARMLESS ALTOVA FROM ANY 3RD PARTY SUIT TO THE EXTENT BASED UPON THE ACCURACY AND ADEQUACY OF THE ADCE OR ABCE SOFTWARE IN YOUR USE.

(c) Limitation of Liability. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE AUTHENTIC SOFTWARE, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, OR ANY PROVISION OF THIS AEULA, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. WHERE LEGALLY, LIABILITY CANNOT BE EXCLUDED, BUT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY, THE ABOVE LIMITATION MAY NOT APPLY TO YOU. IN SUCH STATES AND JURISDICTIONS, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE GREATEST EXTENT PERMITTED BY LAW. THE FOREGOING LIMITATIONS ON LIABILITY ARE INTENDED TO APPLY TO THE WARRANTIES AND DISCLAIMERS ABOVE AND ALL OTHER ASPECTS OF THIS AEULA.

(d) Support. Altova is not obliged to provide technical support with respect to the ADCE or ABCE software, that is provided on an AS IS basis. To the extent that Altova in its sole discretion does provide support for these products, the technical support will be provided in the manner detailed in Section 3(b) of this AEULA and subject to all requirements therein contained.

7. Authentic Software (ADEE, ABEE, ADCE and ABCE) Additional Terms and Conditions

The terms set forth in Section 7 are applicable to the ADEE, ABEE, ADCE and ABCE software licenses and are in addition to the specific terms applicable to those software licenses.

(a) Title. Title to the Authentic Software is not transferred to you. Ownership of all copies of the Authentic Software and of copies made by you is vested in Altova, subject to the rights of use or distribution, as applicable, granted to you in this AEULA. All rights not specifically granted in this AEULA are reserved by Altova.

(b) Acknowledgement of Altova's Rights. You acknowledge that the Authentic Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Authentic Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Authentic Software is protected by copyright, including without limitation by United States

Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Authentic Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Authentic Software and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Authentic Software. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Authentic Software, and such use of any trademark does not give you any right of ownership in that trademark. XMLSpy®, Authentic®, StyleVision®, MapForce®, UModel®, DatabaseSpy®, DiffDog®, SchemaAgent®, SemanticWorks®, MissionKit®, Markup Your Mind®, Nanonull™, and Altova® are trademarks of Altova GmbH (registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows XP, Windows Vista, and Windows 7 are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the W3C are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this AEULA does not grant you any intellectual property rights in the Authentic Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

(c) Common Restrictions.

(i) You may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Authentic Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law in the European Union, if it is essential to do so in order to achieve operability of the Authentic Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Authentic Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department. You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Authentic Software to third parties except to the limited extent expressly provided in this AEULA.

(ii) You may not copy, distribute, or make derivative works of the Authentic Software except as expressly set forth above, and any copies that you are permitted to make pursuant to this Authentic EULA must contain the same copyright, patent and other intellectual property markings that appear on or in the Authentic Software. You may not modify, adapt or translate the Authentic Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Authentic Software; knowingly take any action that would cause the Authentic Software to be placed in the public domain; or use the Authentic Software in any computer environment not specified in this Authentic EULA. You will comply with applicable law and Altova's instructions regarding the use of the Authentic Software. You agree to notify your employees and agents who may have access to the Authentic Software of the restrictions contained in this AEULA and to ensure their compliance with these restrictions. You may not alter or modify the Authentic Software or create a new installer for the Authentic Software.

(d) <u>Authentic Software Activation, Updates, Metering and Data Use.</u>

(i) Altova has a built-in license metering module that helps you to avoid any

unintentional violation of this AEULA. Altova may use your internal network for license metering between installed versions of the Authentic Software. **Altova's Authentic 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 Authentic Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova license server. You agree that Altova may use these measures and you agree to follow any applicable requirements. You further agree that use of license key codes that are not or were not generated by Altova and lawfully obtained from Altova or an authorized reseller as part of an effort to activate or use the Authentic Software violates Altova's intellectual property rights as well as the terms of this AEULA. You agree that efforts to circumvent or disable Altova's copyright protection mechanisms or license management mechanism violate Altova's intellectual property rights as well as the terms of this AEULA. Altova expressly reserves the rights to seek all available legal and equitable remedies to prevent such actions and to recover lost profits, damages and costs.**

(ii) Altova provides a new LiveUpdate notification service to you, which is free of charge. Altova may use your internal network and Internet connection for the purpose of transmitting license-related data to an Altova-operated LiveUpdate server to validate your license at appropriate intervals and determine if there is any update available for you. The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this AEULA. By your acceptance of the terms of this AEULA or use of the Authentic Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this AEULA and/or the Privacy Policy.. Altova has the right in its sole discretion to amend this provision of the AEULA and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

