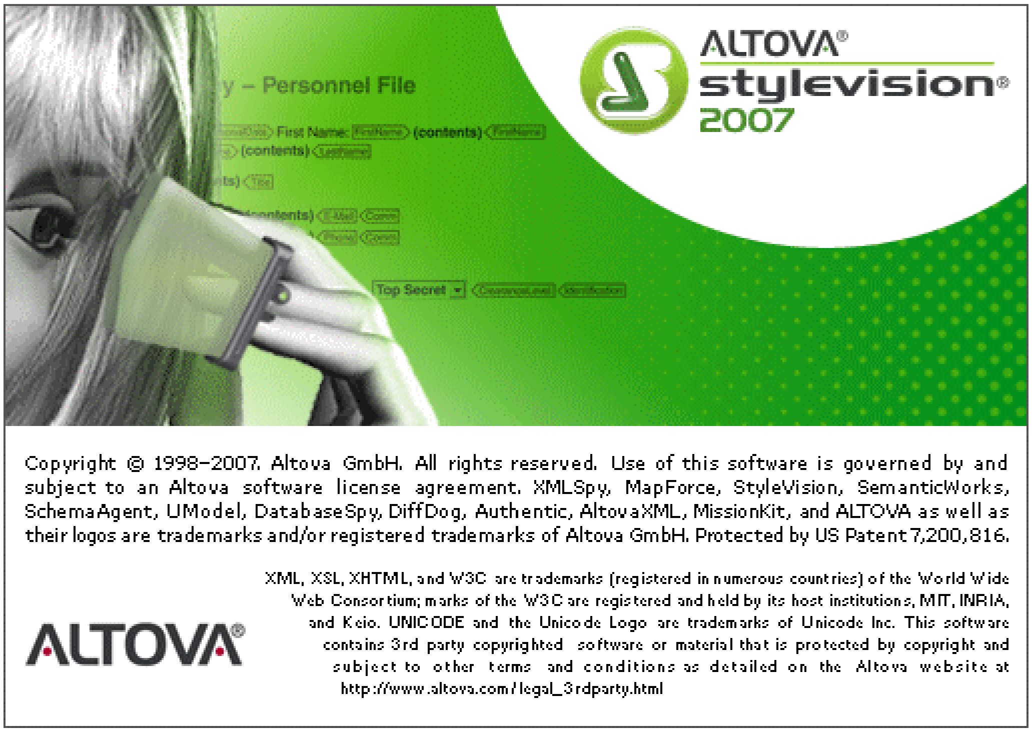
User and Reference Manual



Altova StyleVision 2007 User & Reference Manual

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

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

Published: 2007

Table of Contents

Chapter 1

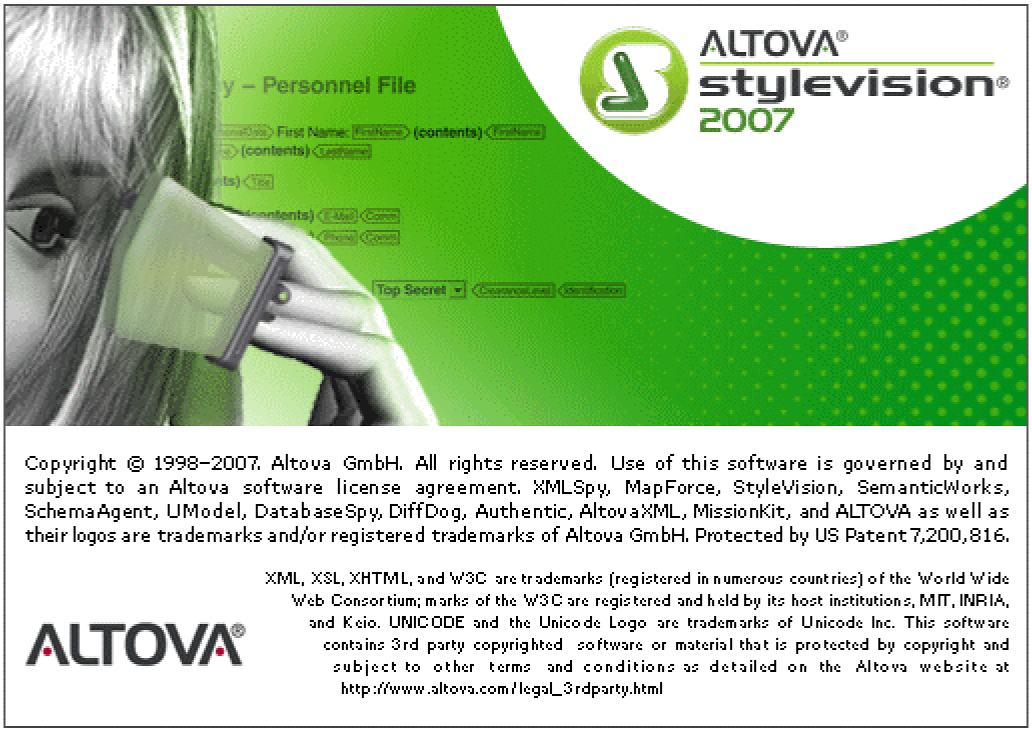
Altova StyleVision 2007

1 Altova StyleVision 2007

Altova® StyleVision® 2007 Professional Edition is an application for graphically designing and editing StyleVision Power Stylesheets. A StyleVision Power Stylesheet (SPS) can be used for the following purposes:

- To control a graphical WYSIWYG view of **XML documents in Authentic View**, which is an XML document editor available in the following Altova products: Altova XMLSpy, Altova StyleVision, Altova Authentic Desktop, and Altova Authentic Browser.
- To enable the editing of **databases (DBs) via Authentic View** and to generate database reports in HTML format.
- To generate **XSLT stylesheets** based on the SPS design. (Both XSLT 1.0 and XSLT 2.0 are supported.) The XSLT stylesheets can be used outside StyleVision to transform XML documents into outputs such as HTML.
- To generate, directly from within StyleVision, **HTML output** from an XML document. In the case of DB-based SPSs, StyleVision can additionally generate, for each SPS, an XML Schema based on the DB and an XML instance document that adheres to this schema and contains data from the DB.

StyleVision also enables you to import an HTML document and create an XML document from it.



Chapter 2

About this Documentation

2 About this Documentation

This documentation is the user manual delivered with StyleVision. It is available as the built-in Help system of StyleVision, can be viewed online at the Altova website, and can also be downloaded from there as a PDF, which you can print.

The user manual is organized into the following sections:

- An introduction, which explains what an SPS is and introduces the main features and concepts of StyleVision.
- A description of the user interface, which provides an overview of the StyleVision GUI.
- A tutorial section, which is a hands-on exercise to familiarize you with StyleVision features.
- High-Level Procedures, which describes usage at a broad level: for example, schema sources, interaction with databases, and command line options.
- Content Editing Procedures, which explains how static and dynamic components are created and edited in the SPS.
- Presentation Procedures, which explains how SPS components are formatted and laid out.
- Additional Editing Procedures, which explains how a range of additional SPS editing features are used.
- Authentic View, which describes how XML documents are edited in Authentic View. The StyleVision GUI contains an Authentic View preview tab, in which you can immediately test the Authentic View output.
- HTML Import, which describes how an XML Schema and XML document can be created that is based on the structure and content of an HTML document.
- A reference section containing descriptions of all symbols and commands used in StyleVision.
- Appendices containing information about the Altova XSLT Engine information and the conversion of DB datatypes to XML Schema datatypes; technical data about StyleVision; and license information.

How to use
We suggest you read the Introduction and User Interface sections first in order to get an overview of StyleVision features and general usage. Doing the tutorial next would provide hands-on experience of creating an SPS. The Procedures sections (High-Level Procedures, Content Editing Procedures, Presentation Procedures, and Additional Editing Procedures) provide detailed descriptions of how to use various StyleVision features. For subsequent reference, the Reference section provides a concise description of all toolbar icon, design symbols, and menu commands, organized according to toolbar and menu. The Authentic View and HTML Import sections provide, respectively, information about editing in Authentic View and converting HTML documents to XML and XML Schema documents.

Support options
Should you have any question or problem related to StyleVision, the following support options are available:

1. Check the Help file (this documentation). The Help file contains a full text-search feature, besides being fully indexed.
2. Check the FAQs and Discussion Forum at the Altova Website.
3. Contact Altova's Support Center.

Commonly used abbreviations
The following abbreviations are used frequently in this documentation:

- **SPS**: StyleVision Power Stylesheet
- **DB**: Database
- **CSS**: Cascading Style Sheets
- **FAQ**: Frequently Asked Questions

Chapter 3

Introduction

3 Introduction

This section introduces you to **Altova® StyleVision® 2007**. It consists of the following sub-sections:

- What Is an SPS?, which explains the role of an SPS in an XML environment and with respect to StyleVision.
- Product Features, which provides an overview of the key features of StyleVision.
- Terminology, which lists terms used in the StyleVision user interface and in this documentation.
- Setting up StyleVision, which describes how StyleVision is to be correctly set up.

3.1 What Is an SPS?

A StyleVision Power Stylesheet (or SPS) is an extended XSLT stylesheet which is used:

1. to control the display and entry of data in the Authentic View of XML documents and databases (DBs); and
2. to specify the output design of an XML document transformation.

An SPS is saved with the file extension `.sps`.

Design of the SPS
An SPS is created graphically in StyleVision. It is based on a schema (DTD or XML Schema); if the SPS is to be used with a DB, it is based on an XML Schema generated automatically by StyleVision from the DB structure. The design of the SPS is flexible. It can contain dynamic and static content. The dynamic content is the data in one XML document or DB. The static content is content entered directly in the SPS. Dynamic content can be included in the design either as straight text or within components such as input fields, combo boxes, and tables. Additionally, dynamic content can be manipulated (using Auto-Calculations) and can be displayed if certain conditions in the source document are fulfilled. Different pieces of content can be placed at various and multiple locations in the SPS. Also, the SPS can contain various other components, such as images, hyperlinks, and JavaScript functions. Each component of the SPS can then be formatted for presentation as required.

The SPS and Authentic View
When a finished SPS is associated with an XML document or DB, that XML document or DB can be edited in Authentic View. Authentic View is an ideal solution for enabling the distributed and graphical editing of an XML document or DB. Multiple users can edit an XML document or DB in the graphical user interface presented by Authentic View. In StyleVision, as you design an SPS, you can preview and test the SPS (in the Authentic View tab for that SPS). For a detailed description of how SPSs work with Authentic View, see SPS and Authentic View.

The SPS and XSLT stylesheets
After you have completed designing the SPS, you can generate XSLT stylesheets based on the design you have created. StyleVision supports both XSLT 1.0 and XSLT 2.0, and from a single SPS, you can generate XSLT stylesheets for HTML, RTF, and XSL-FO output (*Enterprise edition only; in Professional and Standard Editions, only HTML output is supported*). The generated XSLT stylesheets can be used in external transformations to transform XML documents based on the same schema as the SPS from which the XSLT stylesheet was generated. For more information about procedures used with XSLT stylesheets, see the section Generated Files.

The SPS and output
You can also use StyleVision to directly generate output (*HTML, RTF, XSL-FO, and PDF in Enterprise Edition; HTML in Professional and Standard Editions*). The tabs for Output Views display the output for the active SPS document directly in the StyleVision GUI. The required output can also be generated to file: (i) from within the GUI via the **File | Save Generated Files** command; or by invoking StyleVision via the command line. These procedures are described in more detail in the High-Level Procedures section.

Authentic View in Altova Products

Authentic View is a graphical XML document editor available in the following Altova products:

- Altova XMLSpy
- Altova Authentic Desktop
- Altova Authentic Browser
- Altova StyleVision

3.2 Product Features

The main product features of StyleVision are listed below in two groups:

- General product features, which are high-level features
- SPS design features, which are features related to the design of the SPS

General product features

Given below is a list of the main high-level features of StyleVision.

- Multiple SPS designs can be open simultaneously, with one being active at any given time. Each SPS design is shown in a separate tab.
- Template filters allow you to customize the display of the design document. With this feature you can disable the display of templates that are not currently being edited, thus increasing editing efficiency.
- While designing the SPS, Authentic View, output views and stylesheets can be displayed by clicking the respective tabs. This enables you to quickly preview the output and XSLT code, and test Authentic View features.
- When an SPS is associated with an XML source document or source DB, the source document can be edited directly in the Authentic View of StyleVision.
- DB reports can either be viewed in StyleVision or saved as an HTML file.
- IBM DB2 databases, which contain XML columns, are supported.
- Both XSLT versions (1.0 and 2.0) are supported. XSLT 2.0 provides powerful data access and manipulation features.
- In the Enterprise Edition, multiple output formats (HTML, RTF, and PDF) are generated from a single SPS design.
- Conditions can be set on SPS components to process them differently for different outputs. With this level of granularity, different outputs can be flexibly structured to take in the requirements of that particular output.
- Both XSLT files and output files can be generated and saved, either directly from within the GUI or by calling StyleVision from the command line.
- HTML documents can be converted to XML.

SPS design features

Given below is a list of the main StyleVision features specific to designing the SPS.

- The SPS can contain static text, which you enter in the SPS, and dynamic text, which is selected from the source document .
- Dynamic content is inserted in the design by dragging-and-dropping nodes from the schema source.
- Dynamic content can be inserted as text, or in the form of a data-entry device (such as an input field or combo box). When inserted as a data-entry device such as a combo box, additional possibilities are available. For example, the value of the node can be selected (by the Authentic View user) from a list of enumerations.
- The structure of the design is specified and controlled in a single main template. This structure can be modified by optional templates for individual elements—known as global templates because they can be applied globally for that element.
- Design Fragments enable the modularization and re-use of templates in a manner similar to the way functions are used.
- A common feature of XML documents is the repeating data structure. For example, an office department typically has several employees. The data for each employee would be stored in a data structure which is repeated for each employee. In the SPS, the processing for each such data structure is defined once and applied to each relevant node in turn (the employee node in our example).

- Multiple tables of contents can be inserted in XSLT 2.0 SPSs.
- Repeating data structures can also be inserted as dynamic tables. This provides looping in a structured, table format, with each loop through the data structure producing a row (or, if required, a column) of the table.
- A repeating element can be sorted on one or more sort-keys you select, and the sorted element set is sent to the output (HTML).
- The conditional templates feature enables one of a set of templates to be processed according to what conditions in the XML document or system environment are fulfilled. This enables processing that is conditional on information contained in the source document or that cannot be known to the SPS document creator at the time of creation (for example, the date of processing). The available conditions are those that can be tested using XPath 1.0 or XPath 2.0 expressions.
- Auto-Calculations enable you to manipulate data from the source document/s and to display the result. This is useful, when you wish to perform calculations on numbers (for example, sum the prices in an invoice), manipulate strings (for example, change hyphens to slashes), generate content, etc. The available manipulations are those that can be effected using XPath 1.0 or XPath 2.0 expressions.
- When data is edited in Authentic View, the result of Auto-Calculations can also be passed to a node in the source document. This procedure is referred to as updating the XML node (with the value of the Auto-Calculation).
- Additional validation enables individual XML document nodes to be validated (additionally to schema validation) against an XPath expression defined for that node. In this way, Authentic View users can be alerted when the data they enter is invalid; a customized error message for the node can indicate the problem.
- Images can be inserted in the design. The URI for the image can be static (entered in the SPS), or dynamic (taken from a node in the source document), or a combination of both static and dynamic parts.
- Two types of lists can be created: static and dynamic. In a static list, each list item is defined in the SPS. In a dynamic list, a node is created as a list item; the values of all instances of that node are created as the items of the list.
- Static and dynamic links can be inserted in the design. The target URI can be static (entered in the SPS), or dynamic (taken from a node in the source document), or a combination of both static and dynamic parts.
- Static bookmarks can be inserted. These serve as anchors that can be linked to with a hyperlink.
- Parameters can be declared globally for the entire SPS. A parameter is declared with a name and a string value, and can be used in XPath expressions in the SPS. The parameter value you declare is the default value and can be overridden by a value passed from the command line.
- With the Input Formatting feature, the contents of numeric XML Schema datatype nodes can be formatted as required for Authentic View display and, in the case of some formats, optionally for output display. Input Formatting can also be used to format the result of an Auto-Calculation.
- JavaScript functions can be used in the SPS to provide user-defined functionality for Authentic View and HTML output.
- A number of predefined HTML formats are available via the GUI and can be applied to individual SPS components.
- A large number of CSS text formatting and layout properties can be applied to individual SPS components via the Styles entry helper.
- Additionally, CSS styles can be defined for HTML selectors at the global level of an SPS and in external CSS stylesheets. These style rules will be applied to Authentic View and HTML output, thus providing considerable formatting and layout flexibility.
- Styles can also be assigned using XPath expressions. This enables style property values to be selected from XML documents and to set property values conditionally.

3.3 Terminology

This section lists terms used in the StyleVision GUI and in this documentation. Terms are organized into the groups listed below, and within each group, they are listed alphabetically.

- Altova product-related terms
- General XML terms and concepts
- XSLT and XPath terms
- StyleVision-specific terms

Note: If a link below points to a term already in the viewport, the screen display will not change when the link is clicked; in such cases, look for the target term in the current display.

Altova product-related terms

A list of terms that relate to Altova products.

Authentic View An XML document editor view available in the following Altova products: Altova XMLSpy; Altova StyleVision; Altova Authentic Desktop; Altova Authentic Browser. For more details about Authentic View and Altova products, visit the Altova website.

SPS The abbreviated form of StyleVision Power Stylesheet, it is used throughout this documentation to refer to the design document created in StyleVision and saved as a file with the `.sps` extension. For a detailed description, see What Is an SPS?.

General XML terms

Definitions of certain XML terms as used in this documentation.

schema A schema *(with lowercase 's')* refers to any type of schema. Schemas supported by StyleVision are XML Schema (*capitalized*) and DTD.

XML Schema In this documentation, XML Schema (*capitalized*) is used to refer to schemas that are compliant with the W3C's XML Schema specification. XML Schema is considered to be a subset of all schemas (*lowercased*).

URI and URL In this documentation, the more general URI is used exclusively—even when the identifier has only a "locator" aspect, and even for identifiers that use the `http` scheme.

XSLT and XPath terms

There have been changes in terminology from XSLT 1.0 and XPath 1.0 to XSLT 2.0 and XPath 2.0. For example, what was the root node in XPath 1.0 is the document node in XPath 2.0. In this documentation, we use XSLT 2.0 and XPath 2.0 terminology.

absolute XPath A path expression that starts at the root node of the tree containing the context node. In StyleVision, when entering path expressions in dialogs, the expression can be entered as an absolute path if you check the Absolute XPath check box in the dialog. If this check box is unchecked, the path is relative to the context node.

context item / context node The context item is the item (node or string value) relative to which an expression is evaluated. A context node is a context item that is a node. The context item can change within an expression, for example, with each location step, or within a filter expression (predicate).

current node The current node is the node being currently processed. The current node is the same as the context node in expressions that do not have sub-expressions. But where there are sub-expressions, the context node may change. Note that the `current()` function is an XSLT function, not an XPath function, and cannot therefore be used in StyleVision's Auto-Calculations and Conditional Templates. To select the current node in an expression use the `for` expression of XPath 2.0.

document element In a well-formed XML document, the outermost element is known as the document element. It is a child of the document node, and, in a well-formed XML document, there is only one document element. In the GUI the document element is referred to as the root element.

document node The document node represents and contains the entire document. It is the root node of the tree representation of the document, and it is represented in an XPath expression as: `'/'` . In the Schema Sources window of StyleVision, it is represented by the legend: `'/ Root elements'` .

StyleVision-specific terms

Terms that refer to StyleVision mechanisms, concepts, and components.

dynamic items Items that originate in XML data sources. Dynamic items may be text, tables, and lists; also images and hyperlinks (when the URIs are dynamic).

global element An element in the Global Elements list in the Schema Sources window. In an XML Schema, all elements defined as global elements will be listed in the Global Elements list. In a DTD, all elements are global elements and are listed in the Global Elements list. Global templates can be defined only for global elements.

global template A global template may be defined for a global element. Once defined, a global template can be used for that element wherever that element occurs in the document. Alternatively to the global template, processing for a global element may be defined in a local template.

local template A local template is the template that defines how an element (global or non-global) is processed within the main template. The local template applies to that particular occurrence of the element in the main template. Instead of the local template, a global template can be applied to a given occurrence of an element in the main template.

main schema One of the assigned schema sources is designated the main schema; the document node of the Working XML File associated with the main schema is used as the starting point for the main template.

main template The main entry-point template. In StyleVision, this template matches the document element and is the first to be evaluated by the XSLT processor. In the Schema Sources window, it is listed as the child of the document node. The main template defines the basic output document structure and defines how the input document/s are to be processed. It can contain local templates and can reference global templates.

output The output produced by processing an XML document with an XSLT stylesheet. Output files that can be generated by StyleVision would be HTML format. Authentic View is not considered an output, and is referred to separately as Authentic View. XSLT stylesheets generated by StyleVision are also not considered output and are referred to separately as XSLT stylesheets.

static items Items that originate in the SPS and not in XML data sources. Static items may be text, tables, and lists; also images, hyperlinks, and bookmarks (when the URIs are static).

SPS component An SPS component can be: (i) a schema node (for example, an element node); (ii) a static SPS component such as an Auto-Calculation or a text string; or (iii) a predefined format (represented in the SPS by its start and end tags).

template Defined loosely as a set of instructions for processing a node or group of nodes.

Template XML File A Template XML File is assigned to an SPS in StyleVision (Enterprise and Professional editions). It is an XML file that provides the starting data of a new XML document created with a given SPS when that SPS is opened in Authentic View. The Template XML File must be conformant with the schema on which the SPS is based.

Working XML File A Working XML File is an XML file that is assigned to an SPS in StyleVision in order to preview the Authentic View and output of the XML document in StyleVision. Without a Working XML File, the SPS in StyleVision will not have any dynamic XML data to process. If the SPS is based on a schema that has more than one global element, there can be ambiguity about which global element is the document element. Assigning a Working XML File resolves such ambiguity (because a valid XML document will, by definition, have only one document element).

XML document XML document is used in two senses: (i) to refer to a specific XML document; (ii) to refer to any XML data source, including DB sources (from which XML data documents are generated for use with an SPS). Which sense is intended should be clear from the context.

3.4 Setting up StyleVision

Altova StyleVision runs on Windows 2000 and Windows XP. After downloading StyleVision from the Altova website, double-click the executable (`.exe`) file to run the setup program. The setup program will install StyleVision at the desired location. The Altova XSLT Engines (1.0 and 2.0) are built into StyleVision and are used for all internal transformations. You, therefore, do not need to install an XSLT Engine additionally to your StyleVision installation. You will, however, need to have the following components installed:

- Internet Explorer 5.5 or later, for Authentic View and HTML Preview. Internet Explorer 6.0 and later has better XML support and is recommended.

Command line utility: StyleVisionBatch

A command line utility, `StyleVisionBatch.exe`, is included in the installation. This utility can be used to call the file-generation functionality of StyleVision from the command line. StyleVisionBatch is located in the StyleVision application folder.

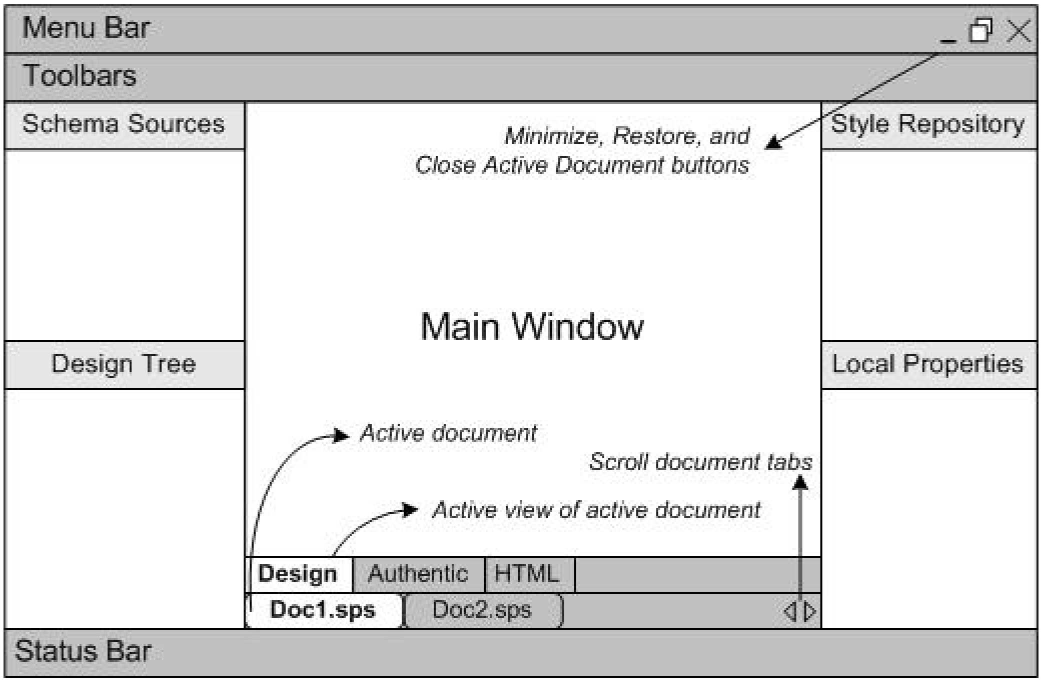
Chapter 4

User Interface

4 User Interface

The StyleVision GUI (*illustration below*) consists of the following parts:

- A **menu bar**. Click on a menu to display the items in that menu. All menus and their items are described in the User Reference section. The menu bar also contains the Minimize, Restore, and Close Active Document buttons.
- A **toolbar area**. The various toolbars and the command shortcuts in each toolbar are described in the User Reference section.
- A tabbed **Main Window**, which displays one or more open SPS documents at a time. In this window, you can edit the design of the SPS, edit the content of Authentic View, and preview the XSLT stylesheets and output.
- The **Design Entry Helpers**—the Schema Sources, Design Tree, Style Repository, Properties, and Styles windows—which can be docked within the application GUI or made to float on the screen.
- A **status bar**, which displays application status information.

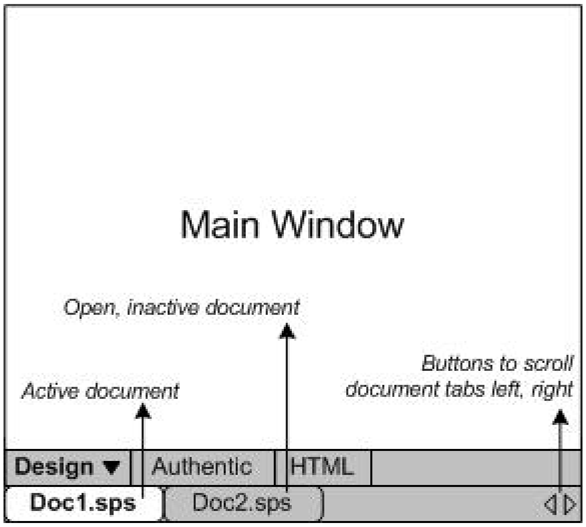


The Main Window and Design Entry Helpers are described in more detail in the sub-sections of this section.

Note: The menu bar and toolbars can be moved by dragging their handles to the required location.

4.1 Main Window

The **Main Window** (*illustration below*) is where the SPS design, Authentic View, XSLT stylesheets, and output previews are displayed.



SPS documents in the Main Window

- Multiple SPS documents can be open in StyleVision, though only one can be active at any time. The names of all open documents are shown in tabs at the bottom of the Main Window, with the tab of the active document being highlighted.
- To make an open document active, click its tab. Alternatively, use the options in the Windows menu.
- If so many documents are open that all document tabs are not visible in the document-tab bar, then click the appropriate scroll button (at the right of the document-tab bar; *see illustration above*) to scroll the tabs into view.
- To close the active document, click the **Close Document** button in the menu bar at the top right of the application window (or select **File | Close**).

Document views

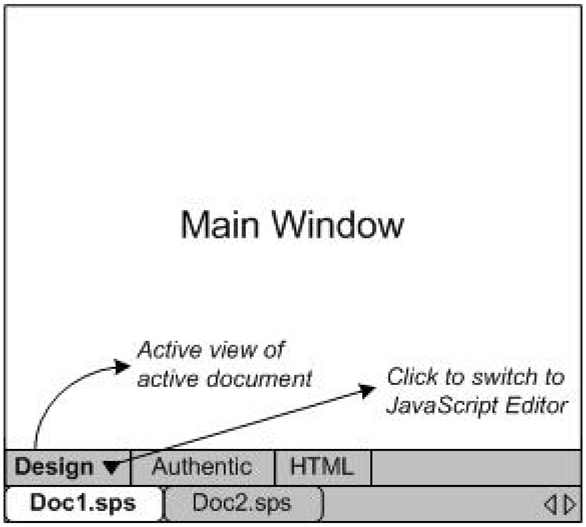
A document is displayed in the following views, one of which can be active at a time:

- Design View, in which you design the SPS and edit JavaScript functions for use in that SPS. The view can be toggled between the design document and the JavaScript Editor by clicking the dropdown menu arrow and selecting Design or JavaScript, as required.
- Authentic View, which enables you to immediately see the Authentic View of an XML document (the Working XML File). The SPS is dynamically applied to the Working XML File, thus enabling you to try out Authentic View.
- Output Views (HTML output). These views are a preview of the actual output format and of the XSLT stylesheet used to generate that output. The view can be toggled between the output preview and the XSLT stylesheet by clicking the dropdown menu arrow and making the appropriate selection.

Each of the views listed above is available as a tab at the bottom of the Main Window in the Views Bar. To select a view, click on its tab. The tab of the selected view is highlighted.

Design View

The **Design View** (*illustration below*) is the view in which the SPS is designed. In Design View, you create the design of the output document by (i) inserting content (using the Entry Helpers, the keyboard, and the various content creation and editing features provided in the menus and toolbars); and (ii) formatting the content using the various formatting features provided in the Entry Helpers and menus. These aspects of the Design View are explained in more detail below.


Main Window
Active view of active document
Click to switch to JavaScript Editor
Design
Authentic
HTML
Doc1.sps
Doc2.sps

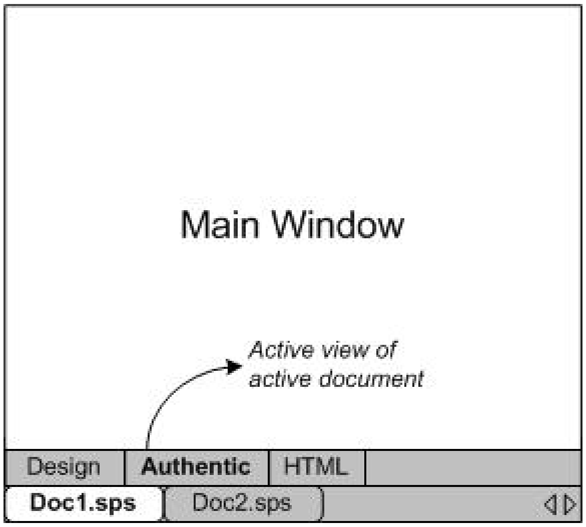
Design View can also be switched to a JavaScript Editor, in which you can create and edit JavaScript functions which then become available in the GUI for use in the SPS. To switch to the JavaScript Editor, click the dropdown button in the Design tab (*see illustration*) and select JavaScript from the dropdown menu. To switch back to Design View, click the dropdown button in the JavaScript tab and select Design from the dropdown menu.

In Design View, the SPS can have several templates: the main template, global templates, and Design Fragments. You can control which of these template types is displayed in Design View by using Template Display Filters, which are available as toolbar icons. These display filters will help you optimize and switch between different displays of your SPS.

Authentic View

In the **Authentic View** tab of the Main Window, you can view and edit the Working XML File in its Authentic View. This view enables you (i) to see how your Authentic XML document will look, and (ii) to test the Authentic View created by the SPS. This is particularly useful if you wish to test the dynamic features of Authentic View. For example, you could test how Authentic View behaves when you:

- Add new elements and attributes
- Add new paragraphs or table rows
- Change values that affect conditional templates



Authentic View and the Working XML File
In order for Authentic View to be displayed, a Working XML File must be assigned to the active SPS document This Working XML File must be valid according to the schema on which the SPS is based.

StyleVision creates a temporary XML file that is based on the Working XML File, and it is this temporary file that is displayed in the Authentic View tab of the Main Window. Modifications that you make in Authentic View will modify the temporary XML file. The Working XML File itself will not be modified till you explicitly save the modifications (with the menu command **File | Save Authentic XML Data**).

If no Working XML File is assigned, you will be prompted to assign a Working XML File when you click the Authentic View tab.

Authentic View limitations
The Authentic View in the Main Window is similar to the full-fledged Authentic View available in XMLSpy and Authentic Desktop except in the following major respects:

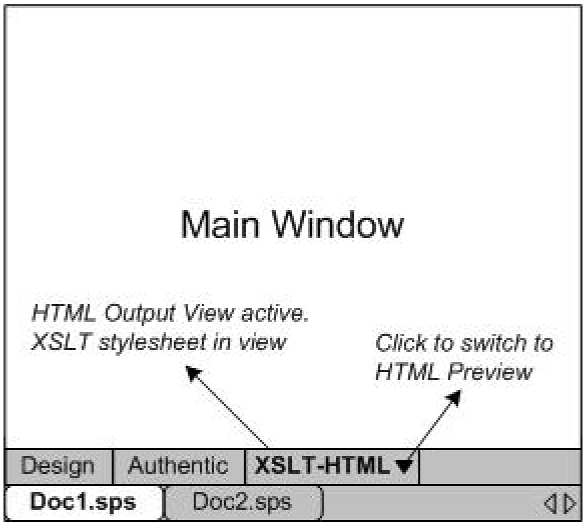
- Authentic View Entry Helpers are not available in the GUI. To insert or append nodes, you must right-click and use the context menus.
- XML tables are not available for insertion.
- Text state icons are not available.

To test these features, you should use the full-fledged Authentic View in XMLSpy or Authentic Desktop. A full description of how to use Authentic View is given in the section Editing in Authentic View. For additional information, please see the Authentic View tutorial in the XMLSpy or Authentic Desktop user manual.

Output Views

The **Output View** tab (*illustration below*) displays: (i) the XSLT-for-HTML stylesheet generated from the SPS design; and (ii) a preview of the HTML output, produced by transforming the Working XML File with the generated XSLT stylesheet.

In the HTML Output View tab, the view can be switched between the XSLT-for-HTML stylesheet and the HTML output preview by clicking the dropdown button in the HTML Output View tab and selecting the XSLT option or the output preview option as required.



XSLT view
The XSLT view displays the XSLT-for-HTML generated from the currently active SPS. The stylesheet is generated afresh each time the XSLT view is selected.

A stylesheet in an Output View tab is displayed with line-numbering and expandable/collapsible elements; click the + and – icons in the left margin to expand/collapse elements. The stylesheet in XSLT view cannot be edited, but can be searched (select **Edit | Find**) and text from it can be copied to the clipboard (with **Edit | Copy**).

Note: The XSLT stylesheets generated from the SPS can be separately generated and saved using the **File | Save Generated Files** command.

HTML preview
HTML preview displays the output produced by transforming the Working XML File with the XSLT-for-HTML. The output is generated afresh each time HTML preview is clicked. Note that it is the saved version of the Working XML File that is transformed—not the temporary version that is edited with Authentic View. This means that any modifications made in Authentic View will be reflected in HTML preview only after these modifications have been saved to the Working XML File (**File | Save Authentic XML Data**).

If no Working XML File is assigned when HTML preview is selected in the HTML View tab, you will be prompted to assign a Working XML File. For DB-based SPSs, there is no need to assign a Working XML File since a temporary non-editable XML file is automatically generated when the DB is loaded and this XML file is used as the Working XML File.

Note: The output files generated from the SPS can be separately generated and saved using the **File | Save Generated Files** command.

4.2 Design Entry Helpers

The **Design Entry Helpers** are GUI components that help you design the SPS and provide information about individual SPS design components. They are enabled when the Design View of a document is active, and are disabled when Authentic View or Output View (HTML View) is active.

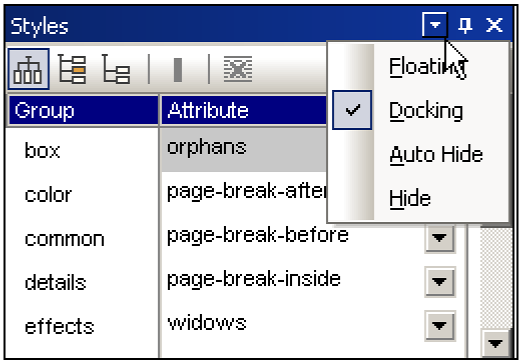
There are five Design Entry Helpers (*listed below*), each of which is described in a sub-section of this section.

- Schema Sources
- Design Tree
- Style Repository
- Styles
- Properties

Docking and floating the Design Entry Helper windows

Design Entry Helper windows can be docked in the StyleVision GUI or can be made to float on your screen. To dock a window, drag the window by its title bar and drop it on any one of the four inner or four outer arrowheads that appear when you start to drag. The inner arrowheads dock the dragged window relative to the window in which the inner arrowheads appear. The four outer arrowheads dock the dragged window at each of the four edges of the interface window. To make a window float, (i) double-click the title bar; or (ii) drag the title bar and drop it anywhere on the screen except on the arrowheads that appear when you start to drag.

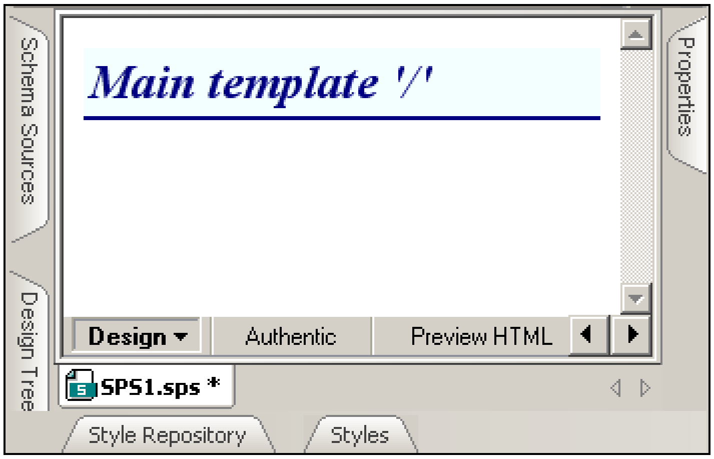
Alternatively, you can also use the following mechanisms. To float a docked window, click the **Menu** button at the top-right of a docked window (*see screenshot below*) and select Floating. This menu can also be accessed by right-clicking the title bar of the docked window.



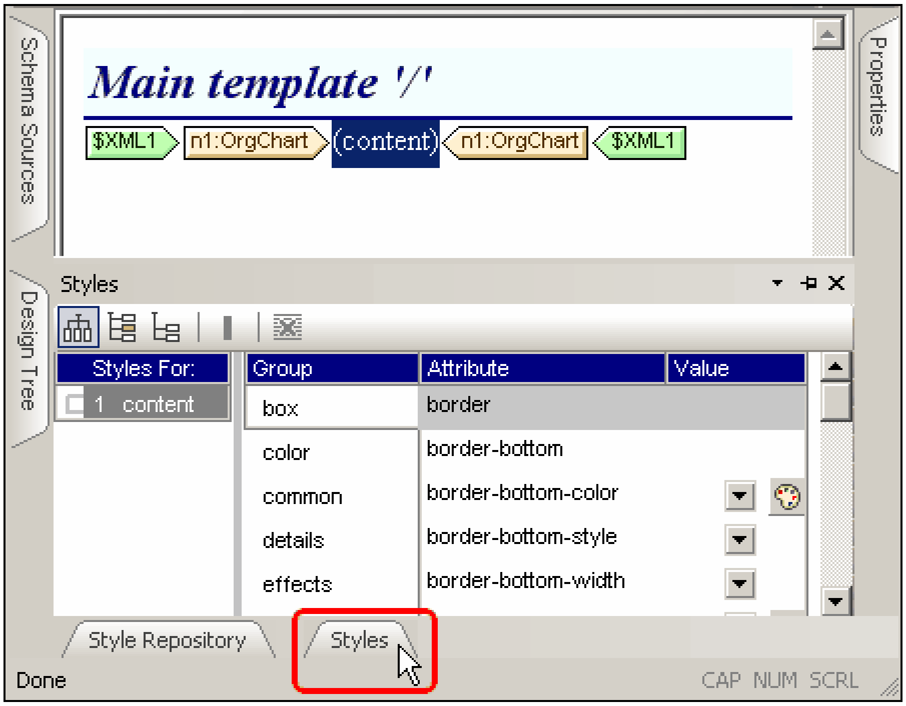
To dock a floating window, right-click the title bar of the floating window and select Docking from the menu that appears; the window will be docked in the position in which it was last docked.

Auto-Hiding Design Entry Helper windows

A docked window can be auto-hidden. When a Design Entry Helper window is auto-hidden, it is minimized to a tab at the edge of the GUI. In the screenshot below, all five Design Entry Helpers have been auto-hidden: two at the left edge of the GUI, two at the bottom edge, and one at the right edge.



Placing the cursor over the tab causes that window to roll out into the GUI and over the Main Window. In the screenshot below, placing the cursor over the Styles tab causes the Styles Design entry helper to roll out into the Main Window.



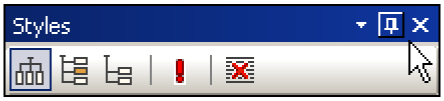
Styles For:
1 content

Group	Attribute	Value
box	border	
color	border-bottom	
common	border-bottom-color	
details	border-bottom-style	
effects	border-bottom-width	

Moving the cursor out of the rolled-out window and from over its tab causes the window to roll back into the tab at the edge of the GUI.

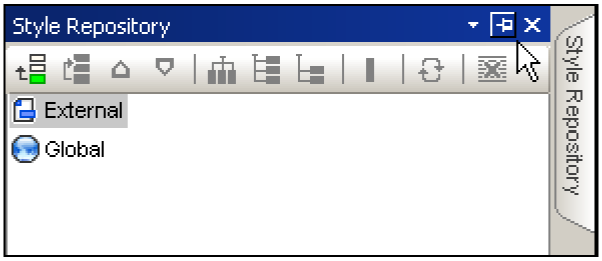
The Auto-Hide feature is useful if you wish to move seldom-used entry helpers out of the GUI while at the same time allowing you easy access to them should you need them. This enables you to create more screen space for the Main Window while still allowing easy access to Design Entry Helper windows.

To auto-hide a window, in a docked window, click the Auto Hide button (the drawing pin icon) at the top right of the window (*screenshot below*). Alternatively, in the Menu, select Auto Hide; (to display the Menu, right-click the title bar of the window or click the Menu button in the title bar of the docked window).



The window will be auto-hidden.

To switch the Auto-Hide feature for a particular window off, place the cursor over the tab so that the window rolls out, and then click the Auto Hide button (*screenshot below*). Alternatively, in the Menu, deselect Auto Hide; (to display the Menu, right-click the title bar of the window or click the Menu button in the title bar of the window).



Note: When the Auto-Hide feature of a Design Entry Helper window is off, the drawing pin icon of that window points downwards; when the feature is on, the drawing pin icon points left.

Hiding (closing) Design Entry Helper windows

When a Design Entry Helper window is hidden it is no longer visible in the GUI, in either its maximized form (docked or floating) or in its minimized form (as a tab at an edge of the GUI, which is done using the Auto-Hide feature).

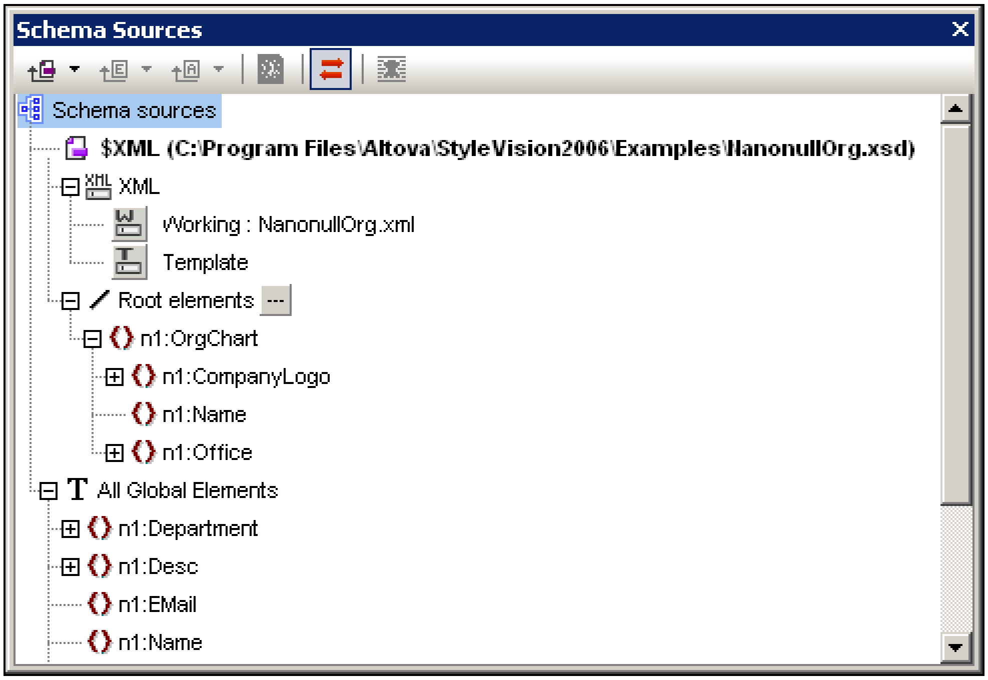
To hide a window, click the Close button at the top right of a docked or floating window (*screenshot below*). Alternatively, in the Menu, select Hide; (to display the Menu, right-click the title bar of the window or click the Menu button in the title bar of the window).

To make a hidden (or closed) window visible again, select the name of the Design Entry Helper in the **View** menu. The Design Entry Helper window is made visible in the position at which it was (docked or floating) when it was was hidden.

Schema Sources

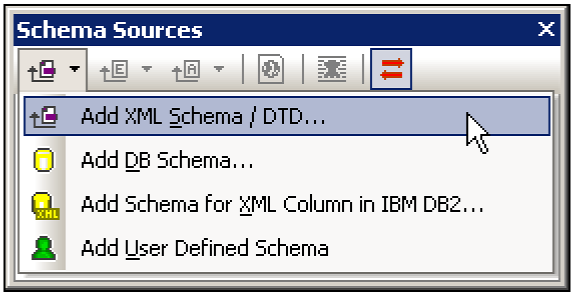
The **Schema Sources** window (*screenshot below*) enables you to do the following:

- Specify the schema or DB on which the SPS will be based.
- Specify the Working XML File and Template XML File for the schema source.
- Select multiple root elements (document elements) for a schema.
- Drag nodes from a schema tree and drop them into the design. These nodes represent the XML content that is to be included in the output.
- Lists all global elements in the schema source. A global element can be created as a global template.



Adding a schema

A schema is added by clicking the dropdown arrow of the **Add Schema** icon in the toolbar. This pops up a menu (*screenshot below*) that enables you to add: (i) an XML Schema or DTD; (ii) a schema generated by StyleVision from a DB; or (iii) a user-defined schema.



The Working XML File and Template XML File

When a schema is added, it is listed under the Schema Sources item. Each schema has two important sub-items (within the XML item), in order:

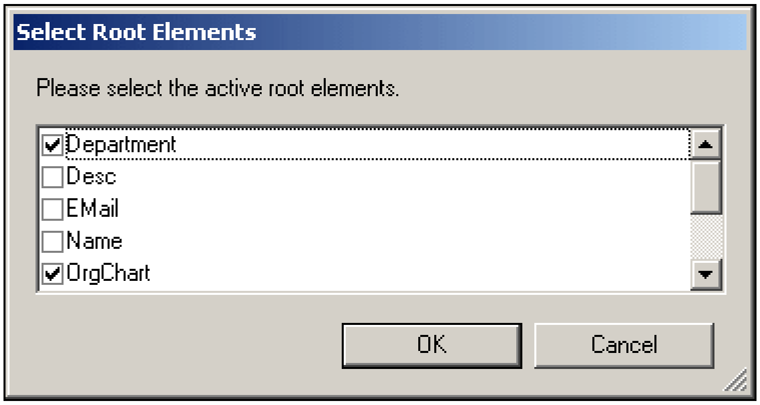
- The Working XML File.
- The Template XML File.

To select or change the Working XML File or Template XML File, click the W or T button, respectively, and browse for the required file.

Root elements

For each schema, there is a Root elements (aka document elements) list. This list consists of all the root elements you select for the schema (see below for how to do this). Each root element can be expanded to show its content model tree. It is from the nodes in these root element trees that the content of the main template is created. Note that the entry point of the main template is the document node of the main schema, which you can select or change at any time (see below for how to do this).

To select the root elements for a schema, do the following: Click the Select ... button at the right of the Root elements item. This pops up the Select Root Elements dialog (*screenshot below*), in which you can select which of the global elements in the schema is/are to be the root elements. See SPS Structure | Schema Sources for an explanation of the possibilities offered by a selection of multiple root elements.



Select Root Elements

Please select the active root elements.

☑ Department
☐ Desc
☐ EMail
☐ Name
☑ OrgChart

Additionally, all the global elements in the schema are listed under the All Global Elements item. For each global element, a global template can be created.

Toolbar and schema tree icons

The following toolbar icons are shortcuts for common Schema Sources entry helper commands.

Adds a schema from a file or a DB, or adds a user-defined schema. The option is selected from a dropdown list.

In a user-defined schema, adds a child element to the document element or appends a sibling element to the selected element. Alternative levels are available by clicking the dropdown arrow of the icon.

In a user-defined schema, adds an attribute to the document element or appends an attribute at the same level to the selected node. Alternative levels are available by clicking the dropdown arrow of the icon.

Make/Remove Global Template, enabled when a global element is selected.

Synchronize tree toggle. When toggled on (icon has border), selecting a node in the tree selects (i) the corresponding node in the design, and (ii) the corresponding node in the Design Tree if the Synchronize Tree icon in the Design Tree window is toggled on. Whe toggled off, the corresponding node in the design is not selected. Switch the toggle off if dragging a node from the tree and dropping it to the desired location in the design proves difficult.

Remove the selected item.

Browse for the Working XML File for that schema.

Browse for the Template XML File for that schema.

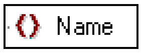
Context menu

The context menu contains the following commands:

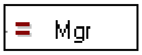
- Add schema
- Set as main schema
- Add element, add attribute, convert to element/attruíbute
- Make/Remove Global Template
- Expand and collapse commands. These enable the tree to be expanded and collapsed. Options include the entire Schema Sources tree, up to the selected level, and to expand from or collapse to the selected item.

Symbols used in schema trees

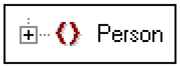
Given below is a list of the symbols in schema trees.



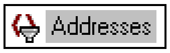
Element.



Attribute.



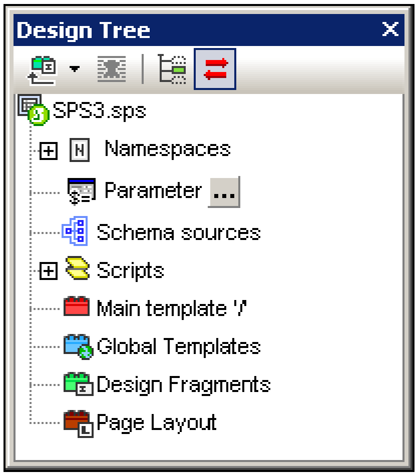
Element with child elements. Double-clicking the element or the +/- symbol to its left causes the element to expand/collapse.



DB Filter applied. Applies only to top-level data table elements in the schema tree.

Design Tree

The **Design Tree** window (*screenshot below*) provides an overview of the SPS design.

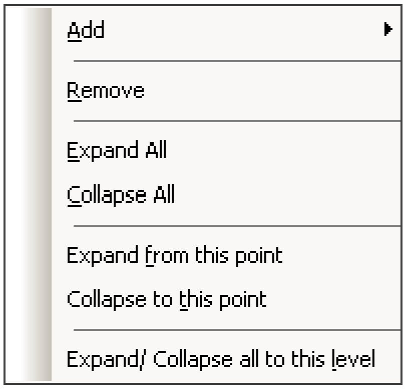


At the root of the Design Tree is the location of the SPS file. The next level is organized into the following categories:

- Namespaces, which displays all the namespaces declared in the SPS.
- Parameters, which enables the Edit Parameters dialog to be displayed. In the Edit Parameters dialog, you can declare and edit user-defined parameters for the SPS and their default values.
- Schema sources, which lists the schema on which the SPS is based, and its location.
- Scripts, which shows all the JavaScript functions that have been defined for the SPS using the JavaScript Editor of StyleVision.
- Main Template, which displays a detailed structure of the main template.
- Global Templates, which gives a detailed graph of all the global templates in the SPS, including the main template.
- Design Fragments, which shows all the Design Fragments in the design, and their structures.
- Layout, which enables you to customize HTML layout items.

Modifying the Design Tree display

The display of the Design Tree can be modified via the context menu (*screenshot below*), which pops up on right-clicking an item in the Design Tree.

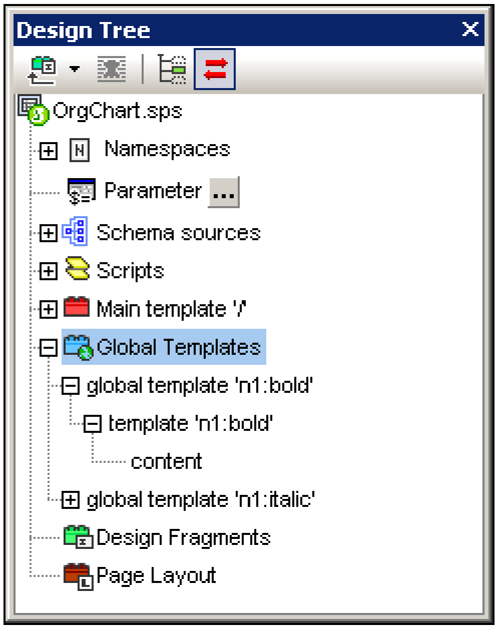


A description of these commands is given in the following table.

Add	Enables page layout items to be added to the Layout category.
Remove (Item)	Applies to certain categories such as Global Templates. Removes the selected item from the Design Tree.
Expand All	Expands all expandable items in all categories of the Design Tree.
Collapse All	Collapses the entire Design Tree to the top-level item, which is the location of the SPS file.
Expand from This Point	Expands all expandable items in the selected item.
Collapse to This Point	Collapses all items within the selected item, up to the selected item.
Expand/Collapse All to This Level	Expands or collapses all categories to the level of the selected item.

Global Templates

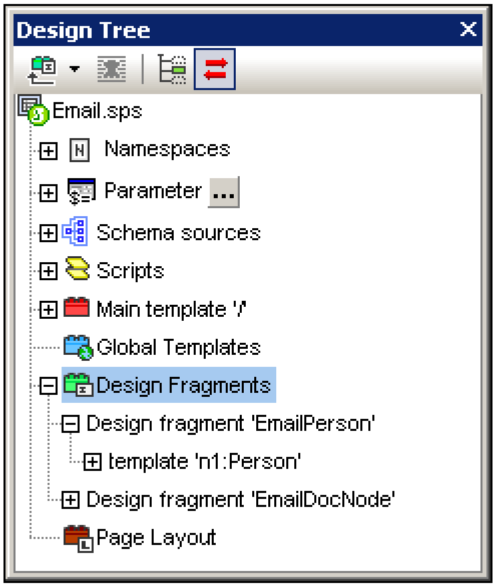
The Global Templates item lists all the global templates in the SPS. These templates are displayed as trees with expandable/collapsible nodes (*see screenshot below*).



Any component in any of the template trees can be removed by selecting it and clicking the Remove button in the toolbar or the Remove command in the context menu. The component is removed from the design and the tree.

Design Fragments

The Design Fragments item lists all the Design Fragments in the SPS. These templates are displayed as trees with expandable/collapsible nodes (*see screenshot below*).



Any component in any of the template trees can be removed by selecting it and clicking the Remove button in the toolbar or the Remove command in the context menu. The component is removed from the design and the tree.

Layout

The Layout item enables you to set the title of the HTML page and hyperlink properties. These settings are made in the Properties entry helper when Layout is selected in the Design Tree entry helper.

Toolbar icons

The following toolbar icons are shortcuts for common Schema Sources entry helper commands.

Adds a Design Fragment or layout item to the design. Clicking the left-hand part of the icon adds a Design Fragment. Clicking the dropdown arrow drops down a list with commands to add a Design Fragment or any of various layout items.

Remove the selected item; icon is active when item in the Global Templates or Layout sub-trees is selected.

Autocollapses items in the design tree.

Synchronize tree toggle. When toggled on (icon has border), selecting a node in the tree selects (i) the corresponding node in the design, and (ii) the corresponding node in the Schema Sources tree if the Synchronize Tree icon in the Schema Sources tree is toggled on. When toggled off, the corresponding nodes in the design and Schema Sources tree are not selected.

Style Repository

In the **Style Repository** window (*screenshot below*), you can assign external CSS stylesheets and define global CSS styles for the SPS. Style rules in external CSS stylesheets and globally defined CSS styles are applied to Authentic View and the HTML output document.

The Style Repository window contains two listings, **External** and **Global**, each in the form of a tree. The External listing contains a list of external CSS stylesheets associated with the SPS. The Global listing contains a list of all the global styles associated with the SPS.

The structure of the listings in the Style Repository is as follows:

```
External
- CSS-1.css
  - Location of file (editable in Style Repository window)
  - Media (can be defined in Style Repository window)
  - Rules (non-editable; must be edited in CSS file)
    - Selector-1
      - Property-1
      - ...
      - Property-N
    - ...
    - Selector-N
+ ...
+ CSS-N.css
Global
- Selector-1
  + Selector-1 Properties
- ...
+ Selector-N
```

Precedence of style rules

If a global style rule and a style rule in an external CSS stylesheet have selectors that identify the same document component, then the global style rule has precedence over that in the external stylesheet, and will be applied. If two or more global style rules select the same document component, then the rule that is listed last from among these rules will be applied. Likewise, if two or more style rules in the external stylesheets select the same document component, then the last of these rules in the last of the containing stylesheets will be applied

Managing styles in the Style Repository

In the Style Repository window you can do the following, using either the icons in the toolbar and/or items in the context menu:

Add: The **Add** icon adds a new external stylesheet entry to the External tree or a new global style entry to the Global tree, respectively, according to whether the External or Global tree was selected. The new entry is appended to the list of already existing entries in the tree. The **Add** command is also available in the context menu. For more details about using external stylesheets and global styles, see Working with CSS Styles.

Insert: The **Insert** icon inserts a new external stylesheet entry above the selected external stylesheet (in the External tree) or a new global style entry above the selected global style (in the Global tree). The **Insert** command is also available in the context menu. For more details about using external stylesheets and global styles, see Working with CSS Styles.

Move Up/Down: The **Move Up** icon and **Move Down** icon move the selected external stylesheet or global style respectively up and down relative to the other entries in its tree. These commands are useful for changing the priority of external stylesheets relative to each other and of global style rules relative to each other. The **Move Up** and **Move Down** commands are also available in the context menu. For more details about how to change the precedence of styles, see Working with CSS Styles.

Views of global style properties: The properties of a global style can be displayed in one of three views: (i) by property group; (ii) all properties sorted alphabetically; (iii) properties with values defined, sorted alphabetically. The view can be changed for each style individually. To change the properties view of a global style, select that style and click one of the View icons in the Style Repository toolbar: **Grouped** ; **List All** ; and **List Non-Empty** . These commands are also available in the context menu under the **View Mode** item.

Toggle Important: Clicking the Toggle Important icon sets the CSS value `!important` on or off for the selected CSS rule.

Reload All: The **Reload All** icon reloads all the external CSS stylesheets.

Reset: The **Reset** icon deletes the selected external stylesheet or global style.

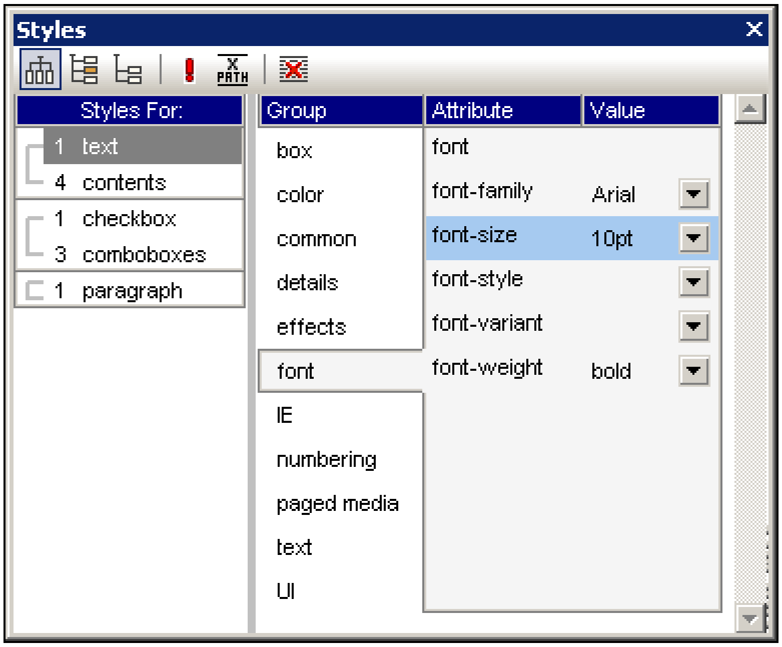
Expand/Collapse All: All expandable items in both the External and Global trees can be expanded and collapsed with one click using the **Expand All** and **Collapse All** commands in the context menu, respectively.

Editing CSS styles in the Style Repository
The following editing mechanisms are provided in the Style Repository:

- You can replace an assigned CSS Stylesheet with another CSS stylesheet, and you can specify the media to which each external CSS stylesheet applies. How to do this is explained in the section External CSS Stylesheets.
- Global styles can have their selectors and properties directly edited in the Style Repository window. How this is done is described in the section Defining CSS Styles Globally.

Styles

The **Styles** window (*screenshot below*) enables CSS styles to be defined locally for SPS components selected in the Design View.



Attribute	Value
font	
font-family	Arial
font-size	10pt
font-style	
font-variant	
font-weight	bold

The Styles window is divided into two broad parts:

- The **Styles For column**, in which the selected component types are listed. One of these component types may be selected at a time for styling. (In the screenshot above, the *1 text* component is selected.) For detailed information about the selection of component types, see Selecting SPS Components to Style.
- The **Property Definitions column**, in which CSS properties are defined for the component type/s selected in the Styles For column. The Property Definitions column can be displayed in three views (*see below*). For the details of how to set local property definitions, see Setting CSS Property Values. The XPath icon toggles on and off the application of XPath expressions as the source of property values. With a property selected, if the XPath icon is toggled on, then an XPath expression can be entered for this property and the return value of the XPath expression is used as the value of that property. In this way, the value of a node in an XML document can be returned, at runtime, as the value of a property. When the XPath icon is toggled off, a static value can be entered as the value of the property.

Three views of Property Definitions

The Property Definitions column shows the properties of the component selected in Design View. The display is available in three views (*listed below*) and can be switched between each other by clicking the respective buttons in the toolbar of the Entry Helper:

- **Grouped** : The properties are organized into groups. In this view, the Property Definitions column is divided into three columns: Group, Attribute, and Value. All the available property groups are displayed in the Group column. When a group is selected, the properties of that group are displayed in the Attribute column. If a value for a property is defined, the value appears in the Value column.

- **List All** : All properties of all groups are listed in a single alphabetically ordered list. The Attribute column is listed first, followed by the Group column and then the Value column.
- **List Non-Empty** : Only properties that have values defined are listed. The columns are ordered, from left to right, as follows: Attribute, Group, and Value. In this view, it will not be possible to define a value for a new property—because no undefined property is listed. However, this is a quick way to see all the defined properties for the selected component type, and the displayed properties can be edited.

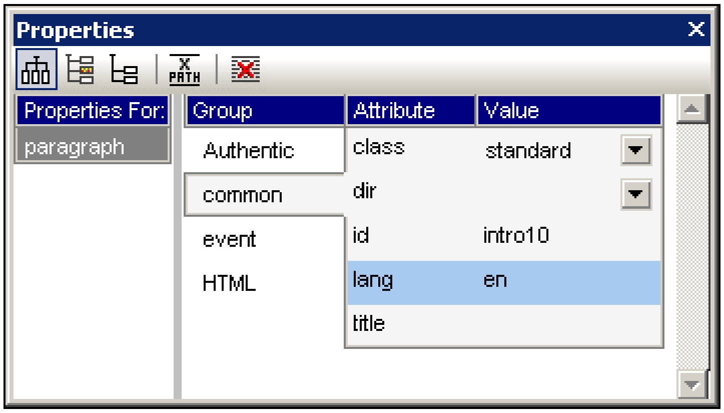
Views can also be changed by right-clicking any item in the Property Definitions column, selecting **View Mode**, and then the required view.

Toggle Important and Reset toolbar icons

Clicking the Toggle Important icon sets the CSS value `!important` on or off for the selected CSS rule. Clicking the Reset icon resets the value of the selected property.

Properties

The **Properties** window (*screenshot below*) enables properties to be defined for SPS components selected in the Design View.



The Properties window is divided into two broad parts:

- The **Properties For column**, in which the selected component types are listed. One of these component types may be selected at a time and properties assigned for it. (In the screenshot above, the *paragraph* component is selected.) For detailed information about how components with properties are grouped, see the section Components and their Property Groups below.
- The **Property Definitions column**, in which component properties are defined for the component type selected in the Properties For column. The Property Definitions column can be displayed in three views (*see below*). For the details of what properties are in each property group, see the section Property Groups below.

Three views of Property Definitions

The Property Definitions column shows the properties of the component selected in Design View. The display is available in three views (*listed below*) and can be switched between each other by clicking the respective buttons in the toolbar of the Entry Helper:

- **Grouped** : The properties are organized into groups. In this view, the Property Definitions column is divided into three columns: Group, Attribute, and Value. All the available property groups are displayed in the Group column. When a group is selected, the properties of that group are displayed in the Attribute column. If a value for a property is defined, the value appears in the Value column.
- **List All** : All properties of all groups are listed in a single alphabetically ordered list. The Attribute column is listed first, followed by the Group column and then the Value column.
- **List Non-Empty** : Only properties that have values defined are listed. The columns are ordered, from left to right, as follows: Attribute, Group, and Value. In this view, it will not be possible to define a value for a new property—because no undefined property is listed. However, this is a quick way to see all the defined properties for the selected component type, and the displayed properties can be edited.

Views can also be changed by right-clicking any item in the Property Definitions column, selecting **View Mode**, and then the required view.

Reset toolbar icon

Clicking the Reset icon ☒ resets the value of the selected property to its default.

Components and their property groups

The availability of property groups is context-sensitive. What property groups are available depends on what design component is selected. The table below lists SPS components and the property groups they have.

Component	Property Group
Template	Authentic
Content	Authentic; Common; Event
Text	Common; Event
Auto-Calculation	AutoCalc; Authentic; Common; Event
Condition Branch	When
Data-Entry Device	Authentic; Common; [Data-Entry Device]; Event; HTML
Image	Image; Authentic; Common; Event; HTML
Link	Link; Authentic; Common; Event; HTML
Table	Authentic; Common; Event; HTML
Paragraph	Paragraph; Authentic; Common; Event; HTML

The following points about component types should be noted:

- Template components are the main template, global templates, and all schema nodes in the design.
- Content components are the `content` and `rest-of-contents` placeholders. These represent the text content of a node or nodes from the XML document.
- A text component is a single string of static text. A single string extends between any two components other than text components, and includes whitespace, if any is present.
- Data-entry devices are input field, multiline input fields, combo boxes, check boxes, radio buttons and buttons; their properties cover the data-entry device as well as the contents of the data-entry device, if any.
- A table component refers to the table structure in the design. Note that it contains sub-components, which are considered components in their own right. The sub-components are: row, column, cell, header, and footer.
- A paragraph component is any predefined format.

The table below contains descriptions of each property group.

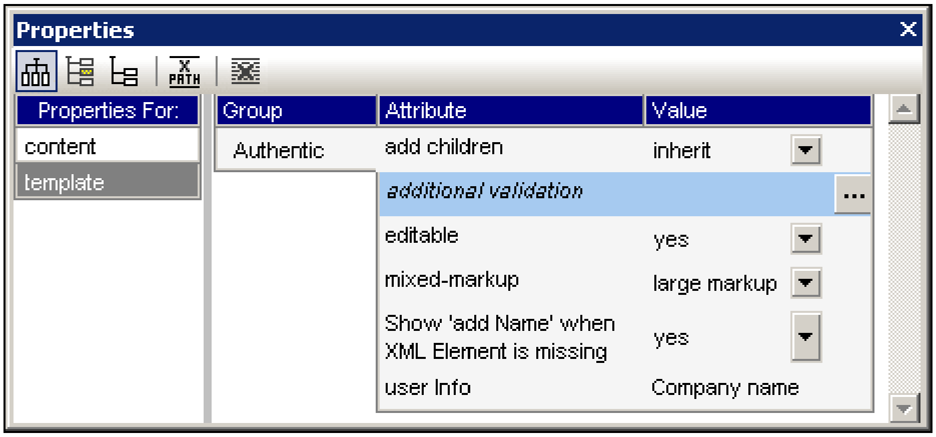
Property Group	Description
AutoCalc	These properties are enabled when an Auto-Calculation is selected. The `Input Formatting` property specifies the formatting of an Auto-Calculation that is a numeric or date datatype. The `XPath` property specifies the XPath expression that is used for the Auto-Calculation.
Authentic	These are SPS-specific properties that are available for templates, contents, AutoCalculations, data-entry devices, images, links, tables, and paragraphs. What properties within the group are available are component-specific. For more details, see Authentic Node Properties.
Common	The *Common* property group is available for all component types except the Template and AutoCalc component types. It contains the following properties that can be defined for the component: `class` (a class name), `dir` (the writing direction), `id` (a unique ID), `lang` (the language), and `title` (a name).
Data-Entry Device	Specifies the value range of combo boxes, check boxes, and radio buttons. Note that this property group does not apply to input fields and buttons.
Event	Contains properties that enable JavaScript functions to be defined for the following client-side HTML events: `onclick, ondblclick, onkeydown, onkeypressed, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup.`
HTML	Available for the following component types: data-entry devices; image; link; table; paragraphs. Note that there are different types of data-entry devices and paragraphs, and that tables have sub-components. These properties are HTML properties that can be set on the corresponding HTML elements (`img`, `table`, `p`, `div`, etc). The available properties therefore vary according to the component selected. Values for these properties can be selected using XPath expressions.

In addition, there are component-specific properties for images, links, paragraphs and other predefined formats, and condition branches. These properties are described in the respective sections.

Setting property values

Property values can be entered in one, two, or three ways, depending on the property (*see screenshot below*):

- Entered directly in the Value column. To do this, select a property, double-click in its Value column, enter the value using the keyboard, and press **Enter** or click anywhere in the GUI.
- By selecting a value from the dropdown list of the combo box for that property. Click the down arrow of the combo box to drop down the list of property-value options.
- By using the Edit button … at the right-hand side of the Value column for that property. Clicking the Edit button pops up a dialog relevant to that property. For example, the entry helper for the `Format` property in the screenshot below pops up the Input Formatting dialog, while that for the `XPath` property pops up the Edit XPath Expression dialog.



Properties

Properties For:
content
template

Group	Attribute	Value
Authentic	add children	inherit
	additional validation	...
	editable	yes
	mixed-markup	large markup
	Show 'add Name' when XML Element is missing	yes
	user Info	Company name

For some properties, in the Common and HTML groups of properties, XPath expressions can be used to provide the values of the property. The XPath icon toggles on and off the application of XPath expressions as the source of property values. With a property selected, if the XPath icon is toggled on, then an XPath expression can be entered for this property and the return value of the XPath expression is used as the value of that property. In this way, the value of a node in an XML document can be returned, at runtime, as the value of a property. When the XPath icon is toggled off, a static value can be entered as the value of the property.

Modifying or deleting a property value
To modify a property value, use any of the applicable methods described in the previous paragraph, Setting Property Values. To delete a property value, select the property and click the Reset icon in the toolbar of the Properties entry helper.

Chapter 5

Quick Start Tutorial

5 Quick Start Tutorial

The objective of this tutorial is to take you quickly through the the key steps in creating an effective SPS. It starts with a section on creating and setting up the SPS, shows you how to insert content in the SPS, how to format the components of the SPS, and how to use two powerful SPS features: Auto-Calculations and conditions. Along the way you will get to know how to structure your output efficiently and how to use a variety of structural and presentation features.

Files required
Files related to this Quick Start tutorial are in the application folder `StyleVision2007/Examples/Tutorials/QuickStart`:

- `QuickStart.xsd`, the XML Schema file on which the SPS is based.
- `QuickStart.xml`, the Working XML File, which is the source of the data displayed in the output previews.
- `QuickStart.sps`, which is the finished SPS file; you can compare the SPS file you create with this file.
- `QuickStart.css`, which is the external CSS stylesheet used in the tutorial.
- `NewsItems.BMP`, an image file that is used in the SPS.

Doing the tutorial
It is best to start at the beginning of the tutorial and work your way through the sections. Also, you should open the XSD and XML files before starting the tutorial and take a look at their structure and contents. Keep the XSD and XML files open while doing the tutorial, so that you can refer to them. Save your SPS document with a name other than `QuickStart.sps` (say `MyQuickStart.sps`) so that you do not overwrite the supplied SPS file. And, of course, remember to save after successfully completing every part.

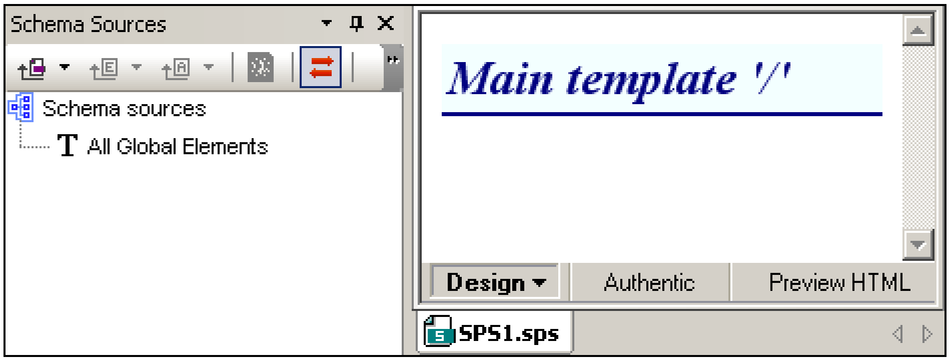
5.1 Creating and Setting Up a New SPS

In this section, you will learn:

- How to create a new SPS document.
- How to add a schema source for the SPS.
- How to select the XSLT version of the SPS.
- How to assign the Working XML File.
- How to specify the output encoding.
- How to save the SPS document.

Creating a new SPS document

Create a new SPS document by clicking **File | New | New (Empty)** or select **New (Empty)** in the dropdown list of the New icon in the application toolbar. A new document titled `SPS1.sps` is created and displayed in Design View (*screenshot below*).

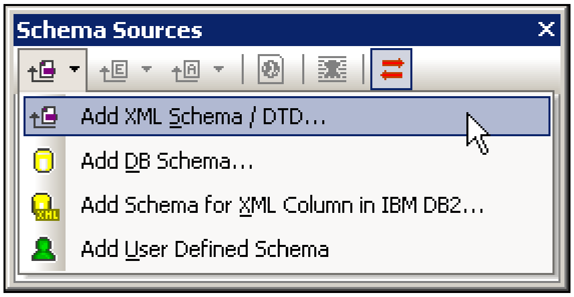


In Design View, an empty main template is displayed. In the Schema Sources entry helper, there are no schema entries.

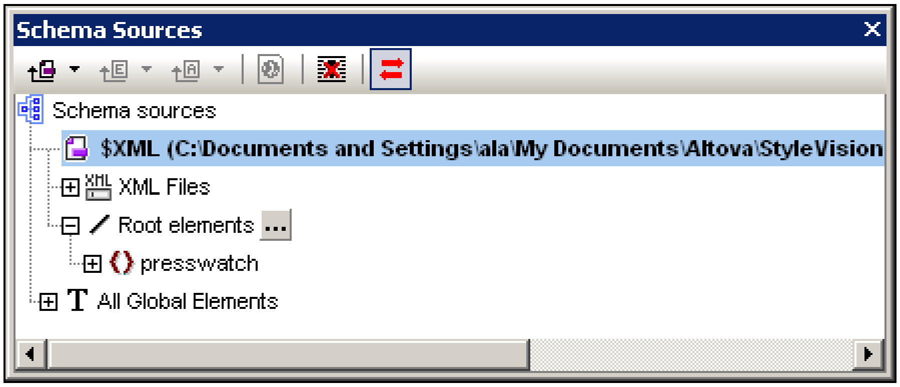
Adding a schema source

For this SPS, you will use the schema, `QuickStart.xsd`. To add this schema as the schema source, do the following:

1. In the Schema Sources entry helper, click the dropdown button of the **Add Schema** icon (*screenshot below*), and select **Add XML Schema / DTD**.



2. In the Open dialog that pops up browse for the file `StyleVision2007/Examples/Tutorials/QuickStart.xsd`, and click **Open**.
3. You will be prompted to select a Working XML File. Select the file `StyleVision2007/Examples/Tutorials/QuickStart.xml`, and click **Open**. The schema will be added to the Schema Sources tree (*screenshot below*). Also, the Working XML File you chose will be assigned to the schema.



Schema Sources

Schema sources
$XML (C:\Documents and Settings\ala\My Documents\Altova\StyleVision
XML Files
Root elements ...
presswatch
All Global Elements

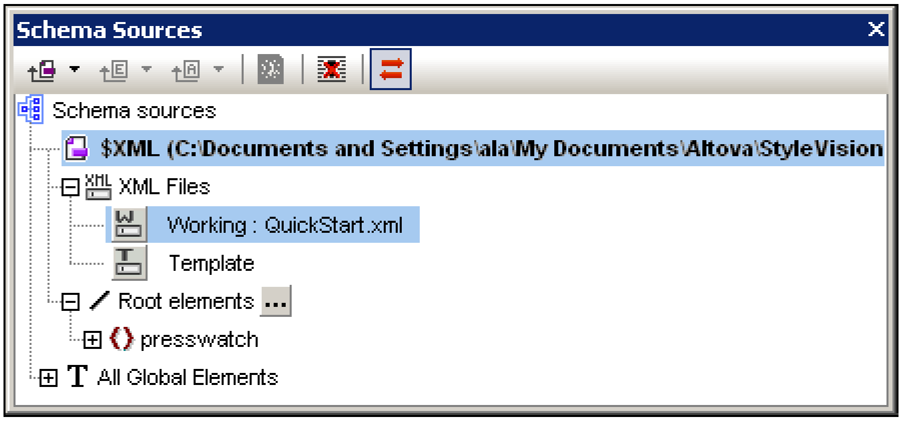
You should note the following points about the Schema Sources tree: (i) The schema is identified by the parameter `$XML1`, and the parameter's value is the URI of the document node of the Working XML File for this schema source. (ii) The XML entry XML directly under the name of the schema, when expanded, contains two entries: the first, W, to select the Working XML File, the second, T, to select the Template XML File. (iii) The Root Elements tree lists the root elements (document elements) you select from among the global elements defined in the schema. The element `presswatch` is selected by default because it is the one global element in the schema that lies clearly at the top of the hierarchy defined in the schema. (iv) All global elements in the schema are listed in the All Global Elements tree.

Selecting the XSLT version

For this SPS you will use XSLT 2.0. To specify the XSLT version, in the toolbar, click the XSLT 20 icon.

Assigning or changing the Working XML File

While adding the XML Schema to the SPS in the previous step, you also assigned a Working XML File to the schema. A Working XML File provides the SPS with a source of XML data to process. To assign or change a Working XML File for a given schema, click the **Add Working XML** button W directly under the name of the schema. In the Open dialog that pops up, browse for the required XML file, and click **Open**. The Working XML File is now assigned, and the filename is entered in the line for the Working XML File entry (*see screenshot below*). Before proceeding, ensure that you have correctly assigned the file `StyleVision2007/Examples/Tutorials/QuickStart.xml` as the Working XML File.



Schema Sources

Schema sources
$XML (C:\Documents and Settings\ala\My Documents\Altova\StyleVision
XML Files
Working : QuickStart.xml
Template
Root elements ...
presswatch
All Global Elements

Specifying the encoding of output
In the Encoding tab of the Options dialog (**Tools | Options**), set the HTML encoding to Unicode UTF-8.

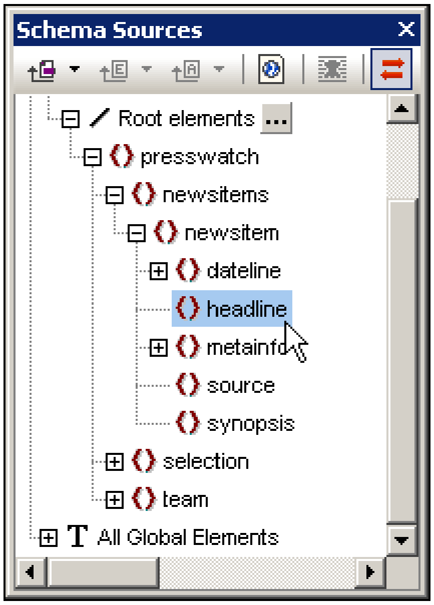
Saving the SPS document
After you have set up the SPS as described above, save it as `MyQuickStart. sps` in the `StyleVision2007/Examples/Tutorials/QuickStart` folder. Do this by clicking the menu command **File | Save Design** or **Ctrl+S**, and then entering the file name in the Save Design dialog that pops up.

5.2 Inserting Dynamic Content (from XML Source)

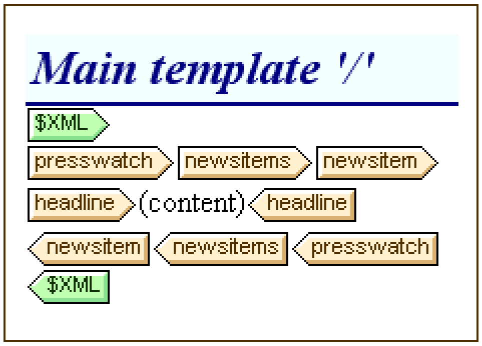
This section introduces mechanisms to insert data from nodes in the XML document. In it you will learn how to drag element and attribute nodes from the schema tree into the design and create these nodes as contents. When a node is created as contents, the data in it is output as a string which is the concatenation of the content of that element's child text nodes and the text nodes of all descendant elements.

Inserting element contents
In your SPS, do the following:

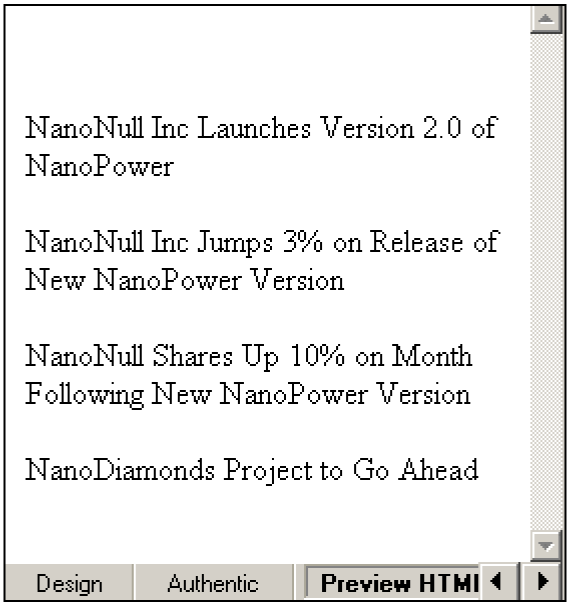
1. In the Schema Sources entry helper, expand the schema tree up to the children of the `newsitem` element (*screenshot below*).



2. Select the `headline` element, drag it into Design View, and, when the arrow cursor turns to an insertion point, drop it into the main template.
3. In the context menu that pops up, select **Create Contents**. The start and end tags of the `headline` element are inserted at the point where you dropped the `headline` element, and they contain `the content` placeholder. The `headline` tags are surrounded by the start and end tags of the ancestor elements of `headline` (*screenshot below*).
4. In the design put elements on different lines (by pressing **Enter**) as shown in the screenshot below.



Click the HTML tab to see a preview of the HTML output (*screenshot below*). The HTML preview shows the contents of the `headline` child elements of `newsitem`, each as a text string.



NanoNull Inc Launches Version 2.0 of NanoPower

NanoNull Inc Jumps 3% on Release of New NanoPower Version

NanoNull Shares Up 10% on Month Following New NanoPower Version

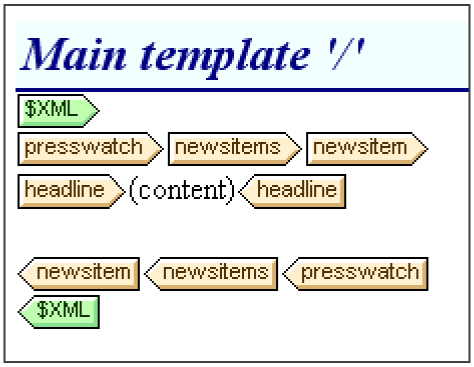
NanoDiamonds Project to Go Ahead

You should also check the preview of Authentic View.

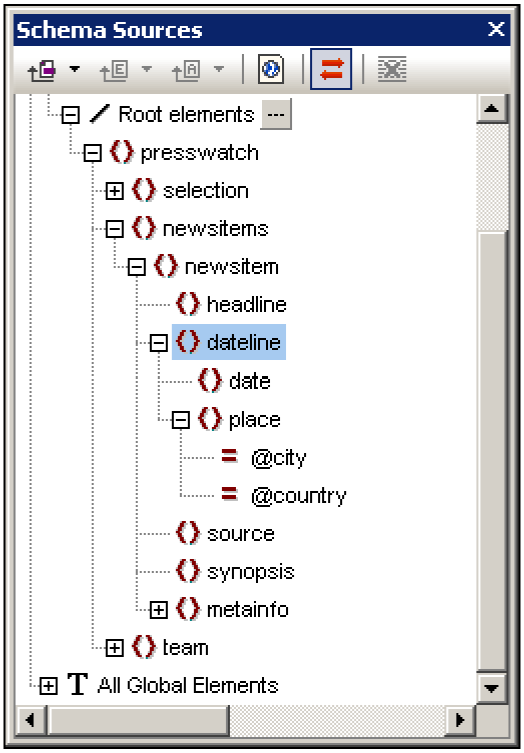
Inserting attribute contents

When an element is inserted into the design as contents, the contents of its attributes are not automatically inserted. You must explicitly drag the attribute node into the design for the attribute's value to be output. In your SPS, now do the following:

1. Place the cursor after the end tag of the `headline` element and press **Enter**. This produces an empty line (*screenshot below*).

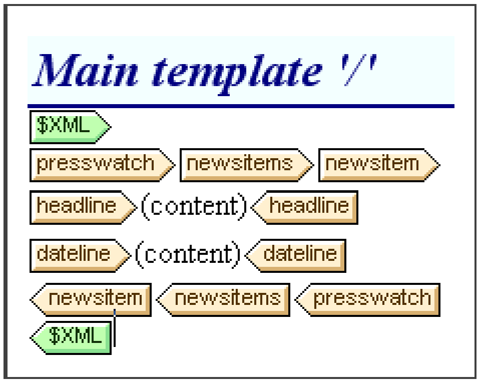


2. In the Schema Sources entry helper, expand the `dateline` element (*screenshot below*).

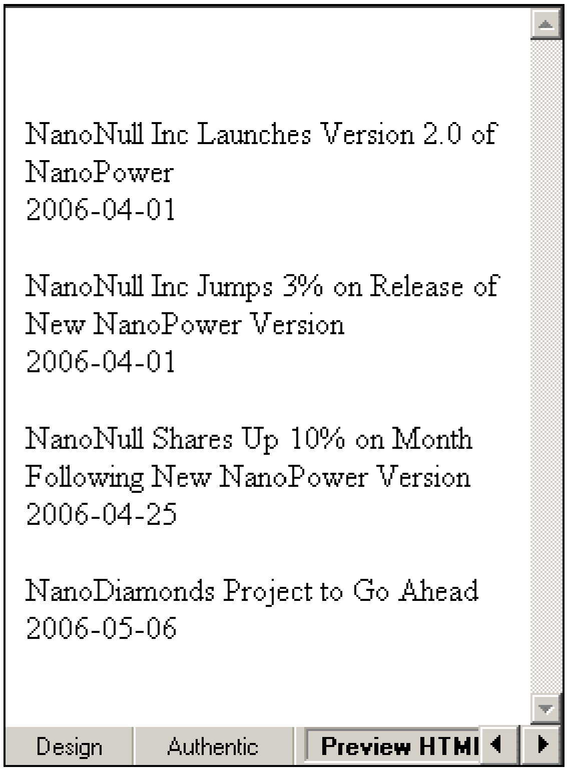


Notice that the `dateline` element has two child elements, `date` and `place`, and that the `place` element has two attributes, `city` and `country`.

3. Drag the `dateline` element into the design and drop it at the beginning of the newly created empty line (*screenshot below*).



4. Switch to HTML Preview and look carefully at the output of `dateline` (*screenshot below*).



NanoNull Inc Launches Version 2.0 of NanoPower
2006-04-01

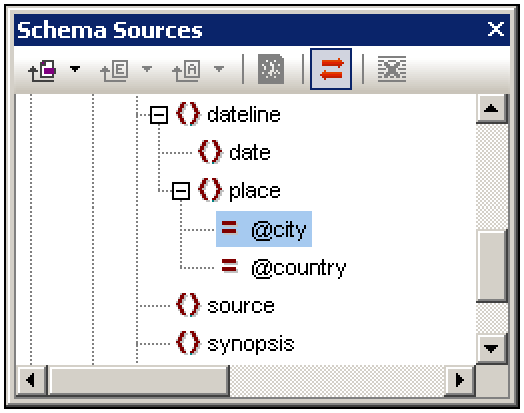
NanoNull Inc Jumps 3% on Release of New NanoPower Version
2006-04-01

NanoNull Shares Up 10% on Month Following New NanoPower Version
2006-04-25

NanoDiamonds Project to Go Ahead
2006-05-06

Notice that while the contents of the `date` children of `dateline` elements have been output, no contents have been output for the `place` children of `dateline`. This is because the `place` data is contained in the attributes `city` and `country` and attribute contents are not output when the attribute's parent element is processed.

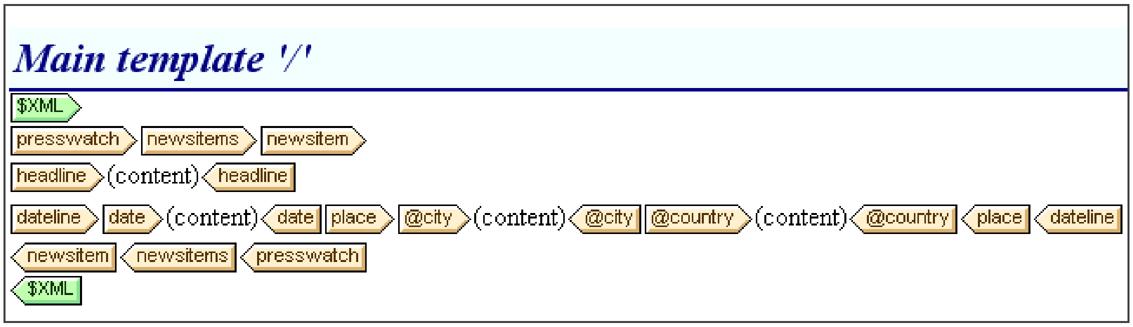
5. In Design View, go to the menu command **Authentic | Auto-Add Date Picker**, and toggle it off to deactivate the auto-addition of the date picker. (The icon will have no border when toggled off.) This step is required if the date picker is not to be inserted automatically when a node of type `xs: date` or `xs: dateTime` is inserted into the design (which you will do in the next step).
6. Drag the `date` element from the Schema Sources entry helper and drop it (create it as contents) in between the start and end tags of the `dateline` element.
7. Select the `city` attribute of the `dateline/place` element (*screenshot below*) in the Schema Sources entry helper.



8. Drag the `@city` attribute node into Design View, and drop it (create as contents) just after the end tag of the `date` element.
9. Drag the `@country` attribute node into Design View, and drop it (create as contents) just

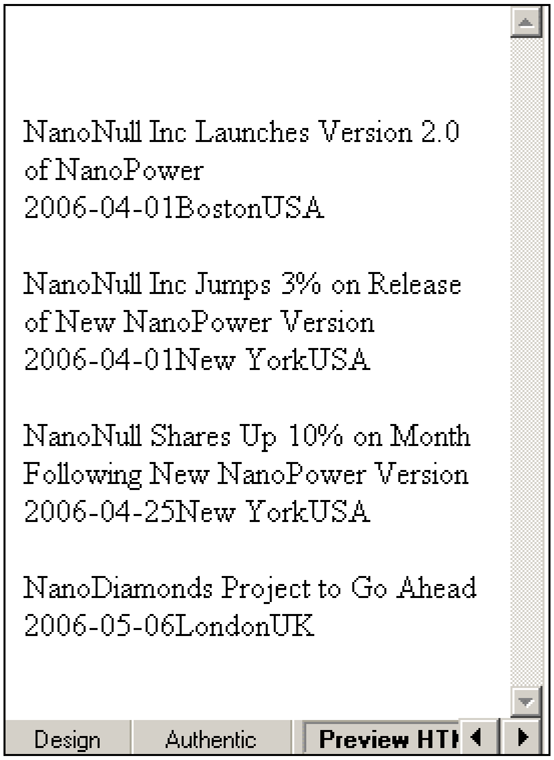
after the end tag of the `@city` attribute.

When you are done, the SPS design should look something like this:



Main template '/'
$XML
presswatch
newsitems
newsitem
headline (content) headline
dateline date (content) date place @city (content) @city @country (content) @country place dateline
newsitem newsitems presswatch
$XML

The HTML Preview will look like this:



NanoNull Inc Launches Version 2.0 of NanoPower
2006-04-01BostonUSA

NanoNull Inc Jumps 3% on Release of New NanoPower Version
2006-04-01New YorkUSA

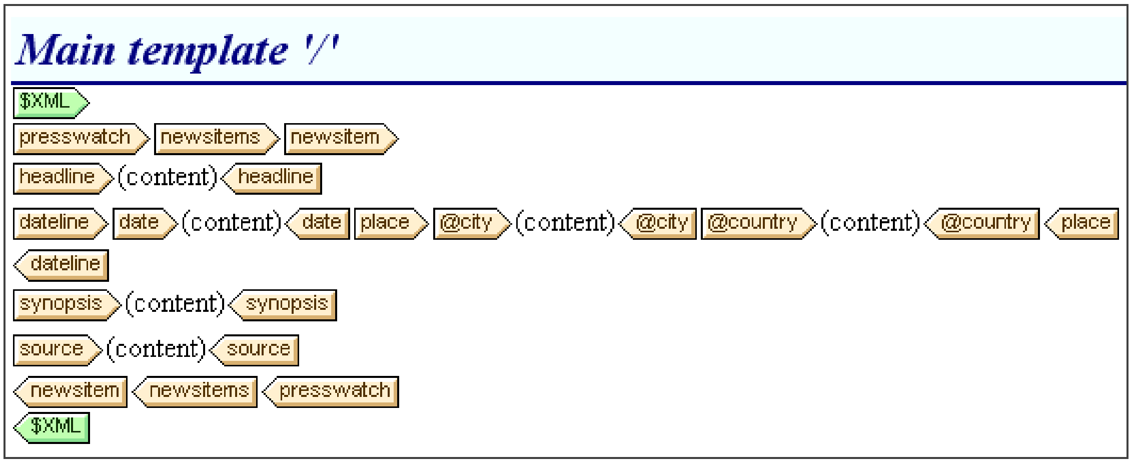
NanoNull Shares Up 10% on Month Following New NanoPower Version
2006-04-25New YorkUSA

NanoDiamonds Project to Go Ahead
2006-05-06LondonUK

Notice that the values of the `@city` and `@country` attributes are now included in the output.

Adding more dynamic content

The contents of elements and attributes from the XML data source can be inserted anywhere in the design using the method described above. To complete this section, add the `synopsis` and `source` elements to the design so that the design now looks like this:


Main template '/'
$XML
presswatch
newsitems
newsitem
headline (content) headline
dateline date (content) date place @city (content) @city @country (content) @country place
dateline
synopsis (content) synopsis
source (content) source
newsitem newsitems presswatch
$XML

Notice that the `synopsis` