User and Reference Manual

Altova Authentic 2017 Browser Edition User & Reference Manual

Published: 2017

Table of Contents

Chapter 1

Altova Authentic Browser Edition

Altova Authentic Browser Edition

Altova® Authentic® 2017 Browser Edition empowers business users to easily create and edit XML and database content through a user interface that closely resembles an easy-to-use word processor. The Browser Edition can be embedded in any Web page and allows editing directly within the Browser. Authentic Browser Edition is available as a plug-in for the **Microsoft Internet Explorer** and **Mozilla Firefox** browsers.

Authentic Browser is available in a single **Enterprise Edition** (license required), in a trusted version and an untrusted version.

Chapter 2

About This Documentation

About This Documentation

This documentation is the Authentic Browser Plug-in user manual. It is organized into the following sections:

- An Overview section that explains: (i) the benefits of using Authentic Browser; (ii) how Authentic Browser works; and (iii) the various Authentic Browser Versions.
- A Server Setup section, which describes the steps required to set up a server for an Authentic Browser project, including a detailed description of the HTML Page for Authentic Plug-in, and a section explaining how the Authentic Browser extension packages can be used for on-demand installation.
- A Client Setup section, which includes a section explaining the various ways of installing the Authentic Browser plug-in in client browsers.
- A three-part reference section (User Reference: Mechanisms, User Reference: Objects, and User Reference: Enumerations) on the mechanisms, objects, and enumerations used to create and customize Authentic View in Authentic Browser.
- To ease deployment of the Authentic Browser Plug-in we provide an ASP.NET Server Control that completely integrates with Visual Studio .NET.

Related documentation
There are two sets of additional Altova documentation that are relevant:

- Documentation for creating StyleVision Power Stylesheets (SPSs) is available with the Altova StyleVision product at the Altova website. An SPS is the file that controls the Authentic View of an XML document. This documentation is relevant for persons developing the Authentic View interface for XML editing.
- Documentation for using Authentic View. Persons using Authentic View should be referred to the Authentic View tutorial and usage sections of the Authentic Desktop User Manual, which is available online at the Altova website.

Chapter 3

Overview

Overview

Altova® Authentic® 2017 Browser Edition enables users to edit XML documents **based on StyleVision Power Stylesheets (.sps files) that have been created in Altova StyleVision**. It is a unique solution that allows live XML content editing from within an Internet browser window on any desktop (client) in your organization.

This Overview section is organized into the following sub-sections:

- Benefits of Authentic Browser: notes the uses and advantages of using Authentic Browser.
- How It Works: briefly explains how the Authentic Browser solution works; it organizes the description in terms of server and client setup requirements.
- Authentic Browser Versions: explains the various Authentic Browser versions currently available at the Altova website.

1 Benefits of Authentic Browser

The Authentic Browser XML editing solution has several benefits, the most important of which are listed below:

- Enables multiple users to access and edit an XML document via their browsers. Currently, **Microsoft Internet Explorer 5.5 or higher** and **Mozilla Firefox** are supported. The 64-bit versions of Internet Explorer 10 and 11 are not supported.
- Dramatically eases organization-wide deployment and application maintenance while also reducing the total cost of ownership.
- Based upon open standards, such as XML Schema and XSLT.
- Is fully Unicode compatible.
- Uses Altova's Authentic View, which is based on the widely used and easily available Internet Explorer browser. Authentic View enables users to edit XML files in a WYSIWYG fashion, i.e. without users having to see the underlying XML code.
- Does not require deployment of any additional software since the Authentic Browser plug-in is a browser add-on.
- The Authentic Browser is an ActiveX control where the COM interface is defined by the Authentic object. The complete object model is described in the User Reference: Objects section of this documentation.

2 How It Works

To implement an Authentic Browser project, you will need one server machine, and one or more client machines that are connected to the server.

Authentic Browser Server
The Authentic Browser server carries out the following functions:

- The server stores files related to the Altova Authentic XML document that is to be edited. These files are:

  1. The XML Schema (XSD) on which the editable XML document is based;
  2. The XML document to be edited;
  3. The SPS or PXF file that controls the layout and input mechanisms of the XML document in Authentic View.

- The server stores the HTML Page for Authentic Plug-in. This HTML page is the access point for Authentic View editing. It contains instructions to access the XML document, and it serves as a container for the Authentic View window in which the XML document is loaded and edited. To access this page, its URL is typed into client browsers.
- If you are deploying the Enterprise edition of Authentic Browser, then the Enterprise license must be stored on the server. Enterprise licenses are issued for one or more specified servers.
- If on-demand installation of the Authentic Browser plug-in is planned, the server stores the Authentic Browser extension package/s (CAB, XPI, and/or CRX file/s) for the download and installation of the plug-in on client machines. (Otherwise, the plug-in is installed directly in client browsers.)

The steps required to prepare the server are described in the section, Server Setup.

Authentic Browser Clients
An XML document can be viewed and edited in Altova's Authentic View if the XML document has an SPS or PXF file assigned to it. Authentic View editing is carried out on client machines, which must be set up as follows:

- Each client must have one of the following browsers installed: **Internet Explorer 5.5 or higher (32-bit and 64-bit)**, **Firefox (32-bit)**
- Additionally, each client machine **must have Internet Explorer 5.5 or higher** installed on it. This is because the Authentic View interface (which will be displayed within the browser window) is generated using Internet Explorer.
- Unless on-demand installation has been planned, the Authentic Browser plug-in must be directly installed on client browsers.

The steps required to prepare clients are described in the section, Client Setup.

Authentic Browser mechanism
After the server and clients have been prepared as described above, the user enters the URL of the HTML page for Authentic Plug-in in the client browser. If the plug-in has not already been installed in the client browser, the HTML page can contain instructions to perform an on-demand installation of the plug-in in the client browser.

Once the plug-in is installed on the client, code in the HTML page causes an Authentic View

editing window to open within the browser window. The XML document to be edited is loaded into the Authentic View window from the server and the user can start editing it and saving changes directly to the XML document.

3 Authentic Browser Versions

Authentic Browser versions are available according to the following criteria:

- *Language:* English (EN), German (DE), Spanish (ES), Japanese (JA)
- *Trusted/Untrusted:* Trusted, Untrusted
- *Client browser:* Microsoft Internet Explorer (32-bit and 64-bit), Mozilla Firefox (32-bit), Google Chrome (32-bit). Also see Browser Requirements.

For each supported language (English, German, Spanish, Japanese), separate trusted and untrusted versions are available for each supported browser (Microsoft Internet Explorer 32-bit and 64-bit, and Mozilla Firefox). One, some, or all of these versions of the Authentic Browser plug-in can be installed on a client machine. Each will be displayed as a separate plug-in in the browser's Add-on Manager.

The different versions for English (EN), German (DE), Spanish (ES), and Japanese (JA) are listed below with their class IDs (for CAB files) and MIME types (for XPI and CRX files). You will need to specify the relevant class ID or MIME type in the HTML page for Authentic Plug-in.

- ***CAB files and their Class IDs (identical Class IDs for 32-bit and 64-bit Internet Explorer)***

EN	Trusted	B4628728-E3F0-44a2-BEC8-F838555AE780
EN	Untrusted	A5985EA9-3332-4ddf-AD7F-F6E98BFEAF94
DE	Trusted	91DDF44A-DFD1-4F47-8EE3-4CBE874584F7
DE	Untrusted	28A640E8-EAEE-4B5D-BEBE-BFA956081E66
ES	Trusted	23B503E7-269B-45CE-BAB2-22AA97BED8E2
ES	Untrusted	8AD3EF86-AC1E-4574-8C13-DE5B6CBECEBE
JA	Trusted	5B15DB5A-1720-4264-BB65-70C3F7A860DA
JA	Untrusted	4B9512D2-A3D3-46e3-82C1-34248BBDCE58

- ***MIME types for XPI and CRX files***

EN	Trusted	application/x-authentic-scriptable-plugin
EN	Untrusted	application/x-authentic-scriptable-plugin-untrusted
DE	Trusted	application/x-authentic-scriptable-plugin-german
DE	Untrusted	application/x-authentic-scriptable-plugin-untrusted-german
ES	Trusted	application/x-authentic-scriptable-plugin-spanish
ES	Untrusted	application/x-authentic-scriptable-plugin-untrusted-spanish
JA	Trusted	application/x-authentic-scriptable-plugin-japanese

JA	Untrusted	`application/x-authentic-scriptable-plugin-untrusted-japanese`

You can download one or more of these versions from the Altova Website according to your requirements.

Note the following points about the various Authentic Browser versions:

- All versions are Unicode versions and each provides full support of multiple character-sets in the XML document. Unicode versions require the use of Windows 8, Windows 7, Windows Vista, or Windows XP on the client workstation.
- Although there is a separate CAB file each for the 32-bit and 64-bit Internet Explorer versions, the Class IDs of the two `.CAB` files (for 32-bit and 64-bit IE browsers) are identical and are as given in the table above for various EN/DE/ES/JA and Trusted/ Untrusted versions.
- The **Trusted version** does not allow access to local files and is, therefore, marked as being "safe for scripting". It can be used in a browser-based scenario and can be invoked from any web page without causing security alerts on the client side.
- The **Untrusted version** is intended for intranet deployment, or for using the Authentic Browser Edition as an ActiveX control in your application. It provides access to local files and is therefore not marked as being "safe for scripting". If you try to use this version from within a browser window, it will ask the user for permission. You can decide whether ActiveX controls should be disabled or enabled, or whether the browser should prompt for permission to enable ActiveX controls (see the browser-specific section for details: Internet Explorer 9).

Note: To install and configure your browser service (e.g. Microsoft's Internet Information Services), refer to the supplier's documentation.

Chapter 4

Server Setup

Server Setup

Setting up the server for Authentic Browser entails the following steps.

Configuring the Browser Service
Install and configure the browser service of your server. If you use Microsoft's Internet Information Services (IIS), note that the installation process creates a default directory called `Inetpub` with subdirectories. The root directory of the server would then be: `//Inetpub/wwwroot`. This is the directory reached with the IP Address of the server if you do not specify (in your browser service configuration) some other directory as the root directory. For details about installing and configuring your browser service, see the supplier's documentation.

Setting up the XSD, XML, and SPS/PXF files
An SPS file enables the XML document to be displayed in an editable format in Authentic View. It is based on an XML Schema (XSD file) and is designed in Altova's StyleVision product. The SPS file, together with the XSD file and XML document to be edited must be stored at a network location that is accessible to all client machines (typically on the Authentic Browser server). The section XSD, XML, and SPS/PXF Files describes this server preparation step in more detail.

Creating the HTML page for Authentic Plug-in
The HTML Page for Authentic Plug-in is the access point for Authentic View editing. It contains instructions to access the XML document, and it serves as a container for the Authentic View window in which the XML document is loaded and edited. To access this page, its URL is typed into client browsers. This HTML page must be created correctly and stored on the server. How to create the HTML page is described in the section, HTML Page for Authentic Plug-in.

Storing extension packages for on-demand installation
Download Authentic Browser (a zipped CAB, XPI, or CRX file) from the Altova website to any location on the server. If you are deploying the Enterprise edition of Authentic Browser, then the package must be stored on the server for which the Enterprise license has been registered. **Do not unzip this file.** On the website, for each language version (English, German, Spanish, and Japanese), there are trusted and untrusted versions of Authentic Browser, each in four formats (CAB 32-bit, CAB 64-bit, XPI, and CRX). For information on which file format and which version to select, see the sub-sections of this section.

Note: Before installing Authentic Browser, make sure that no previous version of Authentic Browser is running. Otherwise the new version might not be registered correctly, and this could result in a defective installation. If that happens, register the plug-in by running: `regsvr32 C:\Windows\Downloaded Program Files\AuthenticPlugin.dll`. (Note that you need administrative privileges to run the program `regsvr32.exe`.)

1 IIS: Configuring the Browser Service

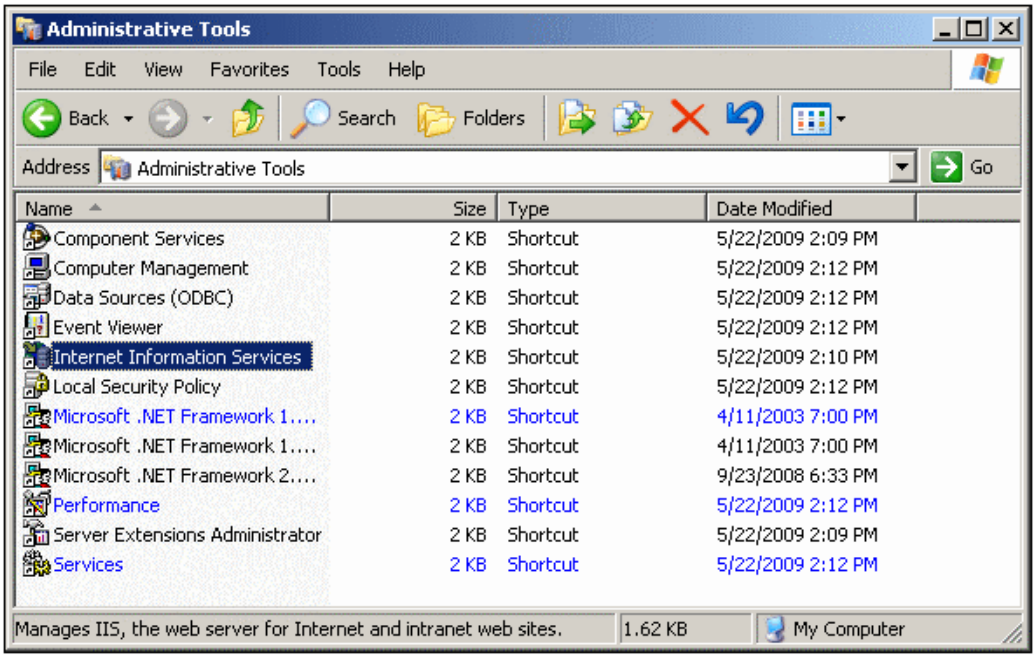
Microsoft's Internet Information Services (IIS) 6 serves up only file types that are defined in the MIME types for that specific site (website or folder). Required filetypes, therefore, must be added to the list of MIME types for a given site.

For working with Authentic Browser, the following filetypes are required and must be added:

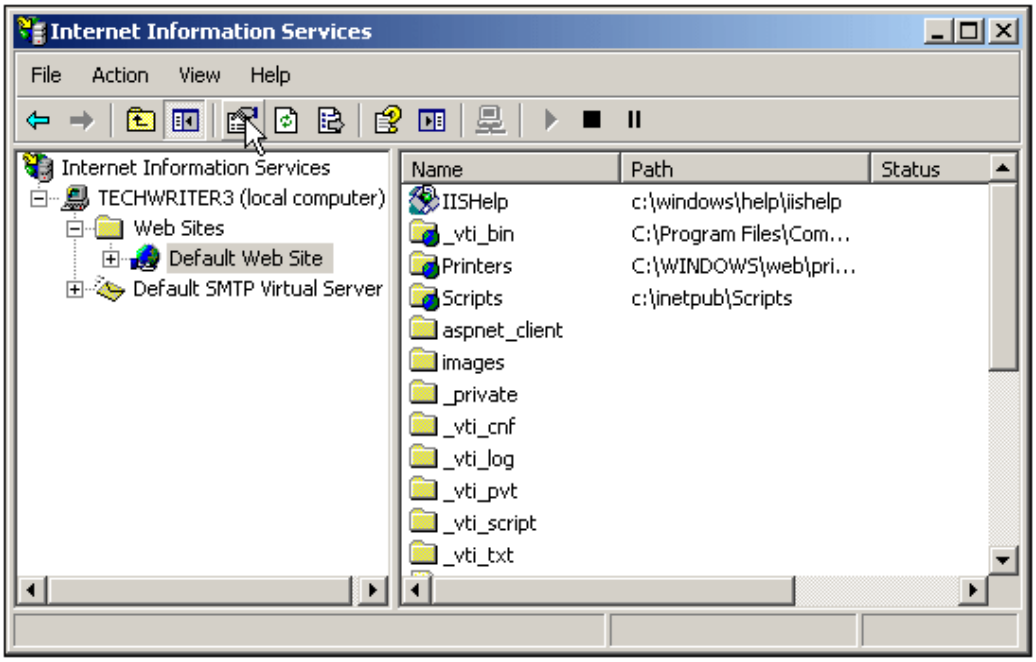
File extension	MIME type	Comment
xsd	text/plain	
sps	text/plain	
pxf	application/x-zip-compressed	
xpi	application/x-xpinstall	For Firefox.

Adding MIME types for a site in Internet Information Services
To add a MIME type to the list of MIME types for a particular site on a Windows XP machine, do the following. The process is similar on other supported systems (Windows Vista, Windows 7 and Windows 8).

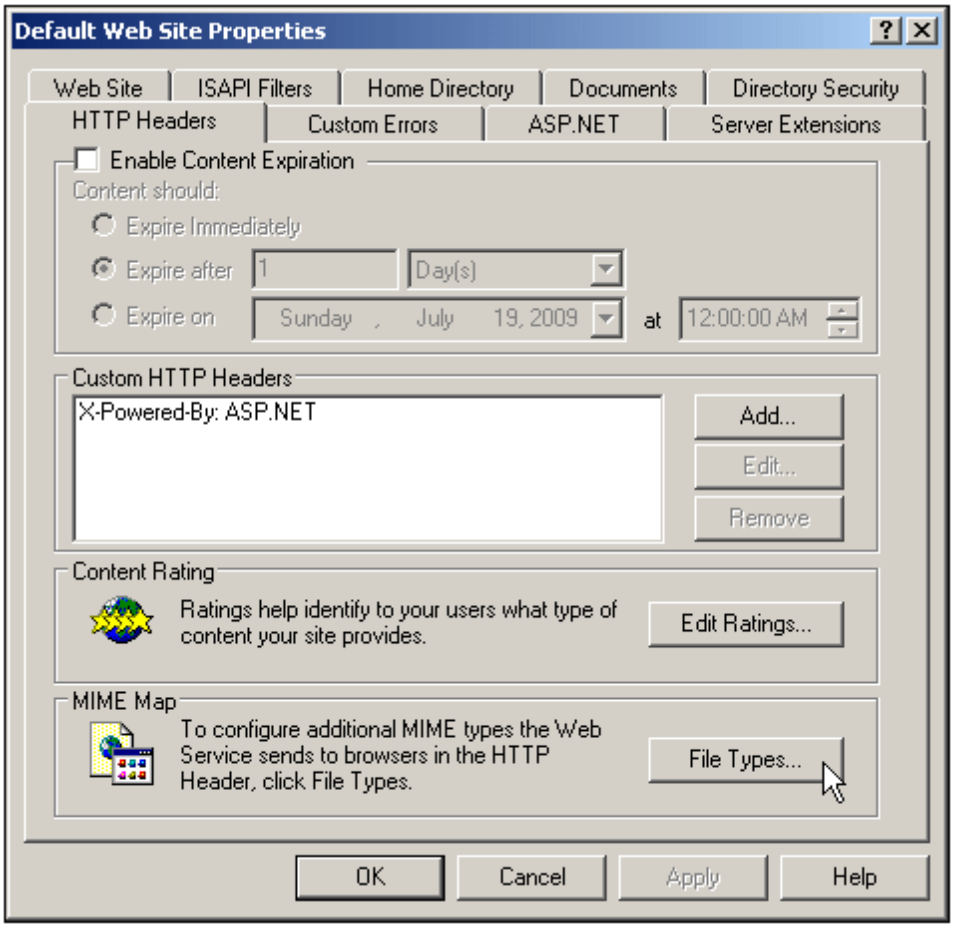
1. Open Control Panel and double-click Administrative Tools.
2. In the folder that pops up (*screenshot below*), double-click Internet Information Services.



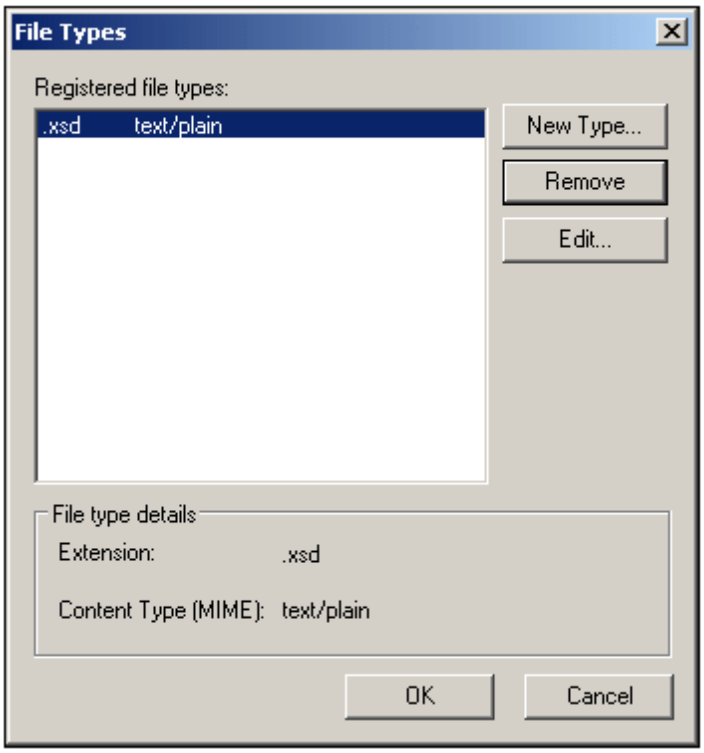
3. In the Internet Information Services (IIS) folder that appears (*screenshot below*), first select the required site (website or folder) in the folders pane (at left) and then click the **Properties** icon (*under cursor in screenshot below*) or the **Properties** command from the context menu (accessed by right-clicking).



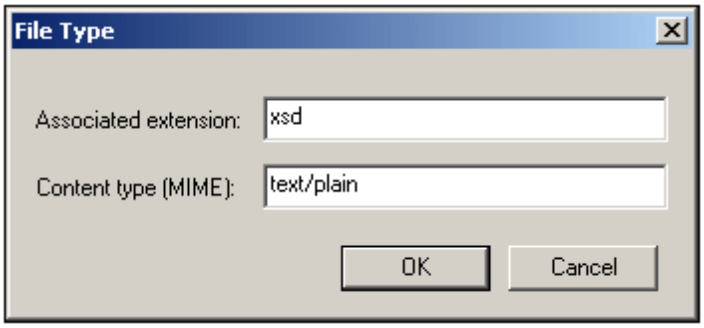
4. In the HTTP Headers tab of the Properties dialog, click the **File Types** button in the MIME Map pane (*see screenshot below*).



5. In the File Types dialog (*screenshot below*), click the **New Type** button.



6. In the dialog that pops up enter the required extension and its MIME type. See the table above for required filetypes and their corresponding MIME types.



7. Confirm with **OK**.

For a description of how to set up remote access to a folder on an internal server via Web-DAV, see this Windows IT Pro article.

2 XSD, XML, and SPS/PXF Files

The main purpose of implementing an Authentic Browser plug-in project is to view and edit an XML document in Authentic View format. This format hides the XML markup of an XML document and enables the editing of the document in a custom-designed WYSIWYG interface. The layout and data input mechanisms of an XML document in Authentic View is defined in an SPS file. The SPS file, the XML file, and the XML Schema file (XSD file) on which both the SPS and XML documents are based must therefore be stored on the network so that they are accessible to Authentic Browser clients.

SPS files

An SPS file is created in Altova's StyleVision product. It is based on the same XML Schema as that on which the XML file is based. In StyleVision, an SPS designer can use drag-and-drop and page layout mechanisms to design the SPS. The SPS file determines the layout and data input mechanisms of the XML document when it is displayed in Authentic View.

In the HTML Page for Authentic Plug-in, the locations of the XML file, SPS file, and XSD file are specified. The XML file is loaded in the Authentic View window (in the HTML page) and displayed according to the design specified in the SPS file. Both the XML file and the SPS file are based on the same XML Schema (though the XML file can be based on a part of the larger schema the SPS uses).

Note: For Authentic Browser to work correctly, you must **ensure that a schema is assigned as a main schema** in the SPS file. For detailed information about creating StyleVision Power Stylesheets (SPSs), see the user manual of Altova's StyleVision product. You can also read a description of the StyleVision product at the Altova website

XML files

The Authentic View user edits the XML document on a client machine by accessing the HTML Page for Authentic Plug-in. This page contains an Authentic View window in which the XML file is loaded. The Authentic View user edits the document in the client browser and saves changes back to the XML document.

XSD files

The XSD file serves two purposes. It is used to validate the XML document, and it provides the generic document structure on which the SPS is based. Both the XML file and the SPS file reference the XSD file. So the XSD file must also be stored in at the correct location as referenced by the XML and SPS files.

Storage location of XSD, XML and SPS/PXF files

The XSD, XML, and SPS (or PXF) files must be saved at network locations that can be accessed by all client machines. The best location is the Authentic Browser server. It is best to locate these files in a single folder on the server and to specify the references within them to each other as relative paths. So, for example, the XML file should be created with the XSD reference within it specified as a relative path.

Note about DB-based SPSs
If Authentic Browser will be used to view or edit databases (DBs) using a DB-based StyleVision Power Stylesheet (SPS), then you need to make the following settings to ensure that the client connects correctly to the DB.

Connection information in the SPS: All the information required to connect to the DB is stored in a connection string in the SPS. The connection string in the SPS is created in StyleVision at the time you create the StyleVision Power Stylesheet. The mechanism used to connect to MS Access DBs is different than for other DBs. For other DBs, ADO connections are used. The settings you need to make for these two types of connection mechanism are described below.

- ***For MS Access DBs:*** In the connection string for MS Access DBs, a UNC path must be used so that clients can correctly connect to the DB. This UNC path is specified when the StyleVision Power Stylesheet is built in StyleVision. You must, however, make sure that the folder containing the DB (or some ancestor folder) is set up for sharing. (In Windows XP, the sharing settings are accessed by right-clicking the folder and selecting **Sharing and Security**.) You must also enable Advanced File Sharing on this machine (**My Computer | Tools | Folder Options**, then uncheck Simple File Sharing). No settings are required on clients.

  Note: The format of the UNC path used in the connection string is: `\\servername \sharename\path\file.mdb`, where `servername` is the name of the server, `sharename` is the name of the shared folder (specified in the Share settings you make for the shared folder on the server), `path` is the path to the DB, and `file.mdb` is the name of an MS Access DB in the shared folder or a descendant folder of the shared folder.

- ***For ADO connections:*** The ADO connection string in the SPS is specified when the SPS is built in StyleVision. It contains all the required connection information, including security information. You must ensure that the driver used when testing the connection string in StyleVision is also installed on all client machines that will host Authentic Browser. This enables the SPS to correctly connect to the DB from clients. Note also that ADO database connections will only work with local file paths and not with `http://` URLs.

Note about PXF Files
An SPS design that uses XSLT 2.0 can be saved as a Power XML Form (PXF) file. The PXF file format has been specially developed by Altova to package the SPS design with related files (such as the schema file, source XML file, image files used in the design, and XSLT files for transformation of the source XML to an output format). The benefit of the PXF file format is that all the files required for Authentic View editing and for the generation of output from Authentic View can be conveniently distributed in a single file.

Note: If a PXF file is located on a web server and will be used with the Authentic Browser Plug-in, you must ensure that the server does not block the file. You can do this by adding (via the IIS administration panel, for example) the following MIME type for PXF (`.pxf`) file extensions: `application/x-zip-compressed`.

3 HTML Page for Authentic Plug-in

The HTML Page for Authentic Plug-in carries out the following important functions:

1. The first time that the HTML Page for Authentic Plug-in is opened on a client, code within the HTML page causes the Authentic Browser plug-in to be downloaded from the server to the client and installed on the client. This code has to be written correctly to identify the correct file on the server. The format of the file that is downloaded (CAB, XPI, or CRX) will depend on the client's browser. The code will also be different because Internet Explorer and Firefox process code differently than each other.
2. Code within the HTML page sets up the Authentic View interface within the browser window, including the dimensions of the Authentic View interface, and the locations of the XML, XSD, and SPS files.
3. It specifies the XML file to edit, as well as the schema file and SPS on which the XML file is based.
4. It contains, within HTML `SCRIPT` elements, definitions for subroutines and the handling of events. For example, it can be specified what action to carry out when a button in the HTML page is clicked. See Internet Explorer Example 1: Simple and Firefox Example 1: Simple for examples.

Note: If you are deploying the **Enterprise edition of Authentic Browser**, then the HTML page must be installed on the server for which the Enterprise license has been registered.

Embedding Authentic Browser

To use the Authentic Browser plug-in, an object that identifies the plug-in must be embedded in the HTML page that downloads the plug-in. This is done using the following HTML elements and attributes:

- Internet Explorer: `<OBJECT clsid="clsid:<CLSID>"/>`
- Firefox: `<embed type="<MIMEType>"/>`

For information on what CLSID and MIME type values to use, see Authentic Browser Versions.

This section

The sub-sections of this section describe how the functions listed above are to be implemented in the HTML page. These sections are organized at the first level by browser-type because of the different ways used to download the Authentic Browser DLL for particular browsers:

- Licensing for Enterprise Edition describes the licensing mechanism for the Enterprise Edition of Authentic Browser
- Internet Explorer describes how to download a CAB file (having a `.cab` extension) via an HTML `OBJECT` element.
- Firefox describes how to download an XPI file (extension `.xpi`) via an HTML `EMBED` element.
- Independent browsers describes how to determine the browser type making the request and how to then download the correct plug-in version for that browser.

In these sub-sections, you will find descriptions and examples of how the HTML `OBJECT` , `EMBED`, and `SCRIPT` elements are to be used, as well as examples of complete HTML pages that invoke the Authentic Browser plug-in. For information about individual objects, see the respective Object

descriptions in the Reference section.

3.1 Licensing for Enterprise Edition

Authentic Browser is available in two editions:

1. **Enterprise Edition**, which enables advanced SPS features and requires a license
2. **Community Edition**, for which no license is required.

The license for Authentic Browser Enterprise Edition can be purchased at the Altova Website.

Overview of setting up the Enterprise Edition license

Given below is a list summarizing the steps you must take to correctly set up your license information for Authentic Browser Enterprise Edition.

- Authentic Browser Enterprise Edition requires the following valid license information: (i) the ***server name*** for which the license is valid; (ii) the ***name of the company*** to which the license has been registered, and (iii) the ***license key***. This information will be sent to you in the License Email you will receive on purchasing your Authentic Browser Enterprise Edition license.
- The license information must be located in the HTML Page for Authentic Plug-in. How exactly this is to be done in the HTML page is explained below. (Note that there is no specific license key file that Authentic Browser will reference.)
- If the licensing information supplied in the HTML Page for Authentic Plug-in is valid, Enterprise Edition functionality in the Authentic Browser File will be unlocked; otherwise, functionality will be limited to that of Community Edition.
- The HTML Page for Authentic Plug-in must be stored on the server for which the license is valid.

How to enter the license information

The three license key parameters (server name, company name, and license key) must be entered in the HTML Page for Authentic Plug-in and can be done in the following ways:

- As parameter values of the `OBJECT` or `EMBED` elements. How to do this is described in the respective HTML page sections for Internet Explorer and Firefox.
- If the HTML page is to be browser-independent, the license parameters can be registered as shown in the code listing in the Browser-Independent Example.
- The license parameters can also be set on the object directly, as shown in the code listing below, which adapts the code listing in the Browser-Independent Example.

```
<SCRIPT LANGUAGE=javascript>
// event subscription if running on Firefox
if ( isFirefoxOnWindows() )
{
        objPlugIn.addEventListener("ControlInitialized",
InitAuthenticPluginPage, false);
}
</SCRIPT>
<SCRIPT LANGUAGE="javascript" FOR=objPlugIn EVENT="ControlInitialized">
// event subscription if running on Internet Explorer
if ( isIEOnWindows() )
{
        InitAuthenticPluginPage();
```

```
        }
        </SCRIPT>

        <SCRIPT type="text/javascript" LANGUAGE="javascript" >
        function InitAuthenticPluginPage( )
        {
        var serverstr='DevAuthBrowTest';
        var basedir='Authentic/';
        objPlugIn.LicServer = 'DevAuthBrowTest';
        objPlugIn.LicCompany = 'Altova';
        objPlugIn.LicKey = 'XXXXXXXXXX';
        objPlugIn.SchemaLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.xsd';
        objPlugIn.XMLDataLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.xml' ;
        objPlugIn.DesignDataLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.sps';
        objPlugIn.StartEditing();
        }
        </SCRIPT>
```

3.2 Internet Explorer

If the HTML page for the Authentic plug-in is opened in Internet Explorer (32-bit or 64-bit), then it must contain the following elements:

- An HTML `OBJECT` element, which (i) causes the correct DLL for the Authentic Plug-in (32-bit or 64-bit) to be downloaded from the server to the client, and (ii) specifies the dimensions of the Authentic View window in the client's browser. The `OBJECT` element contains the location of the Authentic Browser plug-in. **Note that the Authentic Plug-in is available in versions for 32-bit and 64-bit Internet Explorer (except 64-bit IE 10 and 11), so the correct Authentic Browser plug-in version (`.cab` file) must be downloaded to the client.** See the section, Browser-Independent Example, for example code which automatically checks the X-bit version of Internet Explorer and downloads the correct Authentic Plug-in.
- One or more HTML `SCRIPT` elements for defining subroutines and the handling of events. A `SCRIPT` element can be used to specify the XML document to be edited, and the XML Schema and SPS file on which the XML document is based.

This section is organized into the following sub-sections:

- The OBJECT Element, which describes how the HTML `OBJECT` element is to be used in an HTML page for the Authentic plug-in.
- The SCRIPT Element, which describes how HTML `SCRIPT` elements are to be used in an HTML page for the Authentic plug-in.
- Examples of full HTML pages: IE Example 1: Simple and IE Example 2: Sort a Table.

For information about individual objects, see the respective Object descriptions in the Reference section.

Note: The Authentic Browser plug-in is supported for **Internet Explorer 5.5 or higher**. However the 64-bit versions of Internet Explorer 10 and 11 are not supported.

3.2.1 The OBJECT Element

The `OBJECT` element has the following functions:

- It gives the Authentic Browser Plug-in a name (via the `id` attribute).
- It selects which version of the Authentic Browser Plug-in to use (with the value of the `classid` attribute).
- It specifies a `.CAB` file and version number (`codebase` attribute). The `.CAB` file contains a DLL, which registers a COM object (the Authentic Browser Plug-in) with an ID that is the value of the `classid` attribute. Typically, both the 32-bit and 64-bit versions of the Authentic Browser Plug-in will be stored on the server. Each version is identified by a different file name. The `codebase` attribute specifies the file name of the required `.CAB` file (*see code listing below*). The Class IDs for 32-bit and 64-bit `.CAB` files are identical to each other and are as given in the table for various EN/DE and Trusted/Untrusted versions.

  Filename for 32-bit version: `AuthenticBrowserEdition.CAB`
  Filename for 64-bit version: `AuthenticBrowserEdition_x64.CAB`
- It specifies the dimension of the Authentic View window in the client's browser (via the `style` attribute).
- It can specify any number of parameters.

Here is a sample HTML `OBJECT` element. It selects the Trusted Unicode version (with the value given in the `classid` attribute) and sets the dimensions of the Authentic View window in the client's browser to 600 x 500 pixels. All the attributes and parameters available to the `OBJECT` element are explained below.

Note: The version number given below may not be the current version number. See codebase below for details.

For 32-bit Authentic Browser Plug-in:

```
<OBJECT id="objPlugIn" style="WIDTH:600px; HEIGHT:500px"
   codeBase="http://yourserver/cabfiles/
AuthenticBrowserEdition.CAB#Version=12,3,0,0"
   classid="clsid:B4628728-E3F0-44a2-BEC8-F838555AE780">
   <PARAM NAME="XMLDataURL" VALUE="http://yourserver/OrgChart.xml">
   <PARAM NAME="SPSDataURL" VALUE="http://yourserver/OrgChart.sps">
   <PARAM NAME="SchemaDataURL" VALUE="http://yourserver/OrgChart.xsd">
</OBJECT>
```

For 64-bit Authentic Browser Plug-in:

```
<OBJECT id="objPlugIn" style="WIDTH:600px; HEIGHT:500px"
   codeBase="http://yourserver/cabfiles/
AuthenticBrowserEdition_x64.CAB#Version=12,3,0,0"
   classid="clsid:B4628728-E3F0-44a2-BEC8-F838555AE780">
   <PARAM NAME="XMLDataURL" VALUE="http://yourserver/OrgChart.xml">
   <PARAM NAME="SPSDataURL" VALUE="http://yourserver/OrgChart.sps">
   <PARAM NAME="SchemaDataURL" VALUE="http://yourserver/OrgChart.xsd">
</OBJECT>
```

id

The value of the `id` attribute is used as the name of the Authentic Browser Plug-in objects when these are used in scripts. For example, `objPlugIn.SchemaLoadObject.URL` is a call to the object that loads the schema file. See The SCRIPT element for more details.

`style`
This is the usual HTML `style` attribute, and is used to specify the dimensions of the Authentic View window in the client's browser.

`codebase`
The `codebase` attribute gives the location of the `.CAB` file. Note that there is a different `.CAB` file for the 32-bit Authentic Browser plug-in and for the 64-bit Authentic Browser Plug-in, named, respectively: `AuthenticBrowserEdition.CAB` and `AuthenticBrowserEdition_x64.CAB`.

The value of the optional `#Version` extension gives the version number of the component that is currently available on the server. If the client has an earlier version and a newer version is specified in the `codebase` attribute, the newer version is installed from the server. If the `#Version` extension is not specified, no update will take place until the component has been removed manually from the client. **The current version number of the component is listed with the properties of the `.dll` file of the component's `.CAB` file (right-click the file and select the Properties command)**.

`classid`
The Class IDs for 32-bit and 64-bit `.CAB` files are identical to each other and are as given in the list below for various EN/DE and Trusted/Untrusted versions.

From version 5.0 onwards of the Browser Plug-in, the `classid` value for Unicode versions from versions 5.0 onwards is different from that of previous Unicode versions. So, if you are updating the Unicode `.CAB` file on your server from a version prior to 5.0, make sure that you change the `classid` values in your HTML files. Note also that if a new `.CAB` file on the server has the same CLSID as that of a `.CAB` file which has been installed previously on the client, then the new `.CAB` file will not automatically replace the old one on the client. You must remove the previously installed `.CAB` file before downloading the new `.CAB` file. The CLSID values of different language versions are different.

Also, when selecting the version to download to a client, bear in mind that older operating systems may not be able to correctly install the Unicode versions.

Given below are the CLSID values of the current versions (English (EN) and German (DE)):

EN Trusted Unicode:
`clsid:B4628728-E3F0-44a2-BEC8-F838555AE780`

EN Untrusted Unicode:
`clsid:A5985EA9-3332-4ddf-AD7F-F6E98BFEAF94`

DE Trusted Unicode:
`clsid:91DDF44A-DFD1-4F47-8EE3-4CBE874584F7`

DE Untrusted Unicode:
`clsid:28A640E8-EAEE-4B5D-BEBE-BFA956081E66`

Parameters
Any number of the following parameters may be used.

LicServer
The name of the server for which the Authentic Browser Enterprise Edition license key is valid. (No license key is required for Authentic Browser Community Edition.)

LicKey
The license key for validating the use of Authentic Browser Enterprise Edition. (No license key is required for Authentic Browser Community Edition.)

LicCompany
The company name for validating the use of Authentic Browser Enterprise Edition. (No license key is required for Authentic Browser Community Edition.)

XMLDataURL
An absolute URL that gives the location of the XML file to be edited. For the Untrusted versions, you can also use a full local path.

XMLDataSaveURL
An absolute URL that gives the location where the XML file is to be saved. For the Untrusted versions, you can also use a full local path.

SPSDataURL
An absolute URL that gives the location of the StyleVision Power Stylesheet (`.sps` file). For the Untrusted versions, you can also use a full local path.

SchemaDataURL
An absolute URL that gives the location of the associated schema file. For the Untrusted versions, you can also use a full local path.

TextStateBmpURL
The folder where bitmap image for text-state icons are to be stored.

TextStateToolbarLine
The toolbar line in which text-state icons are to be placed. The default is 1.

AutoHideUnusedCommandGroups
Determines whether unused toolbar command groups should be hidden. The default is True.

ToolbarsEnabled
Specifies general support for toolbars. The default is True.

ToolbarTooltipsEnabled
Speficies whether Tooltips are enabled or not.

HideSaveButton
If set to True, removes the Save button from the Authentic toolbar, which is visible by default.

BaseURL
Gives the base URL for use with relative paths.

SaveButtonUsePOST
If set to True, the HTTP POST command is used instead of a PUT when saving the document.

EntryHelpersEnabled
If set to True the Authentic entry helpers are visible

EntryHelperSize
Width of the entry helper window in pixels.

EntryHelperAlignment
Specifies the location of the entry helpers relative to the document window.

`0` = Align toolbar at top of document
`1` = Align toolbar at left of document
`2` = Align toolbar at bottom of document
`3` = Align toolbar at right of document

EntryHelperWindows
Selects which of the entry helper sub-windows are visible.

`1` = Elements
`2` = Attributes
`4` = Entities

Any combination is allowed (bit-check)

SaveButtonAutoEnable
See Authentic.SaveButtonAutoEnable

LoaderSettingsFileURL
Gives the URL of the `LoaderSettingsFile` for package management.

3.2.2 The SCRIPT Element

The SCRIPT elements define the event handlers and subroutines that can be called from within the HTML file.

Here is a sample of a script for handling an event:

```
<SCRIPT LANGUAGE="javascript" FOR=objPlugIn EVENT="ControlInitialized">
      objPlugIn.SchemaLoadObject.URL = "http://yourserver/OrgChart.xsd"
      objPlugIn.XMLDataLoadObject.URL = "http://yourserver/OrgChart.xml"
      objPlugIn.DesignDataLoadObject.URL = "http://yourserver/OrgChart.sps"
      objPlugIn.StartEditing
</SCRIPT>
```

Here is a sample of a script that contains subroutines:

```
<SCRIPT ID=clientEventHandlers LANGUAGE=vbscript>
   Sub BtnOnClick
      objPlugIn.SchemaLoadObject.URL = "http://yourserver/OrgChart.xsd"
      objPlugIn.XMLDataLoadObject.URL = "http://yourserver/OrgChart.xml"
      objPlugIn.DesignDataLoadObject.URL = "http://yourserver/OrgChart.sps"
      objPlugIn.StartEditing
   End Sub
   Sub OnClickFind
      objPlugIn.FindDialog
   End Sub
   Sub BtnOnTestProp
      If objPlugIn.IsRowInsertEnabled Then
         msgbox "true"
      Else
         msgbox "false"
      End If
   End Sub
</SCRIPT>
```

LANGUAGE

The Authentic Browser Plug-in has been tested with JavaScript and VBScript.

Handling events

The value of the ID attribute of the OBJECT element in the HTML body is specified as the value of the FOR attribute. Authentic Browser Plug-in objects that are called must have a name that is this value.

For a list of events see also Events: Reference.

Subroutines

Subroutines can be created for any event that you wish to define in the HTML file. The Authentic Browser Plug-in object name must be the same as the value of the ID attribute of the OBJECT element in the HTML body. In the example above, the prefix is objPlugIn, which must be the value of the ID attribute of the OBJECT element.

The methods, properties, and sub-objects available in the Authentic Browser Plug-in are described in the reference section of this documentation.

3.2.3 IE Example 1: Simple

The HTML code below generates a page that has the following features:

- It installs the Trusted Unicode version of Authentic Browser on the client if this is not already installed.
- The body contains a window of 600px wide and 500px high into which the Authentic Browser is loaded.
- Below the Authentic Browser window is a row of four buttons.
- The Authentic View of `OrgChart.xml` is loaded.
- The **Find** and **Replace** buttons pop up the Find and Replace dialogs respectively.
- The **Save** button saves changes to a file called `SaveFile.xml` located in the root directory of the server
- The **Test Property** button tests a simple property

When this HTML page is opened on the client, the user can start editing the XML file `OrgChart.xml` and save the edited file as `SaveFile.xml`.

You may wish to use this simple HTML page to test whether Authentic Browser functions properly. If you do so, be sure to use the correct URLs to locate the the `CAB` file, the `xsd`, `xml`, and `sps` files, and any other resources on the server. Note that case-sensitivity might be an issue with some servers, so if there is a problem locating a file, check the casing of filenames and of commands in the code. You can expand or modify this example to build more complex solutions using Authentic Browser. Also see The OBJECT Element for details.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
    <title>Minimal XMLSpyDocEditPlugIn page</title>

    <!-- Script for handling the ControlInitialized event -->
    <SCRIPT LANGUAGE="javascript" FOR="objPlugIn" EVENT="ControlInitialized">
         objPlugIn.SchemaLoadObject.URL = "http://yourserver/OrgChart.xsd"
         objPlugIn.XMLDataLoadObject.URL = "http://yourserver/OrgChart.xml"
         objPlugIn.DesignDataLoadObject.URL = "http://yourserver/OrgChart.sps"
         objPlugIn.StartEditing()
    </SCRIPT>

    <!-- Script with subroutines -->
    <SCRIPT ID=clientEventHandlers LANGUAGE=vbscript>
      Sub OnClickFind
         objPlugIn.FindDialog
      End Sub

      Sub OnClickReplace
         objPlugIn.ReplaceDialog
      End Sub

      Sub BtnOnSave
         objPlugIn.XMLDataSaveUrl = "http://yourserver/SaveFile.xml"
         objPlugIn.Save
      End Sub

      Sub BtnOnTestProp
         If objPlugIn.IsRowInsertEnabled Then
            msgbox "true"
```

```
        Else
           msgbox "false"
        End If
      End Sub
    </SCRIPT>
  </head>

  <body>
    <!-- Object element has id with value that must be used -->
    <!-- as name of Authentic Browser Plug-in objects -->
    <!-- Classid selects the Trusted Unicode version -->
    <OBJECT id="objPlugIn"
    <!-- CodeBase selects 32-bit CAB file (AuthenticBrowserEdition.CAB) -->
    <!-- or 64-bit CAB file (AuthenticBrowserEdition_x64.CAB) -->
      CodeBase="http://yourserver/AuthenticBrowserEdition.CAB#Version=12,3,0,0"
    <!-- Class Id for 32-bit and 64-bit CAB files is the same -->
      Classid="clsid:B4628728-E3F0-44a2-BEC8-F838555AE780" width="600"
height="500">
    </OBJECT>
    <p>
      <input type="button" value="Find" name="B4" onclick="OnClickFind()">
      <input type="button" value="Replace" name="B5" onclick="OnClickReplace()">
      <input type="button" value="Save" name="B6" onclick="BtnOnSave()">
      <input type="button" value="Test property" name="B7"
onclick="BtnOnTestProp">
    </p>
  </body>
</html>
```

3.2.4 IE Example 2: Sort a Table

This is an example HTML page with an embedded JavaScript. The sample requires the Authentic Browser Plug-in (CAB file) to be installed on your computer. Note that case-sensitivity might be an issue with some servers, so if there is a problem locating a file, check the casing of filenames and of commands in the code.

The code shows:

- How to access the browser plug-in. Modify the code to refer to your CAB file and the class identifier (`CLSID`) of your browser plug-in version (trusted or untrusted).
- How to load a file into the browser plug-in. Modify the code to refer to your sample document.
- Implement buttons for simple cursor positioning.
- Implement more complex commands like the sorting of tables.
- How to use the `SelectionChanged` event.

Also see The OBJECT Element for details.

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
      <title>test page For Authentic Browser Plug-in</title>

      <SCRIPT LANGUAGE="javascript" For="objPlugIn" EVENT="ControlInitialized">
         var strSampleRoot = "http://myRoot/myPath/myDocBaseName";
         objPlugIn.SchemaLoadObject.URL = strSampleRoot + ".xsd";
         objPlugIn.XMLDataLoadObject.URL = strSampleRoot + ".xml";
         objPlugIn.DesignDataLoadObject.URL = strSampleRoot + ".sps";
         objPlugIn.StartEditing();
      </SCRIPT>

      <SCRIPT ID="clientEventHandlers" LANGUAGE="javascript">
         var objCurrentRange = Null;

         Function BtnDocumentBegin()
{ objPlugIn.AuthenticView.DocumentBegin.Select(); }
         Function BtnDocumentEnd()
{ objPlugIn.AuthenticView.DocumentEnd.Select(); }
         Function BtnWholeDocument()
{ objPlugIn.AuthenticView.WholeDocument.Select(); }
         Function BtnSelectNextWord()
{ objPlugIn.AuthenticView.Selection.SelectNext(1).Select(); }
         Function BtnSortDepartmentOnClick()
         {
            var objCursor = Null;
            var objTableStart = Null;
            var objBubble = Null;
            var strField1 = "";
            var strField1 = "";
            var nColIndex = 0;
            var nRows = 0;

            objCursor = objPlugIn.AuthenticView.Selection;
            If (objCursor.IsInDynamicTable())
```

```
            {
               // calculate current column index
               nColIndex = 0;
               While (True)
               {
                  try { objCursor.GotoPrevious(11); }
               catch (err) { break; }
               nColIndex++;
            }

            // GoTo begin of table
            objTableStart = objCursor.ExpandTo(9).CollapsToBegin().Clone();

            // count number of table rows
            nRows = 1;
            While (True)
            {
               try { objTableStart.GotoNext(10); }
            catch (err) { break; }
            nRows++;
            }

            // bubble sort through table
            For (var i = 0; i < nRows - 1; i++) {
               for(var j = 0; j < nRows-i-1; j++) {
                  objBubble = objCursor.ExpandTo(9).CollapsToBegin().Clone();
                  // Select correct column in jth table row
objBubble.GotoNext(6).Goto(10,j,2).Goto(11,nColIndex,2).ExpandTo(6);
                  strField1 = objBubble.Text;
                  strField2 =
objBubble.GotoNext(10).Goto(11,nColIndex,2).ExpandTo(6).Text;
                  if(strField1 > strField2) {
                      if(!objBubble.MoveRowUp()) {
                             alert('Table row move is not allowed!');
                             return;
                      }
                  }
               }
            }
         }
      </SCRIPT>
   </head>

   <body>
      <Object id="objPlugIn"
      <!-- CodeBase selects 32-bit CAB file (AuthenticBrowserEdition.CAB) -->
      <!-- or 64-bit Cab file (AuthenticBrowserEdition_x64.CAB) -->
         codeBase="http://myCabfileLocation/
AuthenticBrowserEdition.CAB#Version=12,3,0,0"
      <!-- Class Id for 32-bit and 64-bit CAB files is the same -->
         classid="clsid:B4628728-E3F0-44a2-BEC8-F838555AE780"
         width="100%"
         height="80%"
```

```
         VIEWASTEXT>
         <PARAM NAME="EntryHelpersEnabled" VALUE="TRUE">
         <PARAM NAME="SaveButtonAutoEnable" VALUE="TRUE">
      </Object>
      <TABLE>
         <TR>
            <TD><Input Type="button" value="Goto Begin" id="B1"
onclick="BtnDocumentBegin()"></TD>
            <TD><Input Type="button" value="Goto End" name="B2"
onclick="BtnDocumentEnd()"></TD>
            <TD><Input Type="button" value="Whole Document" name="B3"
onclick="BtnWholeDocument()"></TD>
            <TD><Input Type="button" value="Select Next Word" name="B4"
onclick="BtnSelectNextWord()"></TD>
         </TR>
         <TR>
            <TD><Input Type="button" value="Sort Table by this Column" id="B6"
onclick="BtnSortDepartmentOnClick()"</TD>
         </TR>
      </TABLE>
      <TABLE id=SelTable border=1>
         <TR><TD id=SelTable_FirstTextPosition></TD><TD
id=SelTable_LastTextPosition></TD></TR>
         <TR><TD id=SelTable_FirstXMLData></TD><TD
id=SelTable_FirstXMLDataOffset></TD></TR>
         <TR><TD id=SelTable_LastXMLData></TD><TD
id=SelTable_LastXMLDataOffset></TD></TR>
         <TR><TD id=SelTable_Text></TD></TR>
      </TABLE>
   </body>

   <SCRIPT LANGUAGE=javascript For=objPlugIn EVENT=selectionchanged>
      var CurrentSelection =  Null;
      CurrentSelection = objPlugIn.AuthenticView.Selection;
      SelTable_FirstTextPosition.innerHTML = CurrentSelection.FirstTextPosition;
      SelTable_LastTextPosition.innerHTML = CurrentSelection.LastTextPosition;
      SelTable_FirstXMLData.innerHTML =
CurrentSelection.FirstXMLData.Parent.Name;
      SelTable_FirstXMLDataOffset.innerHTML =
CurrentSelection.FirstXMLDataOffset;
      SelTable_LastXMLData.innerHTML = CurrentSelection.LastXMLData.Parent.Name;
      SelTable_LastXMLDataOffset.innerHTML = CurrentSelection.LastXMLDataOffset;
   </SCRIPT>

</html>
```

3.3 Firefox

If the HTML page for the Authentic plug-in is to be opened in Firefox, then it must contain the following elements:

- An HTML `EMBED` element, which (i) causes the DLL for the Authentic Plug-in to be downloaded from the server to the client, and (ii) specifies the dimensions of the Authentic View window in the client's browser. The `EMBED` element contains the location of the Authentic Browser plug-in.
- One or more Event Listeners for defining subroutines and handling events. Event listeners are constructed within `SCRIPT` elements.

This section is organized into the following sub-sections:

- The EMBED Element, which describes how the HTML `EMBED` element is to be used in an HTML page for the Authentic plug-in.
- Adding Event Listeners, which describes how Event Listeners are to be used in an HTML page for the Authentic plug-in.
- Examples of full HTML pages: Firefox Example 1: Simple and Firefox Example 2: Sort a Table.

For information about individual objects, see the respective Object descriptions in the Reference section.

Note:

- The Authentic Browser plug-in is supported for **Firefox**.
- Ensure that the MIME type for XPI and/or CRX files has been added in the browser service of your server to the list of MIME types for the site you will use.

3.3.1 The EMBED Element

The EMBED element has the following functions:

- It gives the Authentic Browser Plug-in a name (via the id attribute).
- It selects which version of the Authentic Browser Plug-in to use (via the type attribute).
- It specifies an XPI file or CRX file (which is the Authentic Browser Plug-in DLL) via the PluginsPage attribute. The value of the PluginsPage attribute is a path that locates the XPI or CRX file.
- It specifies the dimension of the Authentic View window in the client's browser (via the height and width attributes).

Here is a sample HTML EMBED element. It selects the English-language Trusted version (with the value of the type attribute) and sets the width and height of the Authentic View window in the client's browser to 100% and 60%, respectively.

```
<embed
    id="objPlugIn"
    type="application/x-authentic-scriptable-plugin"
    width="100%"
    height="60%"
    PluginsPage="http://your-server-including-path/
AuthenticFirefoxPlugin_trusted.xpi"
    [Name-Of-Parameter="Value of Parameter"]/>
```

id
The value of the id attribute is used as the name of the Authentic Browser Plug-in objects when these are used in scripts. For example, objPlugIn.SchemaLoadObject.URL is a call to the object that loads the schema file.

type
This value is the MIME type of the required Authentic Browser version. In the code listing above, the value application/x-authentic-scriptable-plugin identifies the English-language Trusted version. See the section, Authentic Browser Versions for a list of the available versions and their MIME types.

width, height
These attributes specify the dimensions of the Authentic View window to be created within the browser window.

PluginsPage
The value of this attribute specifies the location of the Authentic Browser XPI file (for Firefox) on your server. Make sure that you use the correct path in the URL that locates the the XPI file. Case-sensitivity might be an issue with some servers, so if there is a problem locating a file, check the casing of paths and filenames.

LicServer
The name of the server for which the Authentic Browser Enterprise Edition license key is valid. (No license key is required for Authentic Browser Community Edition.)

LicKey
The license key for validating the use of Authentic Browser Enterprise Edition. (No license key is required for Authentic Browser Community Edition.)

LicCompany
The company name for validating the use of Authentic Browser Enterprise Edition. (No license key is required for Authentic Browser Community Edition.)

Parameters
A parameter (*see list below*) can be used in the following ways:

- By giving the parameter name and its value as an attribute-value pair of the EMBED element. For example, the parameter ToolbarsEnabled can be specified with a value of true by adding the attribute-value pair ToolbarsEnabled="true" to the EMBED element. See listing above.
- A PARAM element can be specified as a child of the OBJECT element, as in the example below:

```
<object id="objPlugIn"
        type="application/x-authentic-scriptable-plugin"
        width="100%"
        height="60%" >
    <param name="ToolbarsEnabled"
        value="true"/>
</object>
```

  Note, however, that there is a drawback if parameters are specified in this way—that is, as a child of the OBJECT element. Since Firefox does not accept the PLUGINSPAGE attribute of the OBJECT element, Firefox will have no reference to the Authentic Browser XPI file on the server and cannot start installation of the Authentic Browser plug-in on the client. Therefore, this way of specifying parameters is of use only on clients where the plug-in has already been installed.

XMLDataURL
An absolute URL that gives the location of the XML file to be edited. For the Untrusted versions, you can also use a full local path.

XMLDataSaveURL
An absolute URL that gives the location where the XML file is to be saved. For the Untrusted versions, you can also use a full local path.

SPSDataURL
An absolute URL that gives the location of the StyleVision Power Stylesheet (.sps file). For the Untrusted versions, you can also use a full local path.

SchemaDataURL
An absolute URL that gives the location of the associated schema file. For the Untrusted versions, you can also use a full local path.

TextStateBmpURL
The folder where bitmap image for text-state icons are to be stored.

TextStateToolbarLine
The toolbar line in which text-state icons are to be placed. The default is 1.

AutoHideUnusedCommandGroups
Determines whether unused toolbar command groups should be hidden. The default is True.

ToolbarsEnabled
Specifies general support for toolbars. The default is True.

ToolbarTooltipsEnabled
Speficies whether Tooltips are enabled or not.

HideSaveButton
If set to True, removes the Save button from the Authentic toolbar, which is visible by default.

BaseURL
Gives the base URL for use with relative paths.

SaveButtonUsePOST
If set to True, the HTTP POST command is used instead of a PUT when saving the document.

EntryHelpersEnabled
If set to True the Authentic entry helpers are visible

EntryHelperSize
Width of the entry helper window in pixels.

EntryHelperAlignment
Specifies the location of the entry helpers relative to the document window.

`0` = Align toolbar at top of document
`1` = Align toolbar at left of document
`2` = Align toolbar at bottom of document
`3` = Align toolbar at right of document

EntryHelperWindows
Selects which of the entry helper sub-windows are visible.

`1` = Elements
`2` = Attributes
`4` = Entities

Any combination is allowed (bit-check)

SaveButtonAutoEnable
See Authentic.SaveButtonAutoEnable

LoaderSettingsFileURL
Gives the URL of the `LoaderSettingsFile` for package management.

3.3.2 Adding Event Listeners

The following SCRIPT element defines an event listener and registers it with the plug-in object. The event listener function will be called each time the specified event is triggered inside the plug-in.

```
<SCRIPT LANGUAGE="javascript">
var selCount = 0;
function OnSelectionChanged()
{
	selCount = selCount + 1;
	selectionCounter.value = "SelectionCount = " + selCount;
}
var objPlugIn = document.getElementById('objPlugIn');
objPlugIn.addEventListener("selectionchanged", OnSelectionChanged, false)
</SCRIPT>
```

For a list of events see also Events: Reference.

Language
The Authentic Browser Plug-in has been tested with JavaScript and VBScript.

Event listeners
For information about event listeners, see the relevant W3C Recommendation.

Authentic Browser object model
The methods, properties, and sub-objects available in the Authentic Browser Plug-in are described in the reference section of this documentation.

3.3.3 Firefox Example 1: Simple

The HTML code below generates a page that has the following features:

- It installs the Trusted version of Authentic Browser for Firefox on the client if this is not already installed.
- The Authentic Browser window within the page has a width that is 100% that of the browser window and 60% of its height.
- Below the Authentic Browser window is a row of four buttons.
- The Authentic View of `OrgChart.xml` is loaded.
- The **Find** and **Replace** buttons pop up the Find and Replace dialogs respectively.
- The **Save** button saves changes to a file called `SaveFile.xml` located in the root directory of the server.
- The **Test Property** button tests a simple property.

When this HTML page is opened on the client, the user can start editing the XML file `OrgChart.xml` and save the edited file as `SaveFile.xml`.

You may wish to use this simple HTML page to test whether Authentic Browser functions properly. If you do so, be sure to use the IP Address and the correct path to the respective files in the URLs that locate the the `XPI` file, the `xsd`, `xml`, and `sps` files, and any other resource on the server. Note that case-sensitivity might be an issue with some servers, so if there is a problem locating a file, check the casing of filenames and of commands in the code. You can expand or modify this example to build more complex solutions using Authentic Browser.

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
   <title>Minimal Authentic Browser PlugIn page</title>

</head>

<!-- to disable the fast-back cache in Firefox, define an unload handler -->
<BODY id="bodyId" onunload="Unload()">
   <!-- Embed element has id with value that must be used -->
   <!-- as name of Authentic Browser Plug-in objects -->
   <!-- type selects the Trusted Unicode version -->
   <embed
       id="objPlugIn"
       type="application/x-authentic-scriptable-plugin"
       width="100%"
       height="60%"
       PLUGINSPAGE="http://your-server-including-path/
AuthenticFirefoxPlugin_trusted.xpi"
       LicServer="DevAuthBrowTest"
       LicCompany="Altova"
       LicKey="XXXXXXXX" />
   <!-- Script with subroutines -->
   <SCRIPT LANGUAGE="javascript">
       var objPlugIn = document.getElementById('objPlugIn');
       function OnClickFind()
       {
              objPlugIn.FindDialog();
       }

       function OnClickReplace()
       {
```

```
                objPlugIn.ReplaceDialog();
        }

        function BtnOnSave()
        {
                objPlugIn.XMLDataSaveUrl = "http://your-server/Authentic/
SaveFile.xml"
                objPlugIn.Save()
        }

        function BtnOnTestProp()
        {
                alert ( objPlugIn.IsRowInsertEnabled );
        }
        function Unload()
        {
        }
        function InitAuthenticPluginPage( )
        {
                var serverstr='your-server/';
                var basedir='Authentic/';
                objPlugIn.SchemaLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.xsd';
                objPlugIn.XMLDataLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.xml' ;
                objPlugIn.DesignDataLoadObject.URL = 'http://' + serverstr +
basedir + 'OrgChart.sps';
                objPlugIn.StartEditing();
        }
        // event subscription if running on Firefox
        objPlugIn.addEventListener("ControlInitialized", InitAuthenticPluginPage,
false);
    </SCRIPT>
    <p>
      <input type="button" value="Find" name="B4" onclick="OnClickFind()">
      <input type="button" value="Replace" name="B5" onclick="OnClickReplace()">
      <input type="button" value="Save" name="B6" onclick="BtnOnSave()">
      <input type="button" value="Test property" name="B7"
onclick="BtnOnTestProp()">
    </p>
</body>
</html>
```

Note: The script above contains license information for activating Authentic Browser Enterprise Edition. If the three parameters `LicServer`, `LicCompany`, and `LicKey` are not present, then the Authentic Browser functionality will be limited to that of Community Edition.

3.3.4 Firefox Example 2: Sort a Table

This is an example HTML page with an embedded JavaScript. The sample requires the Authentic Browser Plug-in (XPI file) to be installed on your computer. Note that case-sensitivity might be an issue with some servers, so if there is a problem locating a file, check the casing of filenames and of commands in the code.

The code shows:

- How to access the browser plug-in. Modify the code to refer to your XPI file and the class identifier (MIME Type) of your browser plug-in version (trusted or untrusted).
- How to load a file into the browser plug-in. Modify the code to refer to your sample document.
- Implement buttons for simple cursor positioning.
- Implement more complex commands like the sorting of tables.
- How to use the `SelectionChanged` event.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Test Page For Authentic Browser Plug-in</title>

</head>

<!-- to disable the fast-back cache in Firefox, define an unload handler -->
<BODY id="bodyId" onunload="Unload()">
   <embed
       id="objPlugIn"
       type="application/x-authentic-scriptable-plugin"
       width="100%"
       height="60%"
       PLUGINSPAGE="http://your-server-including-path/
AuthenticFirefoxPlugin_trusted.xpi"
       EntryHelpersEnabled="TRUE"
       SaveButtonAutoEnable="TRUE" >
   </embed>
<TABLE>
<SCRIPT LANGUAGE="javascript">
var objCurrentRange = null;
var objPlugIn = document.getElementById('objPlugIn');

function BtnDocumentBegin() { objPlugIn.AuthenticView.DocumentBegin.Select(); }
function BtnDocumentEnd() { objPlugIn.AuthenticView.DocumentEnd.Select(); }
function BtnWholeDocument() { objPlugIn.AuthenticView.WholeDocument.Select(); }
function BtnSelectNextWord()
{ objPlugIn.AuthenticView.Selection.SelectNext(1).Select(); }

function BtnSortDepartmentOnClick()
{
       var objCursor = null;
       var objTableStart = null;
       var objBubble = null;
       var strField1 = "";
       var strField1 = "";
       var nColIndex = 0;
       var nRows = 0;

       objCursor = objPlugIn.AuthenticView.Selection;
       if (objCursor.IsInDynamicTable())
       {
```

```
                // calculate current column index
                nColIndex = 0;
                bContinue = true;
                while ( bContinue )
                {
                        try { objCursor.GotoPrevious(11); }
                        catch (err) { bContinue = false; nColIndex--; }
                        nColIndex++;
                }

                // GoTo begin of table
                objTableStart = objCursor.ExpandTo(9).CollapsToBegin().Clone();

                // count number of table rows
                nRows = 1;
                bContinue = true;
                while ( bContinue )
                {
                        try { objTableStart.GotoNext(10); }
                        catch (err) { bContinue = false; }
                        nRows++;
                }

                // bubble sort through table
                for ( i = 0; i < nRows - 1; i++) {
                        for( j = 0; j < nRows-i-1; j++) {
                                objBubble =
objCursor.ExpandTo(9).CollapsToBegin().Clone();
                                // Select correct column in jth table row

        objBubble.GotoNext(6).Goto(10,j,2).Goto(11,nColIndex,2).ExpandTo(6);
                        strField1 = objBubble.Text;
                                try
                                {
                                        strField2 =
objBubble.GotoNext(10).Goto(11,nColIndex,2).ExpandTo(6).Text;
                                }
                                catch ( err ) { continue; };
                        if(strField1 > strField2) {
                                        if(!objBubble.MoveRowUp()) {
                                                alert('Table row move is not
allowed!');
                                                return;
                                        }
                                }
                        }
                }
        }
}

function InitAuthenticPluginPage( )
{
        var serverstr='your-server/';
        var basedir='Authentic/';
        objPlugIn.SchemaLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.xsd';
        objPlugIn.XMLDataLoadObject.URL = 'http://' + serverstr + basedir +
```

```
'OrgChart.xml' ;
	objPlugIn.DesignDataLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.sps';
	objPlugIn.StartEditing();
}

function Unload()
{
}

function OnSelectionChanged()
{
	var CurrentSelection = null;
	CurrentSelection = objPlugIn.AuthenticView.Selection;
	SelTable_FirstTextPosition.innerHTML =
CurrentSelection.FirstTextPosition;
	SelTable_LastTextPosition.innerHTML = CurrentSelection.LastTextPosition;
	SelTable_FirstXMLData.innerHTML =
CurrentSelection.FirstXMLData.Parent.Name;
	SelTable_FirstXMLDataOffset.innerHTML =
CurrentSelection.FirstXMLDataOffset;
	SelTable_LastXMLData.innerHTML =
CurrentSelection.LastXMLData.Parent.Name;
	SelTable_LastXMLDataOffset.innerHTML =
CurrentSelection.LastXMLDataOffset;
}

objPlugIn.addEventListener("selectionchanged", OnSelectionChanged, false)
// event subscription if running on Firefox
objPlugIn.addEventListener("ControlInitialized", InitAuthenticPluginPage,
false);
</SCRIPT>
<TR>
<TD><Input Type="button" value="Goto Begin" id="B1"
onclick="BtnDocumentBegin()"></TD>
<TD><Input Type="button" value="Goto End" name="B2"
onclick="BtnDocumentEnd()"></TD>
<TD><Input Type="button" value="Whole Document" name="B3"
onclick="BtnWholeDocument()"></TD>
<TD><Input Type="button" value="Select Next Word" name="B4"
onclick="BtnSelectNextWord()"></TD>
</TR>
<TR>
<TD><Input Type="button" value="Sort Table by this Column" id="B6"
onclick="BtnSortDepartmentOnClick()"</TD>
</TR>
</TABLE>
<TABLE id=SelTable border=1>
<TR><TD id=SelTable_FirstTextPosition></TD><TD id=SelTable_LastTextPosition></
TD></TR>
<TR><TD id=SelTable_FirstXMLData></TD><TD id=SelTable_FirstXMLDataOffset></TD></
TR>
<TR><TD id=SelTable_LastXMLData></TD><TD id=SelTable_LastXMLDataOffset></TD></
TR>
<TR><TD id=SelTable_Text></TD></TR>
</TABLE>
</body>
</html>
```

3.4 Browser-Independent

In some projects, it may not be known what browser (Internet Explorer or Firefox) will be used on the client. In such cases, Authentic Browser versions for both Internet Explorer and Firefox can be stored on the server (that is, both the CAB file and XPI file can be stored on the server). In the HTML page you can insert a script to determine which browser has been used to open the HTML page, and accordingly cause the correct Authentic Browser Plug-in to be loaded.

Additionally, if Internet Explorer is the browser that is used on the client, then the correct `.CAB` file (for 32-bit or 64-bit Internet Explorer) must be selected for download from the server. A script can test for the X-bit version of Internet Explorer and select the correct `.CAB` file for download from teh server.

This section contains an example file, which does the following: determines the browser, loads the correct Authentic Browser version, and carries out a few functions.

For information about individual objects, see the respective Object descriptions in the Reference section.

Note:

- The Authentic Browser plug-in is supported for **Internet Explorer 5.5 or higher**, **Mozilla Firefox**
- For Firefox usage, ensure that the MIME type for XPI files has been added in the browser service of your server to the list of MIME types for the site you will use.

3.4.1 Browser-Independent Example

The HTML code below generates a page that has the following features:

- It checks what browser is installed on the client (Internet Explorer, Firefox) and installs an Authentic Browser version for the detected browser type.
- Furthermore, if the installed browser is Internet Explorer, then it checks whether the system is 32-bit or 64-bit, and then selects the correct .CAB file (for 32-bit or 64-bit Internet Explorer).
- The Authentic Browser window within the page has a width that is 100% that of the browser window and 60% of its height.
- Below the Authentic Browser window is a row of five buttons
- The **Start Editing** button loads the Authentic View of `OrgChart.xml`, which is in the root directory of your server
- The **Find** and **Replace** buttons pop up the Find and Replace dialogs respectively
- The **Save** button saves changes to a file called `SaveFile_OrgChart.xml` located in the root directory of the server
- The **Test** property button tests a simple property

When this HTML page is opened on the client, the user can start editing the XML file `OrgChart.xml` and save the edited file as `SaveFile_OrgChart.xml`.

You may wish to use this simple HTML page to test whether Authentic Browser functions properly. If you do so, be sure to use the IP Address and the correct path to the respective files in the URLs that locate the the `XPI` file, the `xsd`, `xml`, and `sps` files, and any other resource on the server. Note that case-sensitivity might be an issue with some servers, so if there is a problem locating a file, check the casing of filenames and of commands in the code. You can expand or modify this example to build more complex solutions using Authentic Browser.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Orgchart.sps Scriptable Plug-in Test - browser independent</title>
      <script type="text/javascript">
         <!--
         function BtnOnSave() { objPlugIn.Save();}

         function InitAuthenticPluginPage( )
         {

          var schema= document.getElementById('xsd');
          var instance=document.getElementById('xml');
          var design=document.getElementById('sps');
          objPlugIn.XMLDataLoadObject.URL =instance.innerHTML;
          objPlugIn.DesignDataLoadObject.URL = design.innerHTML;
          objPlugIn.SchemaLoadObject.URL= schema.innerHTML;
         // alert(schema.innerHTML+"  "+instance.innerHTML+" "
+design.innerHTML);

         /*
          var serverstr='your-server/';
          var basedir='Authentic/';
          objPlugIn.SchemaLoadObject.URL = 'http://' + serverstr + basedir +
```

```
'OrgChart.xsd';
         objPlugIn.XMLDataLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.xml' ;
         objPlugIn.DesignDataLoadObject.URL = 'http://' + serverstr + basedir +
'OrgChart.sps';
        */

         objPlugIn.StartEditing();


        }

        function Unload()
        {
        }
        //-->
      </script>
   <style type="text/css">@page { margin-left:0.60in; margin-right:0.60in;
margin-top:0.79in; margin-bottom:0.79in } @media       screen { br.altova-page-
break { display: none; } } @media print { br.altova-page-break { page-break-
before: always; } }
   </style>
   </head>

   <body id="bodyId" onunload="Unload()">
      <table border="1">
         <tbody>
            <tr><th><span>DesignLoadURL</span></th><td id="sps">http://your-
server/Authentic/Orgchart.sps</td></tr>
            <tr><th><span>SchemaLoadURL</span></th><td id="xsd">http://your-
server/Authentic/Orgchart.xsd</td></tr>
            <tr><th><span>XMLDataLoadURL</span></th><td id="xml">http://your-
server/Authentic/Orgchart.xml</td></tr>
            <tr><th><span>XMLDataSaveURL</span></th><td id="xmlsave">http://your-
server/Authentic/SaveFile_OrgChart.xml</td></tr>
         </tbody>
      </table>
      <center><h3><span>Authentic Platformindependent Plug-in Enterprise
Edition</span></h3></center>
      <span> </span>
      <center>
      <script language="JavaScript" type="text/javascript">
         // return true if the page loads in Firefox
         function isFirefoxOnWindows()
         {
          return ((navigator.userAgent.indexOf('Firef') != -1) &&
(navigator.userAgent.indexOf('Win') != -1));
         }

         // return true if the page loads in Internet Explorer
         function isIEOnWindows()
         {
          return ((navigator.userAgent.indexOf('MSIE') != -1) &&
(navigator.userAgent.indexOf('Win') != -1))
         }

         //return true if Browser is 64bit
```

```
        function is64bitBrowser()
        {
         return ((navigator.userAgent.indexOf('Win64') != -1)&&
(navigator.userAgent.indexOf('x64') != -1))
        }

        //return Codebase for 32 bit  or 64 bit
        function getCodeBase()
        {
          if ( is64bitBrowser() ){
           return('CodeBase="http://your-server/
AuthenticBrowserEdition_x64.CAB#Version=12,2,0,0" ')
          }
          else {
           return('CodeBase="http://your-server/
AuthenticBrowserEdition.CAB#Version=12,2,0,0" ')
          }
        }

        // Create the plugin object instance, according to the browser loading
the page
        // -Firefox uses EMBED tag for embeding plugins and supports PLUGINSPAGE
        // attribute to redirect to an installation file if the plugin is not
        // currently installed;
        // -IE uses <OBJECT> tag for embeding plugins and supports CODEBASE
attribute
        // to indicate a .cab file for the installation if the plugin is not
        // currently installed

        function createObject( codebase, clsid)
        {
         if ( isFirefoxOnWindows() )
         {
         document.write ( '<embed ' +
         'id="objPlugIn" ' +
         'type="application/x-authentic-scriptable-plugin" ' +
         'width="100%" ' +
         'height="60%" ' + 'PLUGINSPAGE="http://your-server/Authentic/
AuthenticFirefoxPlugin_trusted.xpi" ' +
         'SaveButtonAutoEnable="true" ' +
         'EntryHelpersEnabled="true" ' +
         'LicServer="your-server" ' +
         'LicCompany="Altova" ' +
         'LicKey="XXXXXXXXXX" ' +
         'XMLDataSaveUrl="http://your-server/Authentic/SaveFile_OrgChart.xml"> '
+
         '<\/embed>' );
         }
          else if ( isIEOnWindows() )
         {
         document.write ( '<OBJECT ' +
           'id="objPlugIn" ' +
           getCodeBase() +
           'Classid="clsid:B4628728-E3F0-44a2-BEC8-F838555AE780" ' +
           'width="100%" ' +
```

```
            'height="60%" ' +
            '>' +
            '<PARAM NAME="XMLDataSaveUrl" VALUE="http://your-server/Authentic/
SaveFile_OrgChart.xml"> ' +
            '<PARAM NAME="EntryHelpersEnabled" VALUE="TRUE"> ' +
            '<PARAM NAME="SaveButtonAutoEnable" VALUE="TRUE"> ' +
            '<PARAM NAME="LicServer" VALUE="your-server"> ' +
            '<PARAM NAME="LicCompany" VALUE="Altova"> ' +
            '<PARAM NAME="LicKey" VALUE="XXXXXXXXXX"> ' +
            '<\/OBJECT>');
         }
         }

         createObject();
         // after running createObject the plugin object exists. Initialize the
javascript variable to be used in the scripts
         var objPlugIn = document.getElementById('objPlugIn');
      </script>

         <br><br>
         <button onclick="objPlugIn.StartEditing()"><span>Start Editing</span></
button>
         <button onclick="objPlugIn.FindDialog()"><span>Find</span></button>
         <button onclick="objPlugIn.ReplaceDialog();"><span>Replace</span></
button>
         <button onclick="BtnOnSave()"><span>Save</span></button>
         <button onclick="alert ( objPlugIn.IsRowInsertEnabled );"><span>Test</
span><br></button>
         </center>

      <script language="javascript" type="text/javascript">
            // event subscription if running on Firefox
             if ( isFirefoxOnWindows() )
             {
             objPlugIn.addEventListener("ControlInitialized",
InitAuthenticPluginPage, false);
             }
      </script>

      <script event="ControlInitialized" for="objPlugIn" language="javascript"
type="text/javascript">
            // event subscription if running on Internet Explorer
             if ( isIEOnWindows() )
             {
             InitAuthenticPluginPage();
             //if ( isIEx64OnWindows() ) alert("IE x64");
             }
      </script>

   </body>
</html>
```

Note: The script above contains license information for activating Authentic Browser Enterprise Edition. If the three parameters `LicServer`, `LicCompany`, and `LicKey` are not present, then the Authentic Browser functionality will be limited to that of Community Edition.

4 Extension Packages for On-Demand Installation

If on-demand installation of the Authentic Browser plug-in is planned, the zipped Authentic Browser extension package/s (CAB, XPI, and/or CRX file/s) must be stored on the server. When the HTML page for Authentic Plug-in is accessed in the client browser for the first time, instructions in the HTML page cause the relevant extension package to be downloaded from the server to the client, unzipped, and installed on the client.

The required Authentic Browser extension package/s for on-demand installation can be downloaded from the Altova website. They must be stored on the server as a zipped (CAB, XPI, or CRX) file. Whether you store a CAB, XPI, or CRX file on the server depends upon which browser is used on the client to open the HTML page for Authentic Plug-in.

Internet Explorer v5.5 or higher (32-bit)	CAB file (for 32-bit IE)
Internet Explorer (64-bit)	CAB file (for 64-bit IE)
Firefox	XPI file

If a client may use multiple browsers or any browser (Internet Explorer or Firefox), then two or all three extension packages (CAB, XPI, and CRX files) should be stored on the server. If the client browser is not known, you could include a script in the HTML page for Authentic Plug-in to detect which browser is currently being used. The appropriate file (CAB for Internet Explorer; XPI for Firefox) will then be downloaded automatically from the server to the client. For information about this scenario, see HTML Page for Authentic Plug-in | Browser-Independent.

CAB files for Internet Explorer are available for 32-bit IE browsers and 64-bit IE browsers. Again, you may wish to store both types of CAB file (32-bit and 64-bit) on the server. The HTML page for Authentic Plug-in could have a script that determines whether the browser is a 32-bit or 64-bit version and then downloads the correct CAB file. How to create such a script is described in the section, HTML Page for Authentic Plug-in | Browser-Independent.

Downloading and storing the CAB/XPI/CRX file

The CAB file, XPI file, and/or CRX file is to be downloaded from the Altova Website and can be downloaded to any location on your server. If you are deploying the Enterprise edition of Authentic Browser, then the package **must be stored on the server for which the Enterprise license has been registered**.

All three types of installer file (CAB, XPI, CRX) are zipped file formats. **Do not unzip these files**. The file extraction and installation on the client takes place automatically when the client opens the HTML page for Authentic Plug-in for the first time. The location of the CAB/XPI/CRX file on the server is specified in the HTML page for Authentic Plug-in.

Chapter 5

Client Setup

Client Setup

The client machine is the machine on which the HTML Page for Authentic Plug-in is opened and on which the Altova Authentic XML page is edited. To achieve this functionality, the Authentic Browser plug-in must be installed as an add-on in the client browser that will be used to open the HTML Page for Authentic Plug-in.

The Authentic Browser plug-in can be installed in the client browser automatically from the server when the HTML Page for Authentic Plug-in is opened in the client browser. Alternatively, the Authentic Browser plug-in can be installed directly in the client browser, either manually or by using a centrally administered MSI-based push system.

This section describes how to set up the client machine for these functions. It describes the following:

- The Internet browser requirements for enabling Authentic Browser functionality.
- The different methods of installing the Authentic Browser plug-in as an add-on in supported browsers on the client machine. It also describes how to upgrade and de-install the Authentic Browser plug-in.
- IE9 Security Settings to allow the proper functioning of the plug-in.
- IE10 Security Settings to allow the proper functioning of the plug-in. Note that, for Authentic Browser plug-in to be allowed in Internet Explorer 10, the IE 10 mode should be set to Compatible Mode.

1 Browser Requirements

For a client machine to display an Altova Authentic XML page, it requires an Internet browser that allows plug-ins. Since a number of browser providers discontinued support for the NPAPI architecture, Auhentic Browser Edition is currently supported only on **Microsoft Internet Explorer 32-bit version 9 and newer**. If you wish to use another Internet browser, you will need to use an older version that supports plug-ins.

The list below provides information about support levels for Authentic Browser plug-in among the major Internet browsers.

- *Microsoft Internet Explorer 32-bit* version 9 and newer support Authentic Browser.
- *Microsoft Internet Explorer 64-bit* version 9 is the last 64-bit version that can be used. Versions newer than that are Microsoft Edge.
- *Microsoft Edge* does not support Authentic Browser since it never supported ActiveX controls as plug-ins.
- *Firefox version 26.0* is the last to offer NPAPI support, and versions up to 26.0 (latest preferable) can be used for Authentic Browser.
- *Google Chrome version 44* is the last to offer NPAPI support, and versions up to 44 (latest preferable) can be used for Authentic Browser.

Note: Internet Explorer **must** be installed because the Authentic View interface (which will be displayed within the browser window) is generated using Internet Explorer.

2 Authentic Browser Plug-in

This section describes the various ways in which Authentic Browser can be installed on a client machine, as well as how the plug-in can be upgraded and de-installed.

- Installation on Demand
- Manual Installation via Extension Packages
- Manual Installation via MSI
- Push Installation using MSI
- Automatic Updates
- De-installation, Disabling

Installation of multiple versions
There are several versions of Authentic Browser. For each supported language (English, German, Spanish, Japanese), separate trusted and untrusted versions are available for each supported browser (Microsoft Internet Explorer 32-bit and 64-bit, Mozilla Firefox).

One, some, or all of these versions of the Authentic Browser plug-in can be installed on a client machine. Each will be displayed as a separate plug-in in the browser's Add-on Manager.

For more information about versions, see Authentic Browser Versions

2.1 Installation on Demand

The best way of installing the Authentic Browser plug-in is to have it automatically installed when needed. This mechanism works as follows:

1. The extension packages (CAB, XPI, and/or CRX file/s) are stored on the server.
2. When the HTML page for Authentic Plug-in is opened in the client's browser, instructions in the HTML page download the relevant extension package and install the Authentic Browser plug-in in the client browser.

The instruction in the HTML page will be different for different browsers.

For Internet Explorer
Use the `CODEBASE` attribute of the `OBJECT` element to specify the URL of the CAB file. See the example for Internet Explorer for details.

For Mozilla Firefox
Use the `pluginspage` attribute of the `embed` element to specify the URL of the XPI file. See the example for Firefox for details.

Alternative method for Mozilla Firefox
Add a link that points to the extension package and, when clicked, will install the plug-in. For example: `<a href="plugin.crx">Click to install Authentic Browser extension</a>`.

See the section, Extension Packages for On-Demand Installation, for information about server-side setup of the extension packages.

For information about versions, see Authentic Browser Versions

2.2 Manual Installation via Extension Packages

In some browsers, the Authentic Browser plug-in can also be installed manually.

Internet Explorer
There is no way to manually install an Internet Explorer add-on contained in a CAB file. Use Manual Installation via MSI.

Firefox
Drag and drop the XPI file into the Firefox window. The Add-on Manager (**Tools | Add-ons | Extensions**) should display the Authentic Browser plug-in in its list of installed extensions.

For information about versions, see Authentic Browser Versions

2.3 Manual Installation via MSI

You can download an MSI installer (Microsoft Windows Installer) file for Authentic Browser from the Altova website. This file is an executable installer file (with a `.exe` extension) that installs the Authentic Browser plug-in in all the supported browsers (Microsoft Internet Explorer 32-bit and 64-bit, Mozilla Firefox) currently installed on the client machine.

Installation
Download the MSI installer file to the client machine and double-click it to start the installation. The plug-in will be installed (by default to the folder: `C:\Program Files (x86)\Altova \AuthenticBrowserPlugin\`...) and integrated in the currently installed supported browsers in one step. Selective installation for different browsers is possible via the Custom Installation dialog.

The *Add/Remove Programs* list on a client machine will show the installed browser edition.

MSI versions
There is an MSI installer file for each supported language (English, German, Spanish, Japanese), each trusted/untrusted type, and each 32-bit/64-bit browser-type. So, for example, there will be one MSI installer file for English Trusted 32-bit browsers, another for English Trusted 64-bit browsers, another for English Untrusted 32-bit browsers, and so on.

For more information about versions, see Authentic Browser Versions

2.4 Push Installation using MSI

Where installation of new software is managed by a central IT team and installers are pushed out within an administration framework, installation via a native MSI installer (Microsoft Windows Installer) is an advantage. This is especially true for Internet Explorer where installation of a plug-in requires administration rights.

You can download an MSI installer (Microsoft Windows Installer) file for Authentic Browser from the Altova website. This file is an executable installer file (with a `.exe` extension) that installs the Authentic Browser plug-in in all the supported browsers currently installed on the client machine. Currently supported browsers are: Microsoft Internet Explorer 32-bit and 64-bit, Mozilla Firefox.

Installation
Download the MSI installer file to the administration server and include it in your administrative procedures for installing to client machines. The plug-in will be installed (by default to the folder: `C:\Program Files (x86)\Altova\AuthenticBrowserPlugin\...`) and integrated in the currently installed supported browsers in one step. Selective installation for different browsers is possible via the Custom Installation dialog.

For silent installation, use the command line input parameter `/q`.

The *Add/Remove Programs* list on a client machine will show the installed browser edition.

MSI versions
There is an MSI installer file for each supported language (English, German, Spanish, Japanese), each trusted/untrusted type, and each 32-bit/64-bit browser-type. So, for example, there will be one MSI installer file for English Trusted 32-bit browsers, another for English Trusted 64-bit browsers, another for English Untrusted 32-bit browsers, and so on.

For more information on versions, see Authentic Browser Versions

2.5 Automatic Updates

The availability of Authentic Browser upgrades and the installation of these updates varies according to browser.

Internet Explorer
The `codebase` attribute value of the `object` element in the HTML Page for Authentic Plug-in can be suffixed with `#Version=...` to indicate the version number of the plug-in available on the server. If the version of the plug-in installed on the user's machine is less than that version, the user will be asked if the plug-in should be upgraded.

Firefox
The standard mechanism for updating Firefox extensions is as follows. The `install.rdf` file in the Firefox extension contains a link to an `update.rdf` file on a download server. The `install.rdf` file specifies the version installed on the user's machine. The `update.rdf` file lists the versions and extension files available for download on the server. If newer versions are available for download, the user will be informed.

This is currently not supported.

2.6 De-installation, Disabling

If installation has been via MSI, then the *Add/Remove Programs* list on a client machine will show the installed browser edition. If you uninstall the product via the *Add/Remove Programs* list, the plug-in will be removed from the browser and from the MSI list.

When a plug-in is installed in a browser, it is displayed as enabled in the browser's Add-on Manager. If you remove the plug-in from the browser's Add-on Manager, the plug-in will be disabled but stays on disk.

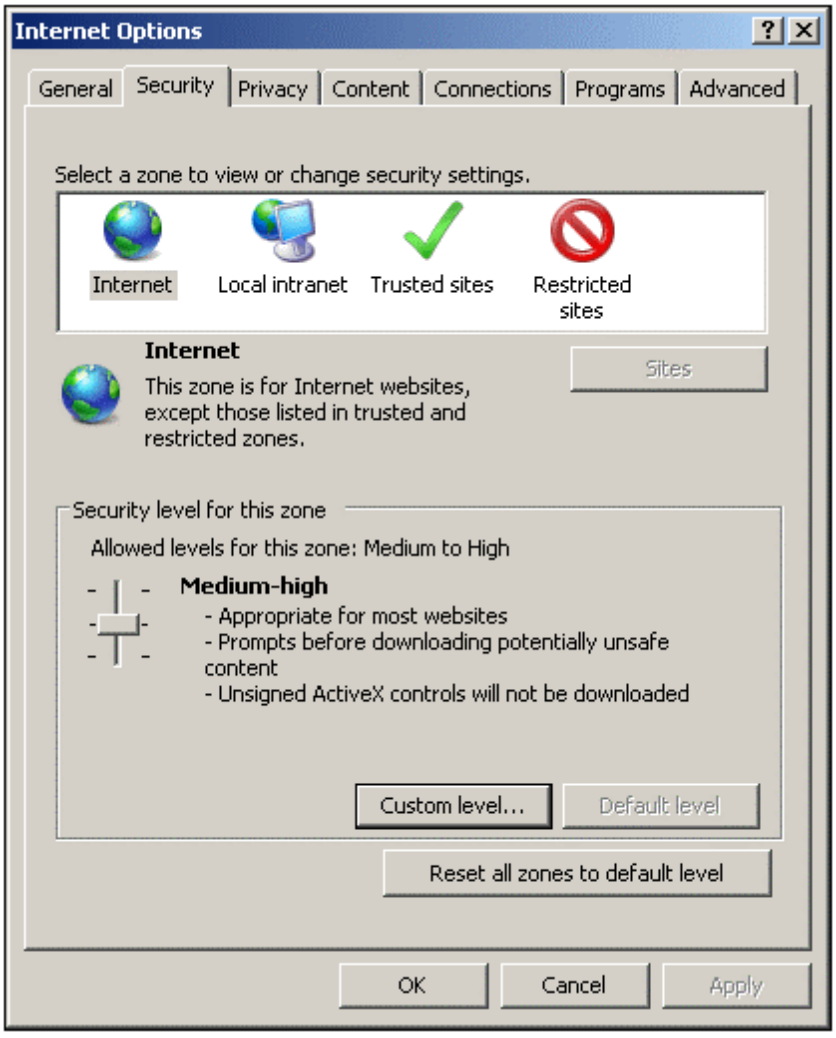
3 IE9 Security Settings

If the Untrusted version of Authentic Browser is being used in Internet Explorer 9, you might get the following error message:

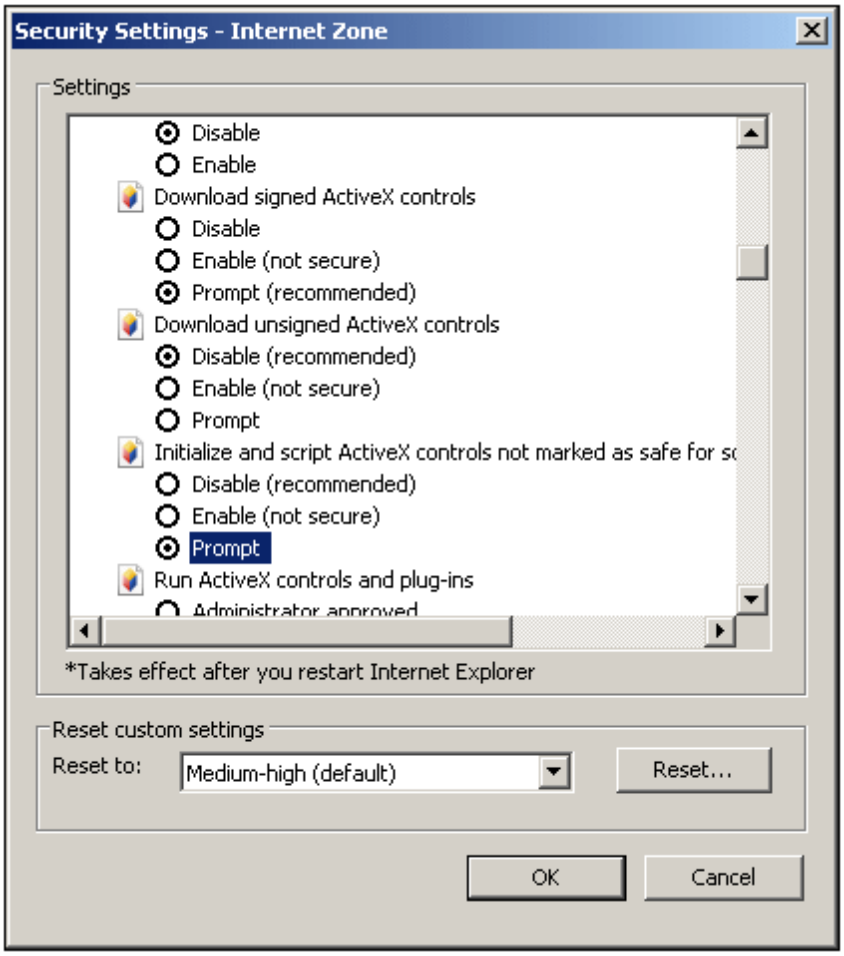
Internet Explorer blocked an ActiveX control, so this page might not display correctly.

To enable ActiveX controls, you must set the relevant security option in the Internet Options of Internet Explorer. Do this as follows:

1. Open the Internet Options dialog (*screenshot below*) of Internet Explorer by clicking the menu command **Tools | Internet Options**.



2. Select the Security tab (*see screenshot above*), then the zone for which you wish to make the setting (Internet or Local intranet).
3. Next, click the **Custom Level** button. This pops up the Security Settings dialog (*screenshot below*).



4. Scroll down to the *ActiveX controls and plug-ins* section, and within this to the setting *Initialize and script ActiveX controls not marked as safe for scripting*. Select the *Prompt* option (*see screenshot above*).
5. Restart Internet Explorer for the new setting to take effect.

From now onwards, each time an HTML Page for Authentic Plug-in attempts to load an ActiveX file, Internet Explorer will prompt you about whether you wish to load the control or not.

4 IE10 Security Settings

Internet Explorer 10 (IE10) has two browser modes:

- Metro-style, which runs Enhanced Protection Mode security, and
- Desktop, which runs Compatible Mode security.

To run Authentic Browser plug-in, you must set IE10 to Compatible mode (**Tool| Options | Security**).

Chapter 6

User Reference

User Reference

This section contains a listing and description of Authentic Browser mechanisms, objects, and enumerations.

1 Mechanisms

This section contains a description of some commonly used Authentic Browser mechanisms.

1.1 Events: Connection Point (IE-Specific)

Authentic Browser provides various connection point events (see also Events: Reference), for which you can provide event handlers in the SCRIPT blocks on your HTML page.

Note: The description in this section **applies to Internet Explorer only**.

The following samples show SCRIPT blocks for the ControlInitialized and SelectionChanged events:

ControlInitialized
This connection point event is triggered immediately after the control is created and initialized. Additional startup scripting for the control can be handled inside the ControlInitialized event handler.

```
<SCRIPT LANGUAGE=javascript FOR=objPlugIn EVENT=controlinitialized>
   // add your code here
</SCRIPT>
```

SelectionChanged
The SelectionChanged event is triggered each time the current selection in the view changes. Use a SCRIPT block to execute your event-specific code.

```
<SCRIPT LANGUAGE=javascript FOR=objPlugIn EVENT=selectionchanged>
   // add your code here
</SCRIPT>
```

Please note that the Authentic.event object is not populated when this event occurs. If event handlers are registered in your script, the Authentic.event object properties contain values from the last event to have occurred.

The Authentic.CurrentSelection object now contains valid information.

Loading SPS, XSD, and XML files without user intervention
If you wish to load the .sps, .xsd and .xml files without user intervention when loading the HTML page, it is the preferred method to write an event handler that handles the ControlInitialized event. Alternatively a property bag can be used as in the second example:

Recommended:

```
<SCRIPT LANGUAGE="javascript" FOR=objPlugIn EVENT="ControlInitialized">
objPlugIn.SchemaLoadObject.URL = "http://yourserver/OrgChart.xsd"
objPlugIn.XMLDataLoadObject.URL = "http://yourserver/OrgChart.xml"
objPlugIn.DesignDataLoadObject.URL = "http://yourserver/OrgChart.sps"
objPlugIn.StartEditing()
</SCRIPT>
```

or

```
<OBJECT id=objPlugIn style="WIDTH:600px; HEIGHT:500px"
codeBase="http://yourserver/cabfiles/
AuthenticBrowserEdition.CAB#Version=12,3,0,0"
classid=clsid:B4628728-E3F0-44a2-BEC8-F838555AE780>
<PARAM NAME="XMLDataURL" VALUE="http://yourserver/OrgChart.xml">
<PARAM NAME="SPSDataURL" VALUE="http://yourserver/OrgChart.sps">
<PARAM NAME="SchemaDataURL" VALUE="http://yourserver/OrgChart.xsd">
</OBJECT>
```

It is not recommended to load these files in an event handler that handles the "body" elements "onload" event as the Authentic Browser PlugIn control may be initialized after the "onload" event is fired. If this is the case the PlugIn's methods and properties will not be available, and the files will not load. Also see The OBJECT Element for details.

Not recommended:

```
<SCRIPT LANGUAGE="javascript">
 function load () {

objPlugIn.SchemaLoadObject.URL = "http://yourserver/OrgChart.xsd"
objPlugIn.XMLDataLoadObject.URL = "http://yourserver/OrgChart.xml"
objPlugIn.DesignDataLoadObject.URL = "http://yourserver/OrgChart.sps"
objPlugIn.StartEditing()
}
</SCRIPT>

<body onload = "load files">
```

1.2 Events: Adding Event Listeners (Firefox-specific)

For each connection point event Authentic Browser provides (see also Events: Reference) you can write event handlers in the SCRIPT blocks on your HTML page by using the addEventListener method.

Note: The description in this section **applies to Firefox only**.

Here is an example on how to provide an event handler for the SelectionChanged event in Firefox:

SelectionChanged
The SelectionChanged event is triggered each time the current selection in the view changes. Use a SCRIPT block to execute your event-specific code.

```
<SCRIPT LANGUAGE="javascript">
var selCount = 0;
function OnSelectionChanged()
{
    selCount = selCount + 1;
    selectionCounter.value = "SelectionCount = " + selCount;
}
var objPlugIn = document.getElementById('objPlugIn');
objPlugIn.addEventListener("selectionchanged", OnSelectionChanged, false)
</SCRIPT>
```

Please note that the Authentic.event object is not populated when this event occurs. If event handlers are registered in your script, the Authentic.event object properties contain values from the last event to have occurred.

The Authentic.CurrentSelection object now contains valid information.

1.3 Events: Toolbar Button

Each toolbar button has default behavior, which may need to be changed. The `AuthenticCommand` event allows you to add additional tasks to, or entirely redefine, the default behavior of toolbar buttons. Scripts can use the `AuthenticCommand` event to receive notification each time the user clicks a toobar icon.

Please note that not every command (from the Authentic.UICommands collection) has its own associated event. To find out which icon the user clicked, the script has to check the AuthenticEvent.srcElement property, which contains a reference to a corresponding AuthenticCommand object.

Example

```
// event handler for OnDocEditCommand
<SCRIPT LANGUAGE=javascript FOR=objPlugIn EVENT=doceditcommand>
 // we are interested in the k_CommandSave button
 if(objPlugIn.event.srcElement.CommandID == 1)
 {
  // instead of the standard HTTP PUT we want to use
  // a HTTP POST
  objPlugIn.SavePOST();

  // no standard execution follows
  objPlugIn.event.cancelBubble = true;
 }
</SCRIPT>
```

Reference
For available commands, see AuthenticToolbarButton.CommandID.

1.4 Events: Reference

List of connection point events

Name	since TypeLib	Description
SelectionChanged	1.0	Selection in the editor has changed. The Authentic.CurrentSelection object now contains valid information. The properties of the Authentic.event object are not set.
ControlInitialized	1.2	The control is now fully loaded and initialized. The properties of the Authentic.event object are not set.
dragover	1.8	A drag-over operation is occuring.
drop	1.8	A drop operation has happened.
keydown	1.8	A key has been pressed but not yet released.
keyup	1.8	A key on the keyboard has been released. This is usually the event to react on keystrokes.
keypressed	1.8	Raised on any keyboard input.
mousemove	1.8	The mouse pointer has been moved.
buttonup	1.8	One of the mouse buttons has been released.
buttondown	1.8	One of the mouse buttons has been pushed.
contextmenu	1.8	Sent after receiving WM_CONTEXTMENU.
editpaste	1.8	Called before a paste operation takes place.
editcut	1.8	Called before a cut operation takes place.
editcopy	1.8	Called before a copy operation takes place.
editclear	1.8	Called before a clear operation takes place.
doceditcommand	1.8	Raised on executing an Authentic command and to implement a custom command handler. See also Events: Toolbarbuttons. The properties of the Authentic.event object are not set.
buttondoubleclick	1.8	One of the mouse buttons has been double-clicked.

If no exception is mentioned in the description the properties of the Authentic.event object are set on calling the event handler.

Note: These event-handlers are synchronous, that is, they are called immediately when the event occurs.

1.5 Accessing and Modifying Document Content

To access and modify document content, you can use the `AuthenticRange`, `AuthenticView`, and `Authentic` objects (and their properties and methods).

Where there is a functionality overlap between the `AuthenticRange` and `AuthenticView` interface on the one hand and the interface provided by the `Authentic` object on the other, the `AuthenticRange` and `AuthenticView` interface is the preferred interface. The overlapping functionality of the `Authentic` object is being phased out and will be obsolete in a future version.

1.6 Editing Operations

When XML data is displayed in Authentic View, it is possible to manipulate individual elements using the standard editing operations of cut, copy, and paste. However, not all the XML data elements may be edited. It is therefore necessary to first test whether editing is possible. The mechanism used is as follows: First, check whether the particular editing operation is enabled; if it is, then call the method to perform that editing operation. The only method that does not have a test is the method `EditSelectAll`, which automatically selects all elements displayed in the document.

The following is a list of properties and methods that perform editing operations. Each property returns a boolean value. The methods have no parameter.

IsEditUndoEnabled	Authentic.EditUndo	Undo an editing operation
IsEditRedoEnabled	Authentic.EditRedo	Redo an editing operation
IsEditCopyEnabled	Authentic.EditCopy	Copy the selected text to the clipboard
IsEditCutEnabled	Authentic.EditCut	Cut the selected text to the clipboard
IsEditPasteEnabled	Authentic.EditPaste	Paste clipboard text to current cursor position
IsEditClearEnabled	Authentic.EditClear	Clear the selected text from the XML document

1.7 Find and Replace

The Authentic.FindDialog method, opens a Find dialog box, in which the user can enter a search term. The Authentic.FindNext method allows the next instance of the same item to be found. The `FindNext` method can be tested using the boolean property `IsFindNextEnabled`. A variation of the `FindDialog` method is the Authentic.ReplaceDialog method. This operation finds a specific item, and is used to replace it with a specific value entered by the user in the Replace dialog box. If the `FindNext` method is then called, the next item is found and replaced.

1.8 Row Operations

In Authentic View, a repeating element structure may be created as a dynamic table, in which each row represents an instance of the repeated element. Once a dynamic table is created, the Authentic View user can manipulate the rows and their data. These row operations can be carried out with the use of a script.

If an external script is to perform row operations then two steps must occur:

- The first step checks whether the row that the cursor is in uses a property. A property, such as `IsRowInsertEnabled`, is used and returns a True or False value.
- If the return value is True, then the required row method can be called.

The following is a list of properties and methods that perform row operations. Each property returns a boolean, and the methods have no parameter.

IsRowInsertEnabled	Authentic.RowInsert	Row insertion operation.
IsRowAppendEnabled	Authentic.RowAppend	Append row operation.
IsRowDeleteEnabled	Authentic.RowDelete	Delete row operation.
IsRowMoveUpEnabled	Authentic.RowMoveUp	Move the XML data up one row.
IsRowMoveDownEnabled	Authentic.RowMoveDown	Move the XML data down one row.
IsRowDuplicateEnabled	Authentic.RowDuplicate	Duplicate the current row.

1.9 Shortcut Keys

The following shortcut keys are valid if the Authentic Browser has the input focus:

CTRL + P	Print document
CTRL + Z	Undo
CTRL + Y	Redo
CTRL + X	Cut
CTRL + C	Copy
CTRL + V	Paste
CTRL + A	Select all
CTRL + F	Open Find dialog box
CTRL + H	Open Find-Replace dialog box

1.10 Text State Buttons

The Authentic View toolbar of Authentic Browser provides support for text state icons. These are icons that insert elements having pre-defined text-formatting properties. To be able to use this function, the element that is to be created as a text state icon must be:

- declared as a global template in the schema, and
- the SPS must contains the necessary definitions (see the StyleVision documentation)

The Plug-in needs to have the path to the `.bmp` that will be used as the text state icon or custom button in the toolbar. This URL to the icon's image file is provided as the value of the Authentic.TextStateBmpURL.

1.11 Entry Helpers

The Authentic Browser enables you to provides entry helper windows as in XMLSpy. This provides the Authentic View user ready access to the elements, attributes, and entities allowed at any location in the document. The entry helpers are disabled by default and do not appear unless they are explicity activated. To enable entry helpers, use the following PROPERTYBAG parameters in the OBJECT tag (Internet Explorer) or as an attribute of the EMBED element (Firefox):

EntryHelpersEnabled	TRUE / FALSE
EntryHelperSize	Size in pixels
EntryHelperAlignment	Takes SPYAuthenticToolbarAllignment values
EntryHelperWindows	Takes SPYAuthenticEntryHelperWindows values. Any combination is allowed (bit-check)

Instead of using the parameters in the OBJECT or EMBED tags you can also use properties and methods in the API:

Properties

- Authentic.EntryHelpersEnabled
- Authentic.EntryHelperAlignment
- Authentic.EntryHelperSize
- Authentic.EntryHelperWindows

Method

- Authentic.RedrawEntryHelpers

1.12 Packages

Authentic Browser Plug-in supports packages that extend program module functionality.

Packages are stored on the server and are installed locally (on the client) on demand via the Authentic Browser GUI. Once a package is installed locally, it is activated on every startup until the user removes it.

Currently, Authentic Browser supports spellchecker packages.

This section is organized into two parts:

- Working with packages, which describes how the packages mechanism works
- Spellchecker packages, which explains how spellchecker packages are to be stored on the server and how they are installed and used

1.12.1 Working with Packages

Packages are stored on the server and are installed locally (on the client), on demand via the Authentic Browser GUI. Once a package is installed locally, it is activated on every startup until the user removes it.

Package management

Package management is an Authentic Browser functionality that gives the Authentic Browser user control over available packages. This functionality is available via the Package Management dialog.

To enable package management functionality, you must use the `LoaderSettingsFileURL` parameter in the HTML Page for Authentic Plug-in. This parameter specifies the URL of the LoaderSettings file for package management (which can reside at any accessible location):

- In an HTML page for Internet Explorer, the `LoaderSettingsFileURL` parameter is used as a child `PARAM` element of the `OBJECT` element:

```
<OBJECT>
    ...
    <PARAM NAME="LoaderSettingsFileURL" VALUE="http://www.server.com/
AuthenticFiles/XMLSpyPlugInLoaderSettings.xml"/>
</OBJECT>
```

- In an HTML page for Firefox, the `LoaderSettingsFileURL` parameter is used as an attribute of the `EMBED` element:.

```
<EMBED
    ...
    LoaderSettingsFileURL="http://www.server.com/AuthenticFiles/
XMLSpyPlugInLoaderSettings.xml"/>
```

The LoaderSettings file contains the definitions of the packages, along with the URLs of the respective packages. Given below is the listing of a LoaderSettings file.

Example of a LoaderSettings XML file

The document element of the LoaderSettings XML file is `loadersettings` (*see listing below*). This element can contain multiple child `package` and/or multiple `zippackage` elements.

- The `package` element is used to reference the older `.pck` spelling packages and is used by Authentic Browser versions previous to version 2011r3. These Authentic Browser versions ignore the `zippackage` element.
- The `zippackage` element is used to reference the `.zip` spelling packages and is used by Authentic Browser versions from version 2011r3 onwards. These Authentic Browser versions ignore the `package` element.

For more details about the spelling packages, see the section, Spellchecker Packages.

```
<?xml version="1.0" encoding="UTF-8"?>
<loadersettings>
```

```
   <zippackage mode="user_demand"  category="spelling">
      <packageurl>Portuguese (Brazilian).zip</packageurl>
      <description>Portuguese (BR) language pack.</description>
   </zippackage>

   <zippackage mode="user_demand"  category="spelling">
      <packageurl>Portuguese (Brazilian).zip</packageurl>
      <description>Portuguese (BR) language pack.</description>
   </zippackage>

   <package mode="user_demand" id="SentrySpellChecker_EAM_only"
      category="spelling" version="1">
      <packageurl>PlugIn/SentrySpellChecker_EAM_only.pck</packageurl>
      <description>Sentry Spellchecker (EN-US)</description>
   </package>

   <package mode="user_demand" id="SentrySpellChecker_EALL"
      category="spelling" version="1">
      <packageurl>PlugIn/SentrySpellChecker_EALL.pck</packageurl>
      <description>Sentry SpellChecker EN with Legal and Medical.</description>
   </package>

</loadersettings>
```

The following should be noted:

- The location of a package must be specified as content of the **packageurl** child of the `package` or `zippackage` element. The location path can be absolute, or be relative to the HTML file that calls the LoaderSettings file. The package can be located anywhere on the server.

- Removing a package from the XML file (by removing a `package` element) causes the package to not be available for installation on the client.

- The **mode** attribute of the `package` element specifies the level of user influence over the decision to install the package. The following values may be used:

  1. **user**: Causes the user to be asked at every Authentic Browser startup whether the package should be installed.
  2. **user_demand**: Installs the package when this is specifically asked for by the user. The user specifically requests this by clicking either the toolbar button to spellcheck or to open the Package Management dialog.
  3. **force**: Causes the package to be installed without notifying the user.

- The content of the `description` child of `package` can be edited will be used as descriptive text in the package management dialog.

One-time installation of packages

If Authentic Browser is updated to a newer version, there is no need to reinstall a package that was previously installed on the client. The already installed package will be used by the new version. Authentic Browser on the client PC reads the LoaderSettings file on the server, retrieves

package information from this file, and checks whether the package is locally installed.

Note: Spelling packages supported in version 2011r3 and later (`.zip` packages) use a different spellchecker than the spelling packages of previous versions (`.pck` packages). Authentic Browser versions from version 2011r3 onwards use either the `.zip` spelling packages. Older versions of Authentic Browser use the `.pck` spelling packages.

1.12.2 Spellchecker Packages

From version 2011r3, Authentic Browser uses Hunspell dictionaries.

Installation location

On a client machine, these dictionaries are located in a `Lexicons` folder at the following location:

On Windows 7/8: `C:\ProgramData\Altova\SpellChecker\Lexicons`
On Windows XP: `C:\Documents and Settings\All Users\Application Data\Altova\SharedBetweenVersions\SpellChecker\Lexicons`

If any Altova product has been installed on the client machine, then a set of built-in dictionaries will have been installed in the `Lexicons` folder at the location listed above. 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).

Additional dictionaries can be downloaded from the Altova website. Hunspell dictionaries can also be downloaded from other websites, for example, from http://wiki.services.openoffice.org/wiki/Dictionaries or http://extensions.services.openoffice.org/en/dictionaries. Also note that since Hunspell dictionaries are based on Myspell dictionaries, Myspell dictionaries can also be installed in the `Lexicons` folder. OpenOffice zips dictionary files as `.oxt` files. So one can change the extension to `.zip` and then unzip the necessary `.aff` and `.dic` files. Users must ensure that they are in conformance with the license agreement governing the use of any dictionary files they install.

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.

Dictionary files and dictionary folder structure on the client

Each installed dictionary on the client machine consists of two Hunspell dictionary files that work together: a `.aff` file and `.dic` file.

All language dictionaries are installed in the `Lexicons` folder at the location listed above. Within the `Lexicons` folder, different language dictionaries are each stored in a different folder: `<language name>\<dictionary files>`. These language folder names will be the dictionary names that are displayed in Authentic Browser. The structure of the `Lexicons` folder and the corresponding dictionary names as they would appear in Authentic Browser are as listed below.

```
Lexicons
       |
       |-- English (British)        Dictionary name: English (British)
       |          |
       |          |-- .aff file
       |          |   .dic file
```



```
|
|-- English (US)              Dictionary name: English (US)
|          |
|          |-- .aff file
|          |   .dic file
|
|-- German                    Dictionary name: German
|          |
|          |-- .aff file
|          |   .dic file
```

Dictionary files on the server

A language dictionary must be saved on the server as a ZIP file. When the ZIP file is downloaded to the client and installed, a language folder is created in the `Lexicons` folder (see Installation Location above) and the unzipped `.aff` and `.dic` files are placed in this folder. For example, the file `English (British).zip` is creates the folder: `<path>\Lexicons\English (British)` and installs the `.aff` and `.dic` files in this folder.

Also, multiple ZIP files can be zipped in a single ZIP file and this single file can be placed on the server. When this ZIP file is downloaded to the client and installed there, the name of the innermost ZIP files are used as the names of the folders in the Lexicons folder. For example, the files `English (British).zip` and `English (US).zip` can be zipped into a file called `English.zip`, which is placed on the server. On installation, two folders will be created in the `Lexicons` folder: `<path>\Lexicons\English (British)` and `<path>\Lexicons\English (US)`.

Dictionary installation mechanism

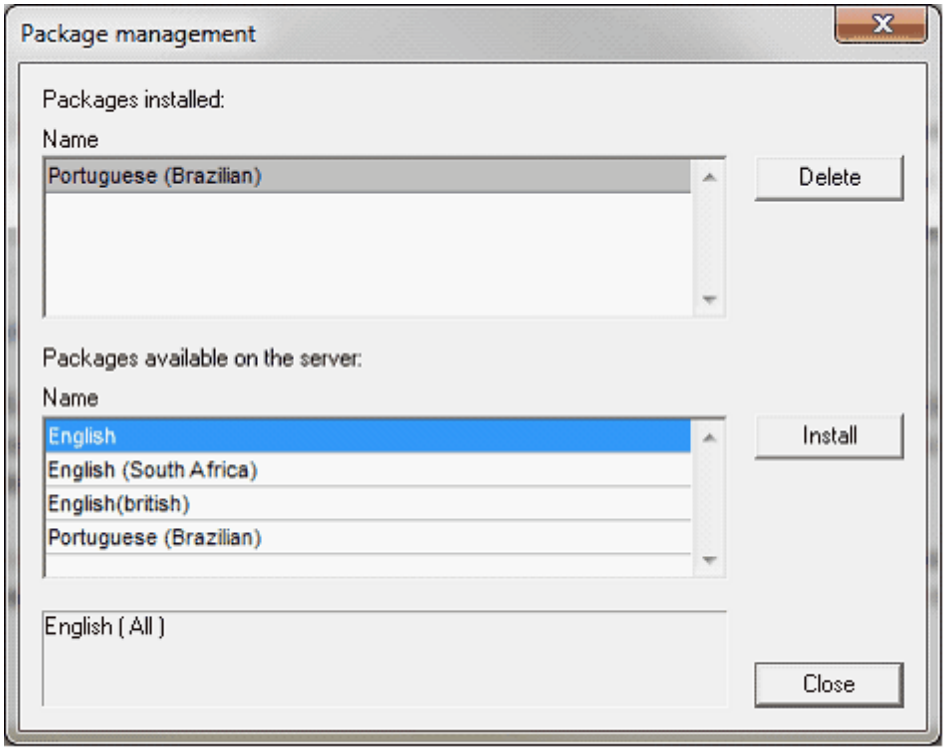
When an HTML page for Authentic Plug-in is opened, the dictionary packages can be installed on the client, either automatically or when the user requests it. The installation mechanism works as follows:

1. The HTML page that is opened in Authentic Browser references the LoaderSettings file.
2. The LoaderSettings file references the spelling package on the server. The LoaderSettings file also specifies, for each spelling package, whether that package should be installed automatically immediately after the HTML page is loaded or whether the user should be asked about installation.
3. The spelling package is installed from the zipped file on the server to the client

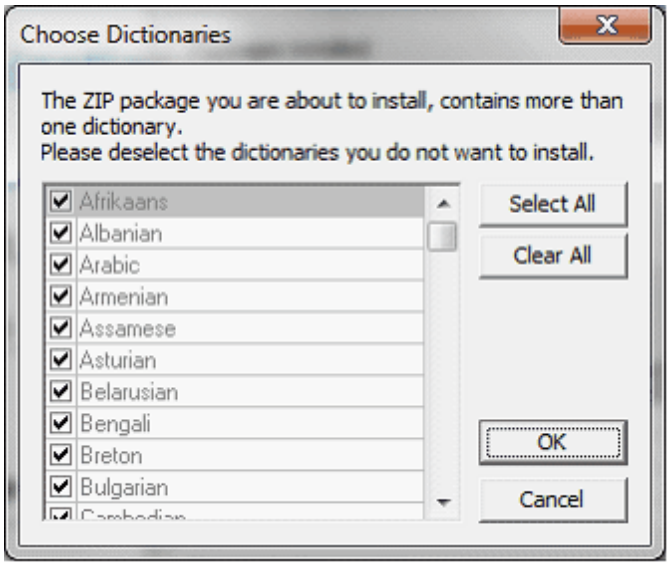
Dictionary installation

Spellchecker packages are stored on the server as ZIP (`.zip`) files and can be installed locally via the Authentic Browser GUI. To install a spellchecker package, click the Package Management icon in the Authentic Browser GUI. This pops up the Package Management dialog (*screenshot below*). The Package Management dialog also pops up if no dictionary is installed and the Spellchecker icon is clicked.

In the Package Management dialog (*screenshot below*), all the packages available on the server are listed in the lower pane. The names displayed are the names of the ZIP files on the server. Select the files you wish to install and then click **Install**. Installed packages can be deleted by selecting them in the upper pane and clicking **Delete**.



If a ZIP file selected for installation contains multiple ZIP packages, then the Choose Packages dialog (*screenshot below*) pops up. In this dialog, you can check the packages you wish to install, or uncheck the packages you wish not to install. Click **OK** to install the checked packages. This mechanism enables you to store each language dictionary separately on the server or within a single ZIP file.



If the installation of a package will overwrite an existing package, a warning message is displayed. The Authentic Browser user can then either go ahead with the overwriting or cancel.

1.13 Using XMLData

XMLData gives you access to the elements of the currently displayed XML file. It enables you to perform all necessary modifications to the elements of the XML structure. The main functionality of XMLData is:

1. Access to the names and values of all kinds of elements (e.g. elements, attributes)
2. Creation of new elements of all kinds.
3. Insertion and appending of new elements.
4. Erasing of existing child elements.

If you are already familiar with the XMLData interface because you used it in the XMLSpy API, please note that special considerations must be made if new elements are created and inserted into the XML file, or existing elements are renamed. Please take a look at "Creation and Insertion of new XMLData objects" and "Name and value of elements" below.

Structure of XMLData
Before you can use the XMLData interface, you have to know how an existing XML file is mapped into a XMLData structure. One major thing you must be aware of is, that XMLData has no seperate branch of objects for attributes.

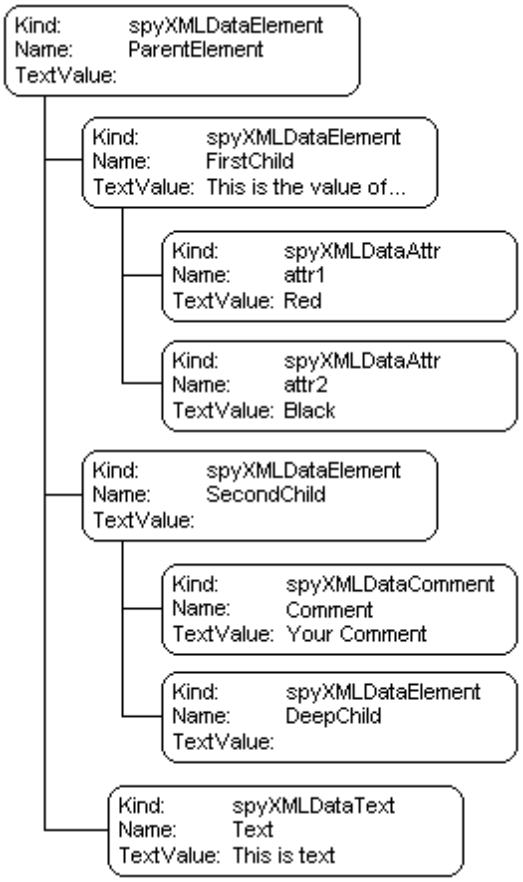
The attributes of an element are also children of the element. The XMLData.Kind property, gives you the opportunity to distinguish between the different types of children of an element.

Example:

This XML code,

```
<ParentElement>
 <FirstChild attr1="Red" attr2="Black">
  This is the value of FirstChild
 </FirstChild>
 <SecondChild>
  <!--Your Comment-->
  </DeepChild>
 </SecondChild>
 This is Text
</ParentElement>
```

is mapped to the following XMLData object structure:



The parent of all XML elements inside of a file is the property Authentic.XMLRoot. Use this XMLData object to get references to all other XML elements in the structure.

Name and value of elements
To get and to modify the name and value of all types of XML elements use the XMLData.Name and XMLData.TextValue properties. It is possible that several kinds of XMLData objects and empty elements do not have an associated text value.

It is not recommended to change the name of an existing XML element in the Authentic View. The name has an important impact how the Authentic View displays the content of the element. Please refer to the StyleVision documentation for detailed information.

Creation and insertion of new XMLData objects
The creation of a new XML language entity requires the following steps:

1. Create the new XMLData object:
   Use the Authentic.CreateChild method to create a new XMLData object. Set name and value before you insert the new XML entity (see point 3).

2. Find the correct location for the new XMLData object:
   To insert a new XMLData object you have to get a reference to the parent first. If the new child is to become the last child of the parent, use the XMLData.AppendChild method to insert the XMLData object.

If the new child should be located elsewhere in the sequence of child objects, use the XMLData.GetFirstChild and XMLData.GetNextChild to move the iterator to the child before which the new child should be inserted.

3. Insert the new child with XMLData.InsertChild. The new child will be inserted immediately before the current child. In the Authentic View it can happen that together with the created element additional child nodes will be added to the XML file. This depends on the node settings you can modify using the StyleVision.

The following example adds a third child between <FirstChild> and the <SecondChild> element:

```
Dim objParent
Dim objChild
Dim objNewChild

Set objNewChild = objPlugIn.CreateChild(spyXMLDataElement)
objNewChild.Name = "OneAndAHalf"

'objParent is set to <ParentElement>
'GetFirstChild(-1) gets all children of the parent element
'and move to <SecondChild>
Set objChild = objParent.GetFirstChild(-1)
Set objChild = objParent.GetNextChild

objParent.InsertChild objNewChild
Set objNewChild = Nothing
```

Child elements should be inserted in a special order. Please avoid to insert attributes into the sequence after any other child elements. That means attributes must not have preceding elements of any other type and any other element must not have a succeeding element of type attribute.

Authentic View requires a special handling of displaying the text value of an XML element except for attributes. Any text (or content) must be part of an extra child node of type text. You can create such an element using Authentic.CreateChild with the parameter value 6. Instead of setting the text value of the element directly please set the text value of the child node.

Copying of existing XMLData objects

If you want to insert existing XMLData objects at a different place in the same file, you cannot use the XMLData.InsertChild and XMLData.AppendChild methods. These methods only work for new XMLData objects.

Instead of using InsertChild or AppendChild, you have to copy the object hierarchy manually. The following function written in JavaScript is an example for recursively copying XMLData:

```
// this function returns a complete copy of the XMLData object
function GetCopy(objXMLData)
{
 var objNew;
 objNew = objPlugIn.CreateChild(objXMLData.Kind);
```

```
  objNew.Name = objXMLData.Name;
  objNew.TextValue = objXMLData.TextValue;

  if(objXMLData.HasChildren) {
   var objChild;
   objChild = objXMLData.GetFirstChild(-1);

   while(objChild){
    try {
     objNew.AppendChild(GetCopy(objChild));
     objChild = objXMLData.GetNextChild();
    }
    catch(e) {
     objChild = null;
    }
   }
  }

  return objNew;
 }
```

Erasing of XMLData objects
XMLData provides two methods for the deletion of child objects, XMLData.EraseAllChildren and XMLData.EraseCurrentChild.

To erase XMLData objects you need access to the parent of the elements you want to remove. Use XMLData.GetFirstChild" and XMLData.GetNextChild to get a reference to the parent XMLData object.

See the method descriptions of XMLData.EraseAllChildren and XMLData.EraseCurrentChild for examples how to erase XML elements.

1.14 DOM and XMLData

The XMLData interface gives you full access to the XML structure of the current document with less methods than DOM and is also 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 plug-in 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;
  }
```

1.15 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.

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.

2 Objects

This section contains a listing and description of the various Authentic Browser objects.

2.1 Authentic

See also

Methods

StartEditing
LoadXML
Reset

Save
SavePOST
SaveXML

ValidateDocument

EditClear
EditCopy
EditCut
EditPaste
EditRedo
EditSelectAll
EditUndo

RowAppend
RowDelete
RowDuplicate
RowInsert
RowMoveDown
RowMoveUp

FindDialog
FindNext
ReplaceDialog

ApplyTextState
IsTextStateApplied
IsTextStateEnabled

MarkUpView

Print
PrintPreview

CreateChild

GetAllowedElements
GetAllAttributes

GetNextVisible

GetPreviousVisible

SelectionMoveTabOrder
SelectionSet
ClearSelection

attachCallBack

ReloadToolbars

StartSpellChecking

GetFileVersion

RedrawEntryHelpers

SetUnmodified
ClearUndoRedo

Properties
AuthenticView
IsEditClearEnabled
IsEditCopyEnabled
IsEditCutEnabled
IsEditPasteEnabled
IsEditRedoEnabled
IsEditUndoEnabled

IsFindNextEnabled

IsRowAppendEnabled
IsRowDeleteEnabled
IsRowDuplicateEnabled
IsRowInsertEnabled
IsRowMoveDownEnabled
IsRowMoveUpEnabled

SchemaLoadObject
XMLDataLoadObject
DesignDataLoadObject

XMLDataSaveUrl

XMLRoot

CurrentSelection

event

validationBadData
validationMessage

ToolbarsEnabled
ToolbarTooltipsEnabled
AutoHideUnusedCommandGroups
TextStateToolbarLine
TextStateBmpURL
ToolbarRows
UICommands
XMLTable
BaseURL

EntryHelpersEnabled
EntryHelperSize
EntryHelperAlignment
EntryHelperWindows

EnableModifications
Modified
SaveButtonAutoEnable

Events
SelectionChanged
ControlInitialized

Description
Authentic Class

2.1.1 Authentic.ApplyTextState

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 bolded. 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 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.

2.1.2 Authentic.attachCallBack

Deprecated

Use connection point events instead as described here.

See also

Declaration: attachCallBack(*bstrName* as String,*varCallBack* as Variant)

Description
The Authentic View provides events which can be handled using custom callback functions. All event handlers take no parameters and any returned value will be ignored. To retrieve information when a specific event is raised you have to read the according properties of the event object.

List of currently available events:

ondragover
ondrop
onkeydown
onkeyup
onkeypressed
onmousemove
onbuttonup
onbuttondown
oneditpaste
oneditcut
oneditcopy

Since version 3.0.0.0:

ondoceditcommand

Since version: 5.3.0.0:

onbuttondoubleclick

JavaScript example:

```
// somwhere in your script:
objPlugIn.attachCallBack("ondragover",OnDragOver);
objPlugIn.attachCallBack("ondrop",OnDrop);

// event handlers
function OnDragOver()
{
 if(!objPlugIn.event.dataTransfer.ownDrag &&
  objPlugIn.event.dataTransfer.type == "TEXT"))
 {
  objPlugIn.event.dataTransfer.dropEffect = 1;
```

```
  objPlugIn.event.cancelBubble = true;
 }
}

// OnDrop() replaces the complete text value of the XML
// element with the selection from the drag operation
function OnDrop()
{
 var objTransfer = objPlugIn.event.dataTransfer;

 if(!objTransfer.ownDrag &&
  (objTransfer.type == "TEXT"))
  objPlugIn.event.srcElement.TextValue = objTransfer.getData();
}
```

2.1.3 AuthenticView

See also

Declaration: AuthenticView as AuthenticView (read-only)

Description
Returns an object that gives access to properties and methods specific to the Authentic view.

AuthenticView overlaps with the existing view specific functionality. Future versions of AuthenticView will include all view-specific functionality. The AuthenticView object is the recommended interface for all future implementations.

Examples
Please see the Bubble sort of dyamic tables for information on how to use the AuthenticView object.

2.1.4 Authentic.AutoHideUnusedCommandGroups

Declaration: AutoHideUnusedCommandGroups as Boolean

Description
True if unused Toolbar - CommandGroups are hidden automatically (e.g. XML - table commands, if the current SPS file does not support XML tables)
Default value: true

2.1.5 Authentic.BaseURL

Declaration: BaseURL as String

Description
This property sets the base URL to resolve relative paths.
If no URL is set, the location of the current XML file is used.

2.1.6 Authentic.ClearSelection

Declaration: ClearSelection()

Description
The method clears the current selection. Any following attempt to get the selection using the CurrentSelection property, fails until the user selects a node or a successful call to SelectionSet() has been processed.

Before a node that has the current selection can be deleted, the selection must be set to a different node or cleared.

2.1.7 Authentic.ClearUndoRedo

Declaration: ClearUndoRedo()

Description
The method clears the whole Undo/Redo buffer.

2.1.8 Authentic.ControlInitialized

See also

Declaration: ControlInitialized

Description
This event is raised when the control is created and initialized.

See also Connection point events.

2.1.9 Authentic.CreateChild

See also

Declaration: CreateChild(*nKind* as SPYXMLDataKind) as XMLData

Return Value
New XML node

Description

The CreateChild method is used to create new nodes which you can insert into the XML structure of the current document using the XMLData interface.

See also XMLData.AppendChild and XMLData.InsertChild

2.1.10 Authentic.CurrentSelection

See also

Declaration: CurrentSelection as AuthenticSelection

Description
The property provides access to the current selection in the Authentic View.

The example code below retrieves the complete text of the current selection:

JavaScript:

```
// somewhere in your script:
GetSelection(objPlugIn.CurrentSelection);

// GetSelection() collects complete text selection
function GetSelection(objSel)
{
 var strText = "";

 var objCurrent = objSel.Start;

 while(!objSel.End.IsSameNode(objCurrent))
 {
  objCurrent = objPlugIn.GetNextVisible(objCurrent);
  strText += objCurrent.TextValue;
 }

 strText += objSel.End.TextValue.substring(0,objSel.EndTextPosition);
 return objSel.Start.TextValue.substr(objSel.StartTextPosition) + strText;
}
```

2.1.11 Authentic.DesignDataLoadObject

See also

Declaration: DesignDataLoadObject as AuthenticLoadObject

Description
The DesignDataLoadObject contains a reference to the SPS document. The SPS document is used to generate the WYSIWYG editing environment and is typically generated by StyleVision.

See also SchemaLoadObject for an example.

2.1.12 Authentic.EditClear

See also

Declaration: EditClear

Description
Clears the current selection.

2.1.13 Authentic.EditCopy

See also

Declaration: EditCopy

Description
Copies the current selection to the clipboard.

2.1.14 Authentic.EditCut

See also

Declaration: EditCut

Description
Cuts the current selection from the document and copies it to the clipboard.

2.1.15 Authentic.EditPaste

See also

Declaration: EditPaste

Description
Pastes the content from the clipboard into the document.

2.1.16 Authentic.EditRedo

See also

Declaration: EditRedo

Description
Redo the last undo step.

2.1.17 Authentic.EditSelectAll

See also

Declaration: EditSelectAll

Description
The method selects the complete document.

2.1.18 Authentic.EditUndo

See also

Declaration: EditUndo

Description
Undo the last action.

2.1.19 Authentic.EnableModifications

Declaration: EnableModifications as Boolean

Description
True if Authentic Browser modifications to the XML content are enabled. See also the Modified property.

Default value: TRUE

2.1.20 Authentic.EntryHelperAlignment

Declaration: EntryHelperAlignment as SPYAuthenticToolbarAllignment

Description
This property can be used to set the position of the entry helpers.The default valueis 3, which places the entry helpers on the right side.

2.1.21 Authentic.EntryHelpersEnabled

Declaration: EntryHelpersEnabled as Boolean

Description
True if Authentic Browser entry helpers are enabled.
This property can be used to enable or disable the entry helpers.

Default value: FALSE

2.1.22 Authentic.EntryHelperSize

Declaration: EntryHelperSize as Integer

Description
This property can be used to set the initial size of the entry helpers area in pixels. Set the property to -1 if this value should be ignored.

Default value: -1

2.1.23 Authentic.EntryHelperWindows

Declaration: EntryHelperWindows as SPYAuthenticEntryHelperWindows

Description
This property can be used to define which of the entry helpers should be displayed. The values can be combined to display more than one entry helper window. The default value is 7 to show all three entry helpers.

2.1.24 Authentic.event

See also

Declaration: event as AuthenticEvent

Description
The event property holds an event data object which contains informations about the current event.

2.1.25 Authentic.FindDialog

See also

Declaration: FindDialog

Description
Displays the FindDialog.

See also Find and replace.

2.1.26 Authentic.FindNext

See also

Declaration: FindNext

Description
The method performs a find next operation.

See also Find and replace.

2.1.27 Authentic.GetAllAttributes

See also

Declaration: GetAllAttributes(*pForElement* as XMLData, *pElements* as Variant)

Description
GetAllAttributes() returns the allowed attributes for the specified element as an array of strings.

JavaScript example:

```
function GetAttributes()
{
 var arrElements = new Array(1);

 var objStart = objPlugIn.CurrentSelection.Start;

 var strText;
 strText = "Valid attributes at current selection:\n\n";

 for(var i = 1;i <= 4;i++)
 {
  objPlugIn.GetAllAttributes(objStart, 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 DisplayAllowedAttributes
 dim arrElements()

 dim objStart
 dim objEnd
 set objStart = objPlugIn.CurrentSelection.Start
 set objEnd = objPlugIn.CurrentSelection.End

 dim strText
```

```
 strText = "Valid attributes at current selection:" & chr(13) & chr(13)

 dim i

 For i = 1 To 4
  objView.GetAllAttributes objStart, 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
```

2.1.28 Authentic.GetAllowedElements

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 arrElements = new Array(1);

 var objStart = objPlugIn.CurrentSelection.Start;
 var objEnd = objPlugIn.CurrentSelection.End;

 var strText;
 strText = "valid elements at current selection:\n\n";

 for(var i = 0;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 arrElements()

 dim objStart
 dim objEnd
 set objStart = objPlugIn.CurrentSelection.Start
 set objEnd = objPlugIn.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
```

2.1.29 Authentic.GetFileVersion

See also

Declaration: GetFileVersion(*strVersion* as String)

Description
The method simply returns the version of the component as a string in the format 5.0.0.0.

2.1.30 Authentic.GetNextVisible

See also

Declaration: GetNextVisible(*pElement* as XMLData) as XMLData

Description
The method gets the next visible XML element in the document.

2.1.31 Authentic.GetPreviousVisible

See also

Declaration: GetPreviousVisible(*pElement* as XMLData) as XMLData

Description
The method gets the previous visible XML element in the document.

2.1.32 Authentic.IsEditClearEnabled

See also

Declaration: IsEditClearEnabled as Boolean

Description
True if EditClear is possible.

See also Editing operations.

2.1.33 Authentic.IsEditCopyEnabled

See also

Declaration: IsEditCopyEnabled as Boolean

Description
True if copy to clipboard is possible.

See also EditCopy and Editing operations.

2.1.34 Authentic.IsEditCutEnabled

See also

Declaration: IsEditCutEnabled as Boolean

Description
True if EditCut is currently possible.

See also Editing operations.

2.1.35 Authentic.IsEditPasteEnabled

See also

Declaration: IsEditPasteEnabled as Boolean

Description
True if EditPaste is possible.

See also Editing operations.

2.1.36 Authentic.IsEditRedoEnabled

See also

Declaration: IsEditRedoEnabled as Boolean

Description
True if EditRedo is currently possible.

See also Editing operations.

2.1.37 Authentic.IsEditUndoEnabled

See also

Declaration: IsEditUndoEnabled as Boolean

Description
True if EditUndo is possible.

See also Editing operations.

2.1.38 Authentic.IsFindNextEnabled

See also

Declaration: IsFindNextEnabled as Boolean

Description
True if FindNext is currently possible. False if no more occurances are left.

See also Find and replace and FindDialog.

2.1.39 Authentic.IsRowAppendEnabled

See also

Declaration: IsRowAppendEnabled as Boolean

Description
True if RowAppend is possible.

See also Row operations.

2.1.40 Authentic.IsRowDeleteEnabled

See also

Declaration: IsRowDeleteEnabled as Boolean

Description
True if RowDelete is possible.

See also Row operations.

2.1.41 Authentic.IsRowDuplicateEnabled

See also

Declaration: IsRowDuplicateEnabled as Boolean

Description
True if RowDuplicate is currently possible.

See also Row operations.

2.1.42 Authentic.IsRowInsertEnabled

See also

Declaration: IsRowInsertEnabled as Boolean

Description
True if RowInsert is possible.

See also Row operations.

2.1.43 Authentic.IsRowMoveDownEnabled

See also

Declaration: IsRowMoveDownEnabled as Boolean

Description
True if RowMoveDown is currently possible.

See also Row operations.

2.1.44 Authentic.IsRowMoveUpEnabled

See also

Declaration: IsRowMoveUpEnabled as Boolean

Description
True if RowMoveUp is possible.

See also Row operations.

2.1.45 Authentic.IsTextStateApplied

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.

2.1.46 Authentic.IsTextStateEnabled

See also

Declaration: IsTextStateEnabled(*elementName* 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.

2.1.47 Authentic.LoadXML

See also

Declaration: LoadXML(*xmlString* as String)

Description
Loads the current XML document with the XML string applied. The new content is displayed immediately.

2.1.48 Authentic.MarkUpView

See also

Declaration: MarkUpView(*kind* as SPYAuthenticMarkupVisibility)

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 different types of markup tags.

2.1.49 Authentic.Modified

Declaration: Modified as Boolean

Description
True if XML content is modified.

This property is read-only.

2.1.50 Authentic.Print

See also

Declaration: Print

Description
Print the current document being edited.

2.1.51 Authentic.PrintPreview

See also

Declaration: PrintPreview

Description
Print preview the document being edited.

2.1.52 Authentic.RedrawEntryHelpers

Declaration: RedrawEntryHelpers()

Description
RedrawEntryHelpers takes the values from the EntryHelpersEnabled, EntryHelperAlignment, EntryHelperSize and EntryHelperWindows properties and redraws the entry helper windows.

2.1.53 Authentic.ReloadToolbars

Declaration: ReloadToolbars()

Description
ReloadToolbars reads the ToolbarRows collection and redraws the toolbars and the view.

2.1.54 Authentic.ReplaceDialog

See also

Declaration: ReplaceDialog

Description
Displays the ReplaceDialog.

See also Find and replace.

2.1.55 Authentic.Reset

Deprecated

Use `Authentic.StartEditing` instead.

See also

Declaration: Reset

Description
Reset the data being edited. Typically called before editing a new set of XML, XSL and SPS documents.

The method does not change the view and its still possible to continue working with the currently displayed document.

2.1.56 Authentic.RowAppend

See also

Declaration: RowAppend

Description
Appends a row at the current position.

See also Row operations.

2.1.57 Authentic.RowDelete

See also

Declaration: RowDelete

Description
Deletes the currently selected row(s).

See also Row operations.

2.1.58 Authentic.RowDuplicate

See also

Declaration: RowDuplicate

Description
The method duplicates the currently selected rows.

See also Row operations.

2.1.59 Authentic.RowInsert

See also

Declaration: RowInsert

Description
Inserts a new row immediately above the current selection.

See also Row operations.

2.1.60 Authentic.RowMoveDown

See also

Declaration: RowMoveDown

Description
Moves the current row one position down.

See also Row operations.

2.1.61 Authentic.RowMoveUp

See also

Declaration: RowMoveUp

Description
Moves the current row one position up.

See also Row operations.

2.1.62 Authentic.Save

See also

Declaration: Save

Description
Saves the document to the URL specified by the property XMLDataSaveUrl. For the Untrusted versions, you can also use a full local path.

The plug-in sends an HTTP PUT request to the server to save the currently displayed XML file.

2.1.63 Authentic.SaveButtonAutoEnable

Declaration: SaveButtonAutoEnable as Boolean

Description
If this property is set to TRUE, the enabled/disabled state of the Save button in the Toolbar of the control is set according to the Modified flag of the document.

Default value is FALSE.

2.1.64 Authentic.SavePOST

See also

Declaration: SavePOST

Description
Saves the document to the URL specified by the property XMLDataSaveUrl. For the Untrusted versions, you can also use a full local path. The plug-in sends an HTTP POST request to the server to save the currently displayed XML file.

Checking whether file was saved or not
If the Authentic plug-in receives an HTTP response of `>= 300` it will assume the file has **not** been saved and will (by default) pop-up first a message box containing the HTTP error response, then a second (suppressible) message box advising the user that the file has not been saved. It is up to the application developer to ensure that the correct HTTP response is returned according to the success or failure of the save attempt. An example of PHP code to achieve this might look like this:

```
<?php
// suppress error messages to prevent any output
// being generated before headers can be sent
error_reporting (0);
$error = false;
$handle = fopen ( "result.xml", "w+" );
if (! $handle)
    $error = true;
else
{
    if (! fwrite($handle, $HTTP_RAW_POST_DATA))
           $error = true;
    else
           fclose($handle);
}
if ($error)
    header( "HTTP/1.1 500 Server Error" );
?>
```

2.1.65 Authentic.SaveXML

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.

2.1.66 Authentic.SchemaLoadObject

See also

Declaration: SchemaLoadObject as AuthenticLoadObject

Description
The SchemaLoadObject contains an reference to the XML Schema document for the current XML file. The Schema document is typically generated using XMLSpy.

Example

```
objPlugIn.SchemaLoadObject.URL = "http://www.YOURSERVER.com/OrgChart.xsd"
objPlugIn.XMLDataLoadObject.URL = "http://www.YOURSERVER.com/OrgChart.xml"
objPlugIn.DesignDataLoadObject.URL = "http://www.YOURSERVER.com/OrgChart.sps"
objPlugIn.StartEditing
```

The code above sets all URL properties of the load objects and calls StartEditing to load and display the files. For the Untrusted versions, you can also use a full local path. The current content and status of the plug-in will be cleared.

2.1.67 Authentic.SelectionChanged

See also

Declaration: SelectionChanged as VT_0019

Description
This event is raised whenever the user changes the current selection.

See also Connection point events.

2.1.68 Authentic.SelectionMoveTabOrder

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.

2.1.69 Authentic.SelectionSet

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.

2.1.70 Authentic.SetUnmodified

Declaration: SetUnmodified()

Description
After calling this method, the current condition of the Undo/Redo buffer is taken as the clean state of the underlying XML document and the Modified flag is set to FALSE.

2.1.71 Authentic.StartEditing

See also

Declaration: StartEditing as Boolean

Return Value
True if all files were successfully loaded and displayed.

Description
Start editing the current document. It is important to set the properties of the load objects SchemaLoadObject, DesignDataLoadObject and XMLDataLoadObject first.

2.1.72 Authentic.StartSpellChecking

Declaration: StartSpellChecking()

Description
This command opens the spell check dialog if a package containing the spell check engine and the required lexicons is available and activated.

2.1.73 Authentic.TextStateBmpURL

Declaration: TextStateBmpURL as String

Description
The URL from which the bitmaps for the text-state icons should be retrieved.
If no URL is specified, no text-state buttons are displayed.

Examples
objPlugIn.TextStateBmpURL = "<http://plugin.xmlspy.com/textstates/>"

```
<PARAM NAME="TextStateBmpURL" VALUE="http://plugin.xmlspy.com/textstates/">
```

2.1.74 Authentic.TextStateToolbarLine

Declaration: TextStateToolbarLine as long

Description
The toolbar (line number) in which the text-state icons should be placed. Text-state icons can be appended to existing toolbars or placed in new toolbars.
Default value: 1

2.1.75 Authentic.ToolbarRows

Declaration: ToolbarRows as AuthenticToolbarRows

Description
Gets a collection of all toolbars to be displayed. See description of AuthenticToolbarRows, how toolbars can be deleted, added or modified.

2.1.76 Authentic.ToolbarsEnabled

Declaration: ToolbarsEnabled as Boolean

Description
True if Authentic Browser Toolbars are enabled.
This property can be used to enable or disable all toolbars.
Default value: true

2.1.77 Authentic.ToolbarTooltipsEnabled

Declaration: ToolbarTooltipsEnabled as Boolean

Description
True if the Tooltips for the Authentic Browser Toolbars are enabled.
Default value: true

2.1.78 Authentic.UICommands

Declaration: UICommands as AuthenticCommands

Description
Gets a collection of all available toolbar commands (with description).
For these commands, buttons can be placed on different toolbars.
Read only.

Example
Get all available commands, command-groups, descriptions, and show them in a message box

```
dim str
for each UICommand in objPlugin.UICommands
str = str & UICommand.CommandID & " | " & UICommand.Group & " | " &
UICommand.ShortDescription & chr(13)
next
msgbox str
```

2.1.79 Authentic.ValidateDocument

See also

Declaration: ValidateDocument(*showResults* as Boolean) as Boolean

Return Value
result of validation

Description
Validates the current XML data for correctness as per the XML schema data. If the parameter showResults is FALSE then the validation errors will be suppressed, otherwise validation errors are shown.

2.1.80 Authentic.validationBadData

See also

Declaration: validationBadData as XMLData

Description
This property can provide additional information about the last validation error. It gets set after a call to `ValidateDocument()` and is either null or holds a reference to the XML element which causes the error.

2.1.81 Authentic.validationMessage

See also

Declaration: validationMessage as String

Description
If the validation failed (after a call to ValidateDocument) this property stores a string with the error message.

2.1.82 Authentic.XMLDataLoadObject

See also

Declaration: XMLDataLoadObject as AuthenticLoadObject

Description
The XMLDataLoadObject contains an reference to the XML document being edited. The XML document is typically defined using XMLSpy, but generated using a database or another business process.

See also SchemaLoadObject for an example.

2.1.83 Authentic.XMLDataSaveUrl

See also

Declaration: XMLDataSaveUrl as String

Description
When the XML data has been modified it is possible to save the data back to a server using an URL. When saving the XML data via an `HTTP PUT` using the `Authentic.Save` method, this property defines the location where the XML data will be saved. When posting the data via an `HTTP POST` using the `Authentic.SavePOST` method, this property defines the location of a server-side script/application which will process the `POST` data. For the Untrusted versions, you can also use a full local path.

See also the Authentic.Save and the Authentic.SavePOST methods.

2.1.84 Authentic.XMLRoot

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 informations.

2.1.85 Authentic.XMLTable

Declaration: XMLTable as AuthenticXMLTableCommands

Description
Get a set of all XML table commands.
Read only.

2.2 AuthenticCommand

Methods

Properties

CommandID
Group
ShortDescription
Name

2.2.1 AuthenticCommand.CommandID

Declaration: CommandID as SPYAuthenticCommand

Description
Get the CommandId of the command
Possible values: see AuthenticToolbarButton
Read only

Example
See Example at Authentic.UICommands

2.2.2 AuthenticCommand.Group

Declaration: Group as SPYAuthenticCommandGroup

Description
The CommandGroup to which the command belongs to.
Read only

Example
See Example at Authentic.UICommands

2.2.3 AuthenticCommand.ShortDescription

Declaration: ShortDescription as String

Description
Short description of the command (e.g. the tooltip text)
Read only

Example
See Example at Authentic.UICommands

2.2.4 AuthenticCommand.Name

Declaration: Name as String

Description
for future use

2.3 AuthenticCommands

Methods

Item

Properties

Count

2.3.1 AuthenticCommands.Count

Declaration: Count as long

Description
Number of available UI commands.
Read only

2.3.2 AuthenticCommands.Item

Declaration: Item (nPosition as long) as AuthenticCommand

Description
Get command of position nPosition. nPosition starts with 1.

2.4 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.

2.4.1 CountItems

Method: `CountItems()`

Return Value
Returns number of menu items.

Errors
2501 Invalid object.

2.4.2 DeleteItem

Method: `DeleteItem(position as integer)`

Return Value
Deletes an existing menu item.

Errors
2501 Invalid object
2502 Invalid index

2.4.3 GetItemText

Method: `GetItemText(position as integer)` menu item name as string

Return Value
Gets the menu item's name.

Errors
2501 Invalid object
2502 Invalid index

2.4.4 InsertItem

Method: `InsertItem(position as integer, menu item name as string, macro name as string)`

Return Value

Inserts a user-defined menu item. 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

2.4.5 SetItemText

Method: `SetItemText(position as integer, menu item name as string)`

Return Value
Sets the menu item's name.

Errors
2501 Invalid object
2502 Invalid index

2.5 AuthenticDataTransfer

See also

Methods
getData

Properties
dropEffect
ownDrag
type

Description
AuthenticDataTransfer interface.

2.5.1 AuthenticDataTransfer.dropEffect

See also

Declaration: 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 set AuthenticEvent.cancelBubble to TRUE.

2.5.2 AuthenticDataTransfer.getData

See also

Declaration: getData as Variant

Description
getData gets the actual data associated with this dataTransfer object. See also AuthenticDataTransfer.type for more informations.

2.5.3 AuthenticDataTransfer.ownDrag

See also

Declaration: ownDrag as Boolean

Description
The property is TRUE if the current dragging source comes from inside of the Authentic View.

2.5.4 AuthenticDataTransfer.type

See also

Declaration: type as String

Description
Holds the type of the data you get with the AuthenticDataTransfer.getData method.

Currently supported data types are:

OWN	data from plug-in itself
TEXT	plain text
UNICODETEXT	plain text as UNICODE
IUNKNOWN	IUnknown reference to IDataObject instance

2.6 AuthenticEvent

See also

Properties
altKey
altLeft
ctrlKey
ctrlLeft
shiftKey
shiftLeft

keyCode
repeat

button

clientX
clientY

dataTransfer

srcElement
fromElement

propertyName

cancelBubble
returnValue

type

Description
AuthenticEvent interface.

2.6.1 AuthenticEvent.altKey

See also

Declaration: altKey as Boolean

Description
True if the right ALT key is pressed.

2.6.2 AuthenticEvent.altLeft

See also

Declaration: altLeft as Boolean

Description
True if the left ALT key is pressed.

2.6.3 AuthenticEvent.button

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.

The onbuttondown and onbuttonup events set the button value in different ways. The onbuttonup event just sets the value for the button which has been released and raised the up event regardless which buttons are also pressed at the moment.

2.6.4 AuthenticEvent.cancelBubble

See also

Declaration: cancelBubble as Boolean

Description
Set cancelBubble to TRUE if the default event handler should not be called.

2.6.5 AuthenticEvent.clientX

See also

Declaration: clientX as long

Description
X value of the current mouse position in client coordinates.

2.6.6 AuthenticEvent.clientY

See also

Declaration: clientY as long

Description
Y value of the current mouse position in client coordinates.

2.6.7 AuthenticEvent.ctrlKey

See also

Declaration: ctrlKey as Boolean

Description
True if the right CTRL key is pressed.

2.6.8 AuthenticEvent.ctrlLeft

See also

Declaration: ctrlLeft as Boolean

Description
True if the left CTRL key is pressed.

2.6.9 AuthenticEvent.dataTransfer

See also

Declaration: dataTransfer as Variant

Description
property dataTransfer

2.6.10 AuthenticEvent.fromElement

See also

Declaration: fromElement as Variant

Description
Currently no event sets this property.

2.6.11 AuthenticEvent.keyCode

See also

Declaration: keyCode as long

Description
Keycode of the currently pressed key.

This property is read-write.

2.6.12 AuthenticEvent.propertyName

See also

Declaration: propertyName as String

Description
Currently no event sets this property.

2.6.13 AuthenticEvent.repeat

See also

Declaration: repeat as Boolean

Description
True if the onkeydown event is repeated.

2.6.14 AuthenticEvent.returnValue

See also

Declaration: returnValue as Variant

Description
Use returnValue to set a return value for your event handler.

2.6.15 AuthenticEvent.shiftKey

See also

Declaration: shiftKey as Boolean

Description
True if the right SHIFT key is pressed.

2.6.16 AuthenticEvent.shiftLeft

See also

Declaration: shiftLeft as Boolean

Description
True if the left SHIFT key is pressed.

2.6.17 AuthenticEvent.srcElement

See also

Declaration: srcElement as Variant

Description
Element which fires the current event.
This is usually an XMLData object.

Since version 3.0.0.0:

The property can also hold a reference to an AuthenticCommand object if was set from an ondoceditcommand event.

2.6.18 AuthenticEvent.type

See also

Declaration: type as String

Description
Currently no event sets this property.

2.7 AuthenticEventContext

The `EventContext` interface gives access to many properties of the context in which a macro is executed.

2.7.1 EvaluateXPath

Method: `EvaluateXPath` (`string` expression) 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.

2.7.2 GetEventContextType

Method: `GetEventContextType ()` as `EventContextType` 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.

2.7.3 GetNormalizedTextValue

Method: `GetNormalizedTextValue()` value 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.

2.7.4 GetVariableValue

Method: `GetVariableValue(name as string, value as string)`

Return Value
Gets the value of the named variable.

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

2.7.5 GetXMLNode

Method: `GetXMLNode()` XMLData object

Return Value
Returns the context XML node or `NULL`

Errors

2201	Invalid object.
2202	No context.
2203	Invalid context
2209	Invalid parameter.

2.7.6 IsAvailable

Method: `IsAvailable()`

Return Value
Returns true if `EventContext` is set.

Errors
2201 Invalid object.

2.7.7 SetVariableValue

Method: `SetVariableValue(name as string, value as string)`

Return Value
Sets the value of the named variable.

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

2.8 AuthenticLoadObject

See also

Properties
String
URL

Description
The XMLSpyXMLLoadSave object is used to set the source for the files you need to load. You can either set the content directly via the String property or as an external location via the URL property.

See also Authentic.SchemaLoadObject, Authentic.DesignDataLoadObject and Authentic.XMLDataLoadObject for more informations about how to use them.

2.8.1 AuthenticLoadObject.String

See also

Declaration: String as String

Description
You can use this property to set the XML structure from a string. The URL property of the object must be empty if you want to use this property.

2.8.2 AuthenticLoadObject.URL

See also

Declaration: URL as String

Description
The property should contain a valid URL for the load or save operation. Currently supported HTTP protocols are http, https, ftp and gopher.

2.9 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		Methods
Application	Clone	MoveBegin
FirstTextPosition	CollapsToBegin	MoveEnd
FirstXMLData	CollapsToEnd	NextCursorPosition
FirstXMLDataOffset	ExpandTo	PreviousCursorPosition
LastTextPosition	Goto	Select
LastXMLData	GotoNext	SelectNext
LastXMLDataOffset	GotoPrevious	SelectPrevious
Parent	IsEmpty	SetFromRange
	IsEqual	

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
Text	Copy	AppendRow
	Cut	DeleteRow
	Delete	DuplicateRow
	Paste	InsertRow
		IsInDynamicTable
		MoveRowDown
		MoveRowUp

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows

Elements	Attributes	Entities
CanPerformActionWith	GetElementAttributeValue	GetEntityNames
CanPerformAction	GetElementAttributeNames	InsertEntity
PerformAction	GetElementHierarchy	
	HasElementAttribute	
	SetElementAttributeValue	

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 currrent 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 specic 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.

2.9.1 AuthenticRange.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

```
' ----------------------------------------------------------
'                         VBScript
' Append row at end of current dynamically growable table
' ----------------------------------------------------------
Dim objRange
Set objRange = objPlugin.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
```

2.9.2 AuthenticRange.Application

See also

Property: Application as Authentic (read-only)

Description
Access the 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.

2.9.3 AuthenticRange.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 modifcation 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.

2.9.4 AuthenticRange.CanPerformActionWith

See also

Method: CanPerformActionWith (*eAction* as SPYAuthenticActions, *out_arrElementNames* as Variant)

Description

CanPerformActionWith 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 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 PerformAction.

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.

2.9.5 AuthenticRange.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.

2.9.6 AuthenticRange.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.

2.9.7 AuthenticRange.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.

2.9.8 AuthenticRange.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.

2.9.9 AuthenticRange.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.

2.9.10 AuthenticRange.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.

2.9.11 AuthenticRange.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

```
' ----------------------------------------------------------
'                       VBScript
' Delete selected row from dynamically growing table
' ----------------------------------------------------------
Dim objRange
Set objRange = objPlugin.ActiveDocument.AuthenticView.Selection

' check if we are in a table
If objRange.IsInDynamicTable Then
     objRange.DeleteRow
End If
```

2.9.12 AuthenticRange.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

```
' ------------------------------------------------------
'                       VBScript
' dupplicate row in current dynamically growable table
' ------------------------------------------------------
Dim objRange
Set objRange = objPlugin.ActiveDocument.AuthenticView.Selection

' check if we can insert soemthing
If objRange.IsInDynamicTable Then
     objRange.DuplicateRow
     ' objRange points to begining of new row
     objRange.Select
End If
```

2.9.13 AuthenticRange.EvaluateXPath

Method: `EvaluateXPath` (string expression) 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 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

2.9.14 AuthenticRange.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.

2.9.15 AuthenticRange.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

```
' ---------------------------------------
'               VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = objPlugin.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
```

2.9.16 AuthenticRange.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 canot be accessed.

Examples

```
' ------------------------------------------------
'                 VBScript
' show name of currently selected XMLData element
' ------------------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.17 AuthenticRange.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 ComboBoxes, CheckBoxes 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

```
' ---------------------------------------------
'                VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.18 AuthenticRange.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.

2.9.19 AuthenticRange.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.

2.9.20 AuthenticRange.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.

Examples
See SetElementAttributeValue.

2.9.21 AuthenticRange.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 InsertEntity.

2.9.22 AuthenticRange.GetVariableValue

Method: `GetVariableValue(name as string, value as string)`

Return Value
Gets the value of the named variable.

Errors

2001	Invalid object.
2005	Invalid parameter
2202	No context.
2204	No such variable in scope
2205	Variable cannot be evaluated
2206	Variable returns sequence

2.9.23 AuthenticRange.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.

2.9.24 AuthenticRange.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

```
' ------------------------------------------
'                VBScript
' Scan through the whole document word-by-word
' ------------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.25 AuthenticRange.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.

2.9.26 AuthenticRange.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

```
' ------------------------------------------
'                  VBScript
' Scan through the whole document tag-by-tag
' ------------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.27 AuthenticRange.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.

2.9.28 AuthenticRange.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.

2.9.29 AuthenticRange.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

```
' ----------------------------------------------------------
'                          VBScript
' Insert the first entity in the list of availabel entities
' ----------------------------------------------------------
Dim objRange
Set objRange = objPlugin.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)
     objRange.Select()
Else
     MsgBox "Sorry, no entities are available for this document"
End If
```

2.9.30 AuthenticRange.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

```
' ------------------------------------------------------------
'                       VBScript
' Insert row at beginning of current dynamically growing table
' ------------------------------------------------------------
Dim objRange
Set objRange = objPlugin.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
```

2.9.31 AuthenticRange.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.

2.9.32 AuthenticRange.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.

2.9.33 AuthenticRange.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.

2.9.34 AuthenticRange.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.

2.9.35 AuthenticRange.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.

2.9.36 AuthenticRange.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.

2.9.37 AuthenticRange.IsInDynamicTable

See also

Method: IsInDynamicTable () as Boolean

Description
Test if the beginning of the range is inside a table that supports the different row operations.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

2.9.38 AuthenticRange.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.

2.9.39 AuthenticRange.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.

2.9.40 AuthenticRange.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.

2.9.41 AuthenticRange.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.

2.9.42 AuthenticRange.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. Decrementing 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

```
' ---------------------------------------
'               VBScript
' ---------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.43 AuthenticRange.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 canot be accessed.

2.9.44 AuthenticRange.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 ComboBoxes, CheckBoxes 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

```
' ---------------------------------------------
'                  VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.45 AuthenticRange.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.

2.9.46 AuthenticRange.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.

2.9.47 AuthenticRange.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.

2.9.48 AuthenticRange.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.

2.9.49 AuthenticRange.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.

2.9.50 AuthenticRange.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.

2.9.51 AuthenticRange.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

```
' ------------------------------------------
'             VBScript
' Insert the innermost element
' ------------------------------------------
Dim objRange
Set objRange = objPlugin.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
```

2.9.52 AuthenticRange.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

```
' ---------------------------------------
'                  VBScript
' ---------------------------------------

' set current selection to end of document
objPlugin.objAuthenticView.DocumentEnd.Select()
```

2.9.53 AuthenticRange.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

```
' ------------------------------------------
'                  VBScript
' Scan through the whole document word-by-word
' ------------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.54 AuthenticRange.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

```
' ------------------------------------------
'                VBScript
' Scan through the whole document tag-by-tag
' ------------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.9.55 AuthenticRange.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

```
' ----------------------------------------
'            VBScript
' Get and set element attributes
' ----------------------------------------
Dim objRange
Set objRange = objPlugin.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
```

2.9.56 AuthenticRange.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.

2.9.57 AuthenticRange.SetVariableValue

Enter topic text ***Method:*** `SetVariableValue(name as string, value as string)`

Return Value
Sets the value of the named variable.

Errors

2001	Invalid object.
2002	Missing context node.
2204	No such variable in scope
2205	Variable cannot be evaluated
2206	Variable returns a sequence
2207	Variable read-only
2208	No modification allowed

2.9.58 AuthenticRange.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.

2.10 AuthenticSelection

See also

Properties

Start
StartTextPosition
End
EndTextPosition

2.10.1 AuthenticSelection.End

See also

Declaration: End as XMLData

Description
XML element where the current selection ends.

2.10.2 AuthenticSelection.EndTextPosition

See also

Declaration: EndTextPosition as long

Description
Position in Authentic.End where the selection ends.

2.10.3 AuthenticSelection.Start

See also

Declaration: Start as XMLData

Description
XML element where the current selection starts.

2.10.4 AuthenticSelection.StartTextPosition

See also

Declaration: StartTextPosition as long

Description
Position in Authentic.Start where the selection starts.

2.11 AuthenticToolbarButton

Methods

Properties

CommandID

2.11.1 AuthenticToolbarButton.CommandID

Declaration: CommandID as SPYAuthenticCommand

Description
The CommandId of the toolbar button.

2.12 AuthenticToolbarButtons

Methods

Item

NewButton
NewCustomButton
NewSeparator

Remove

Properties

Count

2.12.1 AuthenticToolbarButtons.Count

Declaration: Count as long

Description
Get the actual number of buttons on the toolbar
Read only.

2.12.2 AuthenticToolbarButtons.Item

Declaration: Item(n as long) as AuthenticToolbarButton

Description
Get the n'th Button of the current toolbar. n starts at 1.

Example
See the example at AuthenticToolbarButtons.NewButton

2.12.3 AuthenticToolbarButtons.NewButton

Declaration: NewButton(nPosition as long, nCommandId as SPYAuthenticCommand)

Description
Inserts a new Button for the command nCommandId at the position nPosition of the toolbar. nPosition starts at 1.

Example
Add a new toolbar, align it on the bottom and add new toolbar buttons

```
objPlugIn.ToolbarRows.NewRow(3)                    // add a new Toolbar (Row 3)
set ToolbarRow  = objPlugIn.ToolbarRows.Item(3)
set Buttons = ToolbarRow.Buttons
ToolbarRow.Alignment    = 2          // align Toolbar to bottom
Buttons.NewButton       1, 2         // add Print Button
Buttons.NewButton       1, 3         // add PrintPreview Button
Buttons.NewSeparator    2            // add Separator
Buttons.NewButton       1, 4         // add Validate Button
```

When StartEditing or ReloadToolbars is called; the modified Toolbar settings are used.

2.12.4 AuthenticToolbarButtons.NewCustomButton

Declaration: NewCustomButton(nPosition as long, strName as String, strTooltip as String, strBitmapURL as String)

Description
Inserts a new custom button with the name strName at the position nPosition of the toolbar. nPosition starts at 1.
strTooltip is taken as the Tooltip text.

strBitmapURL is the location of the bitmap to display for the new button. This path is relative to the path set with the TextStateBmpURL property.

Example

Imagine you have inserted a custom button with the name "MyFunction" to your toolbars. The following event handler for doceditcommand shows you how to react if the user clicked your button and how to set the enabled / disabled state for it.

```
<SCRIPT LANGUAGE=javascript FOR=objPlugIn EVENT=doceditcommand>
 // event.type is set to "command" if the user clicked the button
 if(objPlugIn.event.type == "command")
 {
  if(objPlugIn.event.srcElement.Name == "MyFunction")
   window.alert("You pressed my custom button!");
 }

 // event.type is set to "update" if the button state must be set
 if(objPlugIn.event.type == "update")
 {
  if(objPlugIn.event.srcElement.Name == "MyFunction")
  {
   // we enable the button if only one element is selected

if(objPlugIn.CurrentSelection.Start.IsSameNode(objPlugIn.CurrentSelection.End))
     objPlugIn.event.returnValue = 1;
   else
    objPlugIn.event.returnValue = 0;

   objPlugIn.event.cancelBubble = true;
  }
 }
</SCRIPT>
```

2.12.5 AuthenticToolbarButtons.NewSeparator

Declaration: NewSeparator(nPosition as long)

Description
Inserts a Separator at the position nPosition of the toolbar. nPosition starts at 1.

Example
See the example at AuthenticToolbarButtons.NewButton

2.12.6 AuthenticToolbarButtons.Remove

Declaration: Remove(nPosition as long)

Description
Removes the button or Separator on position nPosition of the toolbar. nPosition starts at 1.

2.13 AuthenticToolbarRow

Methods

Alignment

Properties

Buttons

2.13.1 AuthenticToolbarRowAlignment

Declaration: Alignment(nAlign as SPYAuthenticToolbarAllignment)

Description
Get or sets the alignment of the toolbar in the plug-in.

Example
Align all toolbars to the bottom:

```
for each ToolbarRow in objPlugin.ToolbarRows
 ToolbarRow.Alignment = 2
next
```

2.13.2 AuthenticToolbarRowButtons

Declaration: Buttons as AuthenticToolbarButtons

Description
Get all Buttons of the toolbar

Example
See the example at AuthenticToolbarButtons.NewButton

2.14 AuthenticToolbarRows

Methods

Item
RemoveRow
NewRow

Properties

Count

2.14.1 AuthenticToolbarRows.Count

Declaration: Count as long

Description
Get the actual number of defined toolbars
Read only.

2.14.2 AuthenticToolbarRows.Item

Declaration: Item(nPosition as long) as AuthenticToolbarRow

Description
Get the toolbar on position nPosition. nPosition starts with "1"
Read only.

Example
See the example at AuthenticToolbarButtons.NewButton

2.14.3 AuthenticToolbarRows.RemoveRow

Declaration: RemoveRow(nPosition as long)

Description
Remove the toolbar at nPosition from the user interface. nPosition starts at 1.

2.14.4 AuthenticToolbarRows.NewRow

Declaration: NewRow(nPosition as long)

Description
Creates and inserts a new Toolbar row. nPosition starts at 1.

Example
See the example at AuthenticToolbarButtons.NewButton

2.15 AuthenticView

See also

Properties	Methods	Events
Application	Goto	
DocumentBegin	Print	
DocumentEnd	Redo	
MarkupVisibility	Undo	
Parent	UpdateXMLInstanceEntities	
Selection		
WholeDocument		

Description

AuthenticView and its child object AuthenticRange provide you with an interface for Authentic View, which allow easy and consistent modification of document contents. Functional overlap with already existing methods and properties in Authentic object is intentional, makeing the content modification functions of the Authentic object obsolete. Usage of the new AuthenticView interface is strongly recommended.

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.

2.15.1 Events

OnBeforeCopy

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

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

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

Event: `OnBeforeDrop (i_nXPos as Long, i_nYPos as Long, i_ipRange as AuthenticRange, i_ipData as cancelBoolean`

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

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

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

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 false;  /* 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.

REMARK: The following events from the scripting environment and IDE Plugin of XMLSpy are still supported but become obsolete with this event:

```
On_AuthenticKeyUp()          IXMLSpyPlugIn.OnEvent (13, ...)    //
nEventId = 13
On_AuthenticKeyDown()        IXMLSpyPlugIn.OnEvent (12, ...)    //
nEventId = 12
On_AuthenticKeyPressed()     IXMLSpyPlugIn.OnEvent (14, ...)    //
nEventId = 14
```

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 OnKeyboard
' event of object objView
Private Function objView_OnKeyboardEvent(ByVal i_keyEvent As Long, ByVal
io_pnKeyCode As Long, ByVal i_nVirtualKeyStatus As Long) As Boolean
    If ((i_keyEvent = XMLSpyLib.spyKeyUp) And ((i_nVirtualKeyStatus And
XMLSpyLib.spyCtrlKeyMask) <> 0)) Then
        MsgBox ("Ctrl " & io_pnKeyCode & " pressed")
        objView_OnKeyboardEvent = True
    Else
        objView_OnKeyboardEvent = False
    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
```

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

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 false;  /* 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*.

REMARK: The following events from the scripting environment and IDE Plugin of XMLSpy are still supported but become obsolete with this event:

```
On_AuthenticMouseMove()                  IXMLSpyPlugIn.OnEvent (15, ...)
     //  nEventId = 15
On_AuthenticButtonUp()             IXMLSpyPlugIn.OnEvent (16, ...)    //
nEventId = 16
On_AuthenticButtonDown()                 IXMLSpyPlugIn.OnEvent (17, ...)
     //  nEventId = 17
On_AuthenticButtonDoubleClick()    IXMLSpyPlugIn.OnEvent (24, ...)    //
nEventId = 24
```

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 OnMouseEvent
' event of object objView. If you click with the left mouse button
' while pressing a control key, the current selection will be set
' to the tag below the current mouse cursor position
Private Function objView_OnMouseEvent(ByVal i_nXPos As Long, ByVal i_nYPos As
Long, ByVal i_eMouseEvent As XMLSpyLib.SPYMouseEvent, ByVal i_pRange As
XMLSpyLib.IAuthenticRange) As Boolean
    If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
        On Error Resume Next
        i_pRange.Select
        objView_OnMouseEvent = True
    Else
        objView_OnMouseEvent = False
    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
```

OnSelectionChanged

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

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 (in the `Authentic/Scripting` folder of the `Examples` project in the Project Window).

2.15.2 AuthenticView.Application

See also

Property: Application as Authentic (read-only)

Description
Access the application object.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

2.15.3 AuthenticView.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.

2.15.4 AuthenticView.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.

2.15.5 AuthenticView.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.

2.15.6 AuthenticView.DisableAttributeEntryHelper

Method: `DisableAttributeEntryHelper()`

Description

`DisableAttributeEntryHelper()` disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors

2000 Invalid object.

2.15.7 AuthenticView.DisableElementEntryHelper

Method: `DisableElementEntryHelper()`

Description

`DisableElementEntryHelper()` disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors

2000 Invalid object.

2.15.8 AuthenticView.DisableEntityEntryHelper

Method: `DisableEntityEntryHelper()`

Description

`DisableEntityEntryHelper()` disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors

2000 Invalid object.

2.15.9 AuthenticView.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.

2.15.10 AuthenticView.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.

2.15.11 AuthenticView.DoNotPerformStandardAction

Method: `DoNotPerformStandardAction ()`

Description

`DoNotPerformStandardAction()` serves as cancel bubble for macros; stops further execution after macro has finished.

Errors

2000 Invalid object.

2.15.12 AuthenticView.EvaluateXPath

Method: `EvaluateXPath` (XMLData, string expression) as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expressions with the given XML context node. The result 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.

2.15.13 AuthenticView.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.

2.15.14 AuthenticView.EventContext

Property: `EventContext` as `EventContext`

Description

`EventContext` property gives access to the running macros context. See the `EventContext` interface description for more below details.

Errors

2000 Invalid object.

2.15.15 AuthenticView.GetToolbarButtonState

Method: `GetToolbarButtonState` (`ButtonIdentifier` as `string`) as `AuthenticToolbarButtonState`

Return Value
The method returns `AuthenticToolbarButtonState`

Description
`GetToolbarButtonState` 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.

2.15.16 AuthenticView.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

```
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.15.17 AuthenticView.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.

2.15.18 AuthenticView.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.

2.15.19 AuthenticView.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.

2.15.20 AuthenticView.Parent

See also

Property: Parent as Authentic (read-only)

Description
Access the Application.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

2.15.21 AuthenticView.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.

2.15.22 AuthenticView.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.

2.15.23 AuthenticView.Selection

See also

Property: Selection as AuthenticRange

Description
Set or get current text selection in user interface.

Errors

2000	The authentic view object is no longer valid.
2002	No cursor selection is active.
2005	Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------
'               VBScript
' ---------------------------------------
Dim objAuthenticView
Set objAuthenticView = objPlugin.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
```

2.15.24 AuthenticView.SetToolbarButtonState

Method: `SetToolbarButtonState` (`ButtonIdentifier` as string, `AuthenticToolbarButtonState` state)

Description

`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.

2.15.25 AuthenticView.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.

2.15.26 AuthenticView.UpdateXMLInstanceEntities

See also

Method: UpdateXMLInstanceEntities ()

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example

```
// ----------------------------------------
//               JavaScript
// ----------------------------------------
var objDocType;
objDocType = objPlugin.XMLRoot.GetFirstChild(10);

if(objDocType)
{
     var objEntity = objPlugin.CreateChild(14);
     objEntity.Name = "child";
     objEntity.TextValue = "SYSTEM \"child.xml\"";
     objDocType.AppendChild(objEntity);

     objPlugin.AuthenticView.UpdateXMLInstanceEntities();
}
```

2.15.27 AuthenticView.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.

2.15.28 AuthenticView.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.

2.16 AuthenticXMLTableCommands

Methods

Insert
Delete
AppendRow
InsertRow
DeleteRow
AppendColumn
InsertColumn
DeleteColumn
JoinLeft
JoinRight
JoinUp
JoinDown
SplitHorizontal
SplitVertical
AlignVerticalTop
AlignVerticalCenter
AlignVerticalBottom
AlignHorizontalLeft
AlignHorizontalRight
AlignHorizontalCenter
AlignHorizontalJustify
EditProperties

Properties

MayInsert
MayDelete
MayAppendRow
MayInsertRow
MayDeleteRow
MayAppendCol
MayInsertCol
MayDeleteCol
MayJoinLeft
MayJoinRight
MayJoinUp
MayJoinDown
MaySplitHorizontal

MaySplitVertical
MayAlignVertical
MayAlignHorizontal
MayEditProperties

2.16.1 AuthenticXMLTableCommands.AlignHorizontalCenter

Declaration: AlignHorizontalCenter()

Description
Sets the horizontal alignment attribute to "center" or clears the attribute if it was set to "center" before.

2.16.2 AuthenticXMLTableCommands.AlignHorizontalJustify

Declaration: AlignHorizontalJustify()

Description
Sets the horizontal alignment attribute to "justify" or clears the attribute if it was set to "justify" before.

2.16.3 AuthenticXMLTableCommands.AlignHorizontalLeft

Declaration: AlignHorizontalLeft()

Description
Sets the horizontal alignment attribute to "left" or clears the attribute if it was set to "left" before.

2.16.4 AuthenticXMLTableCommands.AlignHorizontalRight

Declaration: AlignHorizontalRight()

Description
Sets the horizontal alignment attribute to "right" or clears the attribute if it was set to "right" before.

2.16.5 AuthenticXMLTableCommands.AlignVerticalBottom

Declaration: AlignVerticalBottom()

Description
Sets the vertical alignment attribute to "bottom", or clears the attribute if it was set to "bottom" before.

2.16.6 AuthenticXMLTableCommands.AlignVerticalCenter

Declaration: AlignVerticalCenter()

Description
Sets the vertical alignment attribute to "center", or clears the attribute if it was set to "center" before.

2.16.7 AuthenticXMLTableCommands.AlignVerticalTop

Declaration: AlignVerticalTop()

Description
Sets the vertical alignment attribute to "top", or clears the attribute if it was set to "top" before.

2.16.8 AuthenticXMLTableCommands.AppendCol

Declaration: AppendCol()

Description
Appends a column to the current table.

2.16.9 AuthenticXMLTableCommands.AppendRow

Declaration: AppendRow()

Description
Appends a row to the current table.

2.16.10 AuthenticXMLTableCommands.Delete

Declaration: Delete()

Description
If the user confirmed the dialog the method deletes the currently selected table.

2.16.11 AuthenticXMLTableCommands.DeleteCol

Declaration: DeleteCol()

Description
The method deletes the currently selected column.
If there is only one column left and the user confirmed the dialog, the complete table is deleted.

2.16.12 AuthenticXMLTableCommands.DeleteRow

Declaration: DeleteRow()

Description
The method deletes the currently selected row.
If there is only one row left and the user confirmed the dialog, the complete table is deleted.

2.16.13 AuthenticXMLTableCommands.EditProperties

Declaration: EditProperties()

Description
Displays the properties dialog for the selected table/cell.

2.16.14 AuthenticXMLTableCommands.Insert

Declaration: Insert()

Description
Displays a dialog and inserts a new table into the existing XML.

2.16.15 AuthenticXMLTableCommands.InsertCol

Declaration: InsertCol()

Description
Inserts a new column before the currently selected column.

2.16.16 AuthenticXMLTableCommands.InsertRow

Declaration: InsertRow()

Description
Inserts a row before the currently selected row.

2.16.17 AuthenticXMLTableCommands.JoinDown

Declaration: JoinDown()

Description
Joins the current cell to the cell immediately below it.

2.16.18 AuthenticXMLTableCommands.JoinLeft

Declaration: JoinLeft()

Description
Joins the current cell with the cell immediately to its left.

2.16.19 AuthenticXMLTableCommands.JoinRight

Declaration: JoinRight()

Description
Joins the current cell with the cell immediately to its right.

2.16.20 AuthenticXMLTableCommands.JoinUp

Declaration: JoinUp()

Description
Joins the current cell to the cell immediately above it.

2.16.21 AuthenticXMLTableCommands.MayAlignHorizontal

Declaration: MayAlignHorizontal as Boolean

Description
True if the attributes for horizontal alignment can be set for the current cell.

2.16.22 AuthenticXMLTableCommands.MayAlignVertical

Declaration: MayAlignVertical as Boolean

Description
True if the attributes for vertical alignment can be set for the current cell.

2.16.23 AuthenticXMLTableCommands.MayAppendCol

Declaration: MayAppendCol as Boolean

Description
True if a column can be appended.

2.16.24 AuthenticXMLTableCommands.MayAppendRow

Declaration: MayAppendRow as Boolean

Description
True if a row can be appended.
Enter topic text here.

2.16.25 AuthenticXMLTableCommands.MayDelete

Declaration: MayDelete as Boolean

Description
The property is true if a table is selected.

2.16.26 AuthenticXMLTableCommands.MayDeleteCol

Declaration: MayDeleteCol as Boolean

Description
True if the current column can be deleted.

2.16.27 AuthenticXMLTableCommands.MayDeleteRow

Declaration: MayDeleteRow as Boolean

Description
True if the current row can be deleted.

2.16.28 AuthenticXMLTableCommands.MayEditProperties

Declaration: MayEditProperties as Boolean

Description
The property is true if the properties dialog is available for the selected table/cell.

2.16.29 AuthenticXMLTableCommands.MayInsert

Declaration: MayInsert as Boolean

Description
The property is true if the insertion of a new table is possible at the current selection.

2.16.30 AuthenticXMLTableCommands.MayInsertCol

Declaration: MayInsertCol as Boolean

Description
True if a column can be inserted.

2.16.31 AuthenticXMLTableCommands.MayInsertRow

Declaration: MayInsertRow as Boolean

Description
True if a row can be inserted.

2.16.32 AuthenticXMLTableCommands.MayJoinDown

Declaration: MayJoinDown as Boolean

Description
The property is true if the join down operation is possible for the currently selected cell.

2.16.33 AuthenticXMLTableCommands.MayJoinLeft

Declaration: MayJoinLeft as Boolean

Description
The property is true if the join left operation is possible for the currently selected cell.

2.16.34 AuthenticXMLTableCommands.MayJoinRight

Declaration: MayJoinRight as Boolean

Description
The property is true if the join right operation is possible for the currently selected cell.

2.16.35 AuthenticXMLTableCommands.MayJoinUp

Declaration: MayJoinUp as Boolean

Description
The property is true if the join up operation is possible for the currently selected cell.

2.16.36 AuthenticXMLTableCommands.MaySplitHorizontal

Declaration: MaySplitHorizontal as Boolean

Description
True if the current cell can be splitted horizontally.

2.16.37 AuthenticXMLTableCommands.MaySplitVertical

Declaration: MaySplitVertical as Boolean

Description
True if the current cell can be splitted vertically.

2.16.38 AuthenticXMLTableCommands.SplitHorizontal

Declaration: SplitHorizontal()

Description
The method splits the current cell horizontally.

2.16.39 AuthenticXMLTableCommands.SplitVertical

Declaration: SplitVertical()

Description
The method splits the current cell vertically.

2.17 XMLData

See also

Methods
InsertChild
AppendChild

EraseAllChildren
EraseCurrentChild

GetCurrentChild
GetFirstChild
GetNextChild

GetChild
GetChildKind

IsSameNode

HasChildrenKind

CountChildren
CountChildrenKind

Properties
Name
TextValue

HasChildren
MayHaveChildren

Kind
Parent

Description
You can use the XMLData interface to manipulate the content of the currently displayed XML. This interface is a lightweight COM counterpart of the implementation used inside the plug-in and XMLSpy itself.

To create a new XMLData object use the CreateChild() method of the plug-in interface.

2.17.1 XMLData.AppendChild

See also

Declaration: AppendChild(*pNewData* as XMLData)

Description
AppendChild appends pNewData as last child to the XMLData object. See also "Using XMLData".

Example

```
Dim objCurrentParent
Dim objNewChild

Set objNewChild = objPlugIn.CreateChild(spyXMLDataElement)
Set objCurrentParent = objPlugIn.XMLRoot

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

2.17.2 XMLData.CountChildren

See also

Declaration: CountChildren as long

Description
`CountChildren` gets the number of children.

Available with TypeLibrary version 1.6

Errors
1500 The XMLData object is no longer valid.

2.17.3 XMLData.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 1.6

Errors

1500 The XMLData object is no longer valid.

2.17.4 XMLData.EraseAllChildren

See also

Declaration: EraseAllChildren

Description
EraseAllChildren deletes all associated children of the XMLData object.

Example

The sample erases all elements of the active document.

```
Dim objCurrentParent

Set objCurrentParent = objPlugIn.XMLRoot
objCurrentParent.EraseAllChildren
```

2.17.5 XMLData.EraseChild

Method: `EraseChild` (Child node and new node as XMLData)

Description
Deletes the given child node.

Errors
1500 Invalid object.
1506 Invalid input xml
1510 Invalid parameter.

2.17.6 XMLData.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.

Example

This JavaScript example deletes all elements with the name "EraseMe". The code shows you, that it is possible to call EraseCurrentChild and GetNextChild inside the same loop to continue stepping through the child elements.

```
function DeleteXMLElements(objXMLData)
{
 if(objXMLData == null)
  return;

 if(objXMLData.HasChildren) {
  var objChild;
  objChild = objXMLData.GetFirstChild(-1);

  while(objChild){
   DeleteXMLElements(objChild);

   try{
    if(objChild.Name == "EraseMe")
     objXMLData.EraseCurrentChild();

    objChild = objXMLData.GetNextChild();
   }
   catch(Err) {
    objChild = null;
   }
  }
 }
}
```

2.17.7 XMLData.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 1.6

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

2.17.8 XMLData.GetChildAttribute

Method: `GetChildAttribute` (Name as string) as XMLData object (`NULL` on error)

Description
Retrieves the attribute having the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

2.17.9 XMLData.GetChildElement

Method: `GetChildElement` (Name as string, position as integer) as XMLData object (`NULL` on error)

Description
Retrieves the Nth child element with the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

2.17.10 XMLData.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 1.6

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

2.17.11 XMLData.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.

2.17.12 XMLData.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.

Example

See the example at XMLData.GetNextChild.

2.17.13 XMLData.GetNamespacePrefixForURI

Method: `GetNamespacePrefixForURI` (URI as string) Prefix as string

Description
Returns the namespace prefix of the supplied URI.

Errors
1500 Invalid object.
1510 Invalid parameter.

2.17.14 XMLData.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.

Example

```
On Error Resume Next
Set objParent = objPlugIn.XMLRoot

'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)
```

2.17.15 XMLData.GetTextValueXMLDecoded

Method: `GetTextValueXMLDecoded` as string

Description
Gets the decoded text value of the XML.

Errors
1500 Invalid object.
1510 Invalid parameter.

2.17.16 XMLData.HasChildren

See also

Declaration: HasChildren as Boolean

Description
The property is true, if the object is parent of other XMLData objects.

This property is read-only.

2.17.17 XMLData.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 1.6

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

2.17.18 XMLData.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).

2.17.19 XMLData.InsertChildAfter

Method: `InsertChildAfter` (Child node and new node as XMLData)

Description
Inserts a new XML node after the given node.

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

2.17.20 XMLData.InsertChildBefore

Method: `InsertChildBefore` (Child node and new node as XMLData)

Description
Inserts a new XML node before the given node.

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

2.17.21 XMLData.IsSameNode

See also

Declaration: IsSameNode(*pNodeToCompare* as XMLData) as Boolean

Description
Returns true if pNodeToCompare references to the same node as the object itself.

2.17.22 XMLData.Kind

See also

Declaration: Kind as SPYXMLDataKind

Description
Kind of this XMLData object.

This property is read-only.

2.17.23 XMLData.MayHaveChildren

See also

Declaration: MayHaveChildren as Boolean

Description
Tells if it is allowed to add children to this XMLData object.

This property is read-only.

2.17.24 XMLData.Name

See also

Declaration: Name as String

Description
Used to modify and to get the name of the XMLData object.

2.17.25 XMLData.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.

2.17.26 XMLData.SetTextValueXMLDecoded

Method: `SetTextValueXMLDecoded` (string value)

Description
Sets the encoded text value of the XML.

Errors
1500 Invalid object.
1513 Modification not allowed.

2.17.27 XMLData.TextValue

See also

Declaration: TextValue as String

Description
Used to modify and to get the text value of this XMLData object. See also "Using XMLData".

3 Enumerations

This section contains a listing and description of Authentic Browser enumerations.

3.1 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange objects.

Possible values:

spyAuthenticInsertAt	= 0
spyAuthenticApply	= 1
spyAuthenticClearSurr	= 2
spyAuthenticAppend	= 3
spyAuthenticInsertBefore	= 4
spyAuthenticRemove	= 5

3.2 SPYAuthenticCommand

Description
Enumeration of all available commands.

Possible values:

// CommandGroupMain
k_CommandSeparator = 0
k_CommandSave = 1
k_CommandPrint = 2
k_CommandPrintPreview = 3
k_CommandValidate = 4
k_CommandUndo = 5
k_CommandRedo = 6

// CommandGroupEdit
k_CommandEditCut = 7
k_CommandEditCopy = 8
k_CommandEditPaste = 9
k_CommandEditFind = 10
k_CommandEditRepeat = 11
k_CommandEditReplace = 12

// CommandGroupMarkup
k_CommandMarkupHide = 13
k_CommandMarkupLarge = 14

// CommandGroupRow
k_CommandRowAppend = 15
k_CommandRowInsert = 16
k_CommandRowDuplicate = 17
k_CommandRowMoveUp = 18
k_CommandRowMoveDown = 19
k_CommandRowDelete = 20

// CommandGroupXMLTables
k_CommandXMLTableInsert = 21
k_CommandXMLTableDelete = 22
k_CommandXMLTableAppendRow = 23
k_CommandXMLTableInsertRow = 24
k_CommandXMLTableDeleteRow = 25
k_CommandXMLTableAppendCol = 26
k_CommandXMLTableInsertCol = 27
k_CommandXMLTableDeleteCol = 28
k_CommandXMLTableJoinRight = 29
k_CommandXMLTableJoinLeft = 30
k_CommandXMLTableJoinUp = 31
k_CommandXMLTableJoinDown = 32
k_CommandXMLTableSplitHorz = 33
k_CommandXMLTableSplitVert = 34
k_CommandXMLTableVAlignTop = 35
k_CommandXMLTableVAlignCenter = 36
k_CommandXMLTableVAlignBottom = 37

```
        k_CommandXMLTableAlignLeft          = 38
        k_CommandXMLTableAlignCenter        = 39
        k_CommandXMLTableAlignRight         = 40
        k_CommandXMLTableAlignJustify       = 41
        k_CommandXMLTableEditProperties     = 42

    // since TypeLib 1.2
        k_CommandCheckSpelling              = 43
        k_CommandAbout                      = 44
        k_CommandPackageManagement          = 45
```

3.3 SPYAuthenticCommandGroup

Description
Groups to which a command can belong.

Possible values:

k_CommandGroupMain	= 0
k_CommandGroupEdit	= 1
k_CommandGroupMarkup	= 2
k_CommandGroupRow	= 3
k_CommandGroupXMLTables	= 4

3.4 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange objects.

Possible values:

spyAuthenticDocumentBegin = 0
spyAuthenticDocumentEnd = 1
spyAuthenticRangeBegin = 2
spyAuthenticRangeEnd = 3

3.5 SPYAuthenticElementActions

Description
Actions that can be used with GetAllowedElements.

Possible values:

k_ActionInsertAt	= 0
k_ActionApply	= 1
k_ActionClearSurr	= 2
k_ActionAppend	= 3
k_ActionInsertBefore	= 4
k_ActionRemove	= 5

3.6 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange and AuthenticView objects.

Possible values:

spyAuthenticChar	= 0
spyAuthenticWord	= 1
spyAuthenticLine	= 3
spyAuthenticParagraph	= 4
spyAuthenticTag	= 6
spyAuthenticDocument	= 8
spyAuthenticTable	= 9
spyAuthenticTableRow	= 10
spyAuthenticTableColumn	= 11

3.7 SPYAuthenticEntryHelperWindows

Description
Identification of the available entry helper windows.

Possible values:

k_Elements	= 1
k_Attributes	= 2
k_Entities	= 4

3.8 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup.

Possible values:

spyAuthenticMarkupHidden	= 0
spyAuthenticMarkupSmall	= 1
spyAuthenticMarkupLarge	= 2
spyAuthenticMarkupMixed	= 3

3.9 SPYAuthenticToolbarAllignment

Description
Values to specify toolbar alignment.

Possible values:

k_ToolbarAlignTop	= 0
k_ToolbarAlignLeft	= 1
k_ToolbarAlignBottom	= 2
k_ToolbarAlignRight	= 3

3.10 SPYAuthenticToolbarButtonState

Description
Authentic toolbar button states are given by the following enumeration:

Possible values:

authenticToolbarButtonDefault	= 0
authenticToolbarButtonEnabled	= 1
authenticToolbarButtonDisabled	= 2

3.11 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
spyXMLDataP	= 9
spyXMLDataDefDoctype	= 10
spyXMLDataDefExternalID	= 11
spyXMLDataDefElement	= 12
spyXMLDataDefAttlist	= 13
spyXMLDataDefEntity	= 14
spyXMLDataDefNotation	= 15
spyXMLDataKindsCount	= 16

Chapter 7

ASP.NET Web Applications

ASP.NET Web Applications

To ease deployment of **Altova® Authentic® Browser Edition** we provide an ASP.NET Server Control that completely integrates with Visual Studio .NET and allows ASP.NET developers to drag-and-drop **Authentic Browser** onto their web forms.

To use the ASP.NET Server Control

1. Download the **ASP.NET Server Control for Authentic Browser Plug-in** from the Altova website.
2. Start Visual Studio and register the assembly (the downloaded package) in Visual Studio .NET by adding it to the Toolbox. You could do this by selecting the **Tools | Choose Toolbox Items** command, and, in the Toolbox Items dialog, selecting the .NET Framework Components, and then browsing for the assembly (`Altova:Authentic.WebControls.dll`).
3. The `AuthenticDocumentView` control appears in the General pane of the Toolbox. Drag this control into the Design View and resize or modify as required. (The control will invoke Authentic Browser on the client-side web page.)
4. Add the plugin-related files (Schema, SPS, XML, image files, etc) to your project. One way to do this is to add a new folder to your Visual Studio project via Solution Explorer and add the plugin-related files to the new project folder. For each of these plugin-related files, set the following properties: ***Build Action =*** *Content* and ***Copy to Output Directory =*** *Yes, when newer*.
5. Set the following properties for the `AuthenticDocumentView` control: (i) *Select the Trusted/Untrusted version*; (ii) Add the SPS-related resources (*SchemaDataURL, SPSDataURL, XMLDataURL*); (iii) in the case of the Enterprise edition, add the license information.
6. For IIS version 7.0 or newer, add a static content handler for the SPS file. Do this by adding the following lines to your `web.config` file:

```
<system.webServer>
    <staticContent>
        <mimeMap fileExtension=".sps" mimeType="text/xml" />
    </staticContent>
</system.webServer>
```

Example ASP.NET project

An example ASP.NET web application project for Visual Studio 2008, called `QuickStartAppl`, has been included with the ASP.NET Server Control package. It is located in the `Example` folder of the unzipped package.

Chapter 8

License Information

License Information

This section contains:

- Information about the intellectual property rights related to this software product
- The End User License Agreement for Authentict 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.

1 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 Altova End User License Agreement for Authentic

The license agreement below applies to the following Altova products:

Authentic® Enterprise Edition

THIS IS A LEGAL DOCUMENT -- RETAIN FOR YOUR RECORDS

ALTOVA® END-USER LICENSE AGREEMENT

Licensor:
Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Austria

Important - Read Carefully. Notice to User:

This End User License Agreement ("Agreement") is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software ("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 Software, you agree to be bound by the terms of this Agreement 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 Software, and you must destroy any downloaded copies of the Software in your possession or control. You may print a copy of this Agreement as part of the installation process at the time of acceptance. Alternatively, a copy of this Agreement may be found at http://www.altova.com/license_agreements.html and a copy of the Privacy Policy may be found at http://www.altova.com/privacy.

1. SOFTWARE LICENSE

(a) License Grant for Authentic Desktop Enterprise Edition ("ADEE"). Upon your acceptance of this Agreement, 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)(ii) and (iii), 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)(ii) and (iii) below Altova makes no warranties or representations about the performance of Software in a terminal server environment and the foregoing are expressly excluded from the limited warranty in Section 4 hereof and technical support is not available with respect to issues arising from use in such an environment.

(i) Named-Use. 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) Concurrent Use in Same Local Area Network (LAN). 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) Concurrent Use in Virtual Environment. 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 ten (10) 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 Software in a terminal server, application virtualization

session, or virtual machine environment and the foregoing are expressly excluded from the limited warranty in Section 5 hereof and technical support is not available with respect to issues arising from use in such environments.

(iv) Key Codes for ADEE. 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.

(b) License Grant for Authentic Browser-Plugin Enterprise Edition ("ABEE"). Upon your acceptance of this Agreement, 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 ABEE software license. If you have purchased an ABEE software license then under the terms of the Agreement, 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.

(i) Key Codes for ABEE. 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.**

(ii) ABEE Software Specific Restrictions. In addition to the restrictions and obligations provided in other sections of this Agreement 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 the Altova End User License Agreement, (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).

(c) Backup and Archival Copies. You may make one (1) backup and one (1) archival copy of the 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 Software. You may not transfer the rights to a backup or archival copy unless you transfer all rights in the Software as provided in this Agreement.

(d) 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 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.

(e) Title. Title to the Software is not transferred to you. Ownership of all copies of the 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 Agreement. All rights not specifically granted in this Agreement are reserved by Altova.

(f) Reverse Engineering. Except and to the limited extent as may be otherwise specifically provided by applicable law in the European Union, 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 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 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.

(g) Other Restrictions. You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software to third parties except to the limited extent set forth in Section 3 or as otherwise expressly provided. You may not copy the Software except as expressly set forth above, and any copies that you are permitted to make pursuant to this Agreement must contain the same copyright, patent and other intellectual property markings that appear on or in the Software. You may not modify, adapt or translate the Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software to be placed in the public domain; or use the Software in any computer environment not specified in this Agreement. You may not permit any use of or access to the Software by any third party in connection with a commercial service offering, such as for a cloud-based or web-based SaaS offering.

You will comply with applicable law and Altova's instructions regarding the use of the Software. You agree to notify your employees and agents who may have access to the Software of the restrictions contained in this Agreement and to ensure their compliance with these restrictions.

(h) NO GUARANTEE. THE 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 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 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 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

THIRD PARTY CLAIMS, DEMANDS, OR SUITS THAT ARE BASED UPON THE ACCURACY AND ADEQUACY OF THE SOFTWARE IN YOUR USE OR ANY DATA GENERATED BY THE SOFTWARE IN YOUR USE.

2. INTELLECTUAL PROPERTY RIGHTS

You acknowledge that the 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. You acknowledge that 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. You will take no actions which adversely affect Altova's intellectual property rights in the 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 Software, and such use of any trademark does not give you any right of ownership in that trademark. Altova®, XMLSpy®, Authentic®, StyleVision®, MapForce®, UModel®, DatabaseSpy®, DiffDog®, SchemaAgent®, SemanticWorks®, MissionKit®, Markup Your Mind®, Nanonull™, RaptorXML™, RaptorXML Server™, RaptorXML +XBRL Server™, Powered By RaptorXML™, FlowForce Server™, StyleVision Server™, and MapForce Server™ are trademarks of Altova GmbH. (pending or registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows XP, Windows Vista, Windows 7, and Windows 8 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 Agreement does not grant you any intellectual property rights in the Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

3. LIMITED TRANSFER RIGHTS

Notwithstanding the foregoing, you may transfer all your rights to use the Software to another person or legal entity provided that: (a) you also transfer this Agreement, the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, updates and prior versions, and all copies of font software converted into other formats, to such person or entity; (b) you retain no copies, including backups and copies stored on a computer; (c) the receiving party secures a personalized key code from Altova; and (d) the receiving party accepts the terms and conditions of this Agreement and any other terms and conditions upon which you legally purchased a license to the Software. Notwithstanding the foregoing, you may not transfer education, pre-release, or not-for-resale copies of the Software.

4. PRE-RELEASE AND EVALUATION PRODUCT ADDITIONAL TERMS

If the product you have received with this license is pre-commercial release or beta Software ("Pre-release Software"), then this Section 4 applies. In addition, this section applies to all evaluation and/or demonstration copies of Altova software ("Evaluation Software") and continues in effect until you purchase a license. To the extent that any provision in this section is in conflict with any other term or condition in this Agreement, this section shall supersede such other term(s) and condition(s) with respect to the Pre-release and/or Evaluation Software, but only to the extent necessary to resolve the conflict. You acknowledge that the Pre-release Software is a pre-release

version, does not represent final product from Altova, and may contain bugs, errors and other problems that could cause system or other failures and data loss. CONSEQUENTLY, THE PRE-RELEASE AND/OR 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. Upon such expiration date, your license will expire unless otherwise extended. Your license to use any output created with the Evaluation Software that contains generated program code (including Unrestricted Source Code) such as Java, C++, C, VB.NET or XSLT and associated project files and build scripts as well as generated XML, XML Schemas, documentation, UML diagrams, and database structures terminates automatically upon the expiration of the designated evaluation period but the license to use such output is revived upon your purchase of a license for the Software that you evaluated and used to create such output. Access to any files created with the Evaluation Software is entirely at your risk. You acknowledge that Altova has not promised or guaranteed to you that Pre-release Software will be announced or made available to anyone in the future, that Altova has no express or implied obligation to you to announce or introduce the Pre-release Software, and that Altova may not introduce a product similar to or compatible with the Pre-release Software. Accordingly, you acknowledge that any research or development that you perform regarding the Pre-release Software or any product associated with the Pre-release Software is done entirely at your own risk. During the term of this Agreement, if requested by Altova, you will provide feedback to Altova regarding testing and use of the Pre-release Software, including error or bug reports. If you have been provided the Pre-release Software pursuant to a separate written agreement, your use of the Software is governed by such agreement. You may not sublicense, lease, loan, rent, distribute or otherwise transfer the Pre-release Software. Upon receipt of a later unreleased version of the Pre-release Software or release by Altova of a publicly released commercial version of the Software, whether as a stand-alone product or as part of a larger product, you agree to return or destroy all earlier Pre-release Software received from Altova and to abide by the terms of the license agreement for any such later versions of the Pre-release Software.

5. LIMITED WARRANTY AND LIMITATION OF LIABILITY

(a) Limited Warranty and Customer Remedies. Altova warrants to the person or entity that first purchases a license for use of the Software pursuant to the terms of this Agreement 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 6 of this Agreement. 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 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 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 Software has resulted from accident, abuse, misapplication, abnormal use, Trojan horse, virus, or any other malicious external code. Any replacement 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 and/or Pre-release Software.

(b) No Other Warranties and Disclaimer. THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA OR ITS

SUPPLIER'S 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.

(c) Limitation of Liability. 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 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 AGREEMENT SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE 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 Agreement between Altova and you.

(d) Infringement Claims. 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 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 Section 5 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 Software. If the 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 Software or obtain a license to

continue using the 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 Agreement 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 situations where the alleged infringement, whether patent or otherwise, is the result of a combination of the Altova software and additional elements supplied by you.

6. SUPPORT AND MAINTENANCE

Altova offers multiple optional "Support & Maintenance Package(s)" ("SMP") for the version of Software product edition that you have licensed, which you may elect to purchase in addition to your Software license. The Support Period, hereinafter defined, covered by such SMP shall be delineated at such time as you elect to purchase a SMP. 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:

(a) 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 6(a), 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.

(b) If you have purchased SMP, then solely for the duration of its delineated Support Period, you are eligible to receive the version of the 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 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 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.

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 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 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 Software and Altova will not provide you with generic consultation, assistance, or advice under any circumstances.

Updating Software may require the updating of software not covered by this Agreement before installation. Updates of the operating system and application software not specifically covered by this Agreement are your responsibility and will not be provided by Altova under this Agreement. Altova's obligations under this Section 6 are contingent upon your proper use of the Software and your compliance with the terms and conditions of this Agreement at all times. Altova shall be under no obligation to provide the above technical support if, in Altova's opinion, the Software has failed due to the following conditions: (i) damage caused by the relocation of the Software to another location or CPU; (ii) alterations, modifications or attempts to change the Software without Altova's written approval; (iii) causes external to the Software, such as natural disasters, the failure or fluctuation of electrical power, or computer equipment failure; (iv) your failure to maintain the Software at Altova's specified release level; or (v) use of the Software with other software without Altova's prior written approval. It will be your sole responsibility to: (i) comply with all Altova-specified operating and troubleshooting procedures and then notify Altova immediately of Software malfunction and provide Altova with complete information thereof; (ii) provide for the security of your confidential information; (iii) establish and maintain backup systems and procedures necessary to reconstruct lost or altered files, data or programs.

7. SOFTWARE ACTIVATION, UPDATES AND LICENSE METERING

(a) License Metering. The Software includes a built-in license metering module that is designed to assist you with monitoring license compliance in small local area networks (LAN). The metering module attempts to communicate with other machines on your local area network (LAN). You permit Altova to use your internal network for license monitoring for this purpose. This license metering module may be used to assist with your license compliance but should not be the sole method. Should your firewall settings block said communications, you must deploy an accurate means of monitoring usage by the end user and preventing users from using the Software more than the Permitted Number.

(b) License Compliance Monitoring. You are required to utilize a process or tool to ensure that the Permitted Number is not exceeded. Without prejudice or waiver of any potential violations of the Agreement, Altova may provide you with additional compliance tools should you be unable to accurately account for license usage within your organization. If provided with such a tool by Altova, you (a) are required to use it in order to comply with the terms of this Agreement and (b) permit Altova to use your internal network for license monitoring and metering and to generate compliance reports that are communicated to Altova from time to time.

(c) Software Activation. The Software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration, use, or update to an Altova Master License Server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova Master 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 Software violates Altova's intellectual property rights as well as the terms of this Agreement. You agree that efforts to circumvent or disable Altova's copyright protection mechanisms, the license management mechanism, or the Altova Master License Server violate Altova's intellectual property rights as well as the terms of this Agreement. 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.

(d) LiveUpdate. 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.

(e) Use of Data. 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 Agreement. By your acceptance of the terms of this Agreement and/or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Agreement and/or the Privacy Policy. Altova has the right in its sole discretion to amend this provision of the Agreement 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.

(f) Audit Rights. You agree that Altova may audit your use of the Software for compliance with the terms of this Agreement at any time, upon reasonable notice. In the event that such audit reveals any use of the Software by you other than in full compliance with the terms of this Agreement, you shall reimburse Altova for all reasonable expenses related to such audit in addition to any other liabilities you may incur as a result of such non-compliance.

(g) 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 Agreement, you consent to the transfer of all such information to the United States and the processing of that information as described in this Agreement and the Privacy Policy.

8. TERM AND TERMINATION

This Agreement may be terminated (a) by your giving Altova written notice of termination; (b) by Altova, at its option, giving you written notice of termination if you commit a breach of this Agreement and fail to cure such breach within ten (10) days after notice from Altova; or (c) at the request of an authorized Altova reseller in the event that you fail to make your license payment or other monies due and payable. This Agreement automatically terminates upon the expiration of the ABEE License Term. In addition the Agreement governing your use of a previous version of the Software that you have upgraded or updated is terminated upon your acceptance of the terms and conditions of the Agreement accompanying such upgrade or update. Upon any termination of the Agreement, you must cease all use of the Software that this Agreement governs, 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 Software remain in your possession or control. The terms and conditions set forth in Sections 1(c)-(h), 2, 5, 7, 9, 11-14 survive termination as applicable.

9. RESTRICTED RIGHTS NOTICE AND EXPORT RESTRICTIONS

The 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 Agreement. Manufacturer is Altova GmbH, Rudolfsplatz 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Software or Documentation except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the 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.

10. U.S. GOVERNMENT ENTITIES

Notwithstanding the foregoing, if you are an agency, instrumentality or department of the federal government of the United States, then this Agreement shall be governed in accordance with the laws of the United States of America, and in the absence of applicable federal law, the laws of the Commonwealth of Massachusetts will apply. Further, and notwithstanding anything to the contrary in this Agreement (including but not limited to Section 5 (Indemnification)), all claims, demands, complaints and disputes will be subject to the Contract Disputes Act (41 U.S.C. §§7101 et seq.), the Tucker Act (28 U.S.C. §1346(a) and §1491), or the Federal Tort Claims Act (28 U.S.C. §§1346(b), 2401-2402, 2671-2672, 2674-2680), FAR 1.601(a) and 43.102 (Contract Modifications); FAR 12.302(b), as applicable, or other applicable governing authority. For the avoidance of doubt, if you are an agency, instrumentality, or department of the federal, state or local government of the U.S. or a U.S. public and accredited educational institution, then your indemnification obligations are only applicable to the extent they would not cause you to violate any applicable law (e.g., the Anti-Deficiency Act), and you have any legally required authorization or authorizing statute.

11. THIRD PARTY SOFTWARE

The 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 Agreement. By accepting this Agreement, you are also accepting the additional terms and conditions, if any, set forth therein.

12. JURISDICTION, CHOICE OF LAW, AND VENUE

If you are located in the European Union and are using the Software in the European Union and not in the United States, then this Agreement 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 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.

If you are located in the United States or are using the Software in the United States then this Agreement 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 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.

If you are located outside of the European Union or the United States and are not using the Software in the United States, then this Agreement 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 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 Agreement 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.

13. TRANSLATIONS

Where Altova has provided you with a foreign translation of the English language version, you agree that the translation is provided for your convenience only and that the English language version will control. If there is any contradiction between the English language version and a translation, then the English language version shall take precedence.

14. GENERAL PROVISIONS

This Agreement 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 Agreement 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 Agreement 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 Agreement. This Agreement 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 Agreement 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 Agreement. If, for any reason, any provision of this Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Agreement, and this Agreement 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.

Last updated: 2016/10/07

Index

A

B

C

R

S

T

U

X