(iii) Notice to European Users. Please note that the information as described in paragraph 7(d) above may be transferred outside of the European Economic Area, for purposes of processing, analysis, and review, by Altova, Inc. a company located in Beverly, Massachusetts, U.S.A., or its subsidiaries or Altova's subsidiaries or divisions, or authorized partners, located worldwide. You are advised that the United States uses a sectoral model of privacy protection that relies on a mix of legislation, governmental regulation, and self-regulation. You are further advised that the Council of the European Union has found that this model does not provide "adequate" privacy protections as contemplated by Article 25 of the European Union's Data Directive. (Directive 95/46/EC, 1995 O.J. (L 281) 31). Article 26 of the European Union's Data Directive allows for transfer of personal data from the European Union to a third country if the individual has unambiguously given his consent to the transfer of personal information, regardless of the third country's level of protection. By agreeing to this AEULA, you consent to the transfer of all such information to the United States and the processing of that information as described in this AEULA and the Privacy Policy.

(e) Disclaimer. THE AUTHENTIC SOFTWARE IS NEITHER GUARANTEED NOR WARRANTED TO BE ERROR-FREE NOR SHALL ANY LIABILITY BE ASSUMED BY ALTOVA IN THIS RESPECT. NOTWITHSTANDING ANY SUPPORT FOR ANY TECHNICAL STANDARD, THE AUTHENTIC SOFTWARE IS NOT INTENDED FOR USE IN OR IN CONNECTION WITH, WITHOUT LIMITATION, THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL EQUIPMENT, MEDICAL DEVICES OR LIFE SUPPORT SYSTEMS, MEDICAL OR HEALTH CARE APPLICATIONS, OR OTHER APPLICATIONS WHERE THE FAILURE OF THE AUTHENTIC SOFTWARE OR ERRORS IN DATA PROCESSING COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR THE ACCURACY AND ADEQUACY OF THE AUTHENTIC SOFTWARE AND ANY DATA GENERATED OR PROCESSED BY THE

SOFTWARE FOR YOUR INTENDED USE AND YOU WILL DEFEND, INDEMNIFY AND HOLD ALTOVA, ITS OFFICERS AND EMPLOYEES HARMLESS FROM ANY 3RD PARTY CLAIMS, DEMANDS, OR SUITS THAT ARE BASED UPON THE ACCURACY AND ADEQUACY OF THE AUTHENTIC SOFTWARE IN YOUR USE OR ANY DATA GENERATED BY THE AUTHENTIC SOFTWARE IN YOUR USE.

(f) Restricted Rights Notice and Export Restrictions. The Authentic Software was developed entirely at private expense and is commercial computer software provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. §12.211 and 12.212) or DFARS 227. 7202 (48 C.F.R. §227-7202) as applicable. Consistent with the above, as applicable, Commercial Computer Software and Commercial Computer Documentation licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this AEULA. Manufacturer is Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Authentic Software or documentation except as authorized by United States law and the laws of the jurisdiction in which the Authentic Software was obtained. In particular, but without limitation, the Authentic Software or Documentation may not be exported or re-exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

(g) Termination. Without prejudice to any other rights or remedies of Altova, this AEULA may be terminated (i) by you giving Altova written notice of termination or (ii) by Altova, at its option, giving you written notice of termination or (iii) Altova giving you written notice of termination if you fail to comply with the terms and conditions of the AEULA. This AEULA automatically terminates upon the expiration of the ABEE License Term. Upon any termination or expiration of this AEULA, you must cease all use of Authentic Software, licensed hereunder, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Authentic Software remain in your possession or control. The terms and conditions set forth in Sections 1(e), 2(c)-(d), 3(c)-(d), 4(c)-(d), 5(b)-(c), 6(b)-(c) and 7 survive termination of this AEULA as applicable.

(h) Third Party Software. The Authentic Software may contain third party software which requires notices and/or additional terms and conditions. Such required third party software notices and/or additional terms and conditions are located at our website at http://www.altova.com/legal_3rdparty.html and are made a part of and incorporated by reference into this AEULA. By accepting this AEULA, you are also accepting the additional terms and conditions, if any, set forth therein.

(i) General Legal Provisions. This AEULA contains the entire agreement and understanding of the parties with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this AEULA shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova's Web site for Altova and the address shown in Altova's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This AEULA will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party's right at a later time to enforce the same or any other term of this AEULA. This AEULA may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this AEULA by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly

authorized and empowered to enter into and perform this AEULA. If, for any reason, any provision of this AEULA is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this AEULA, and this AEULA shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

(i) If you are located in the European Union and are using the Authentic Software in the European Union and not in the United States, then this AEULA will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Authentic Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht, Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

(ii) If you are located in the United States or are using the Authentic Software in the United States then this AEULA will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Authentic Software resides in the federal or state courts of the Commonwealth of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of the Commonwealth of Massachusetts in connection with any such dispute or claim.

(iii) If you are located outside of the European Union or the United States and are not using the Authentic Software in the United States, then this AEULA will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Authentic Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim. This AEULA will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

Last Updated: 2011-10-01

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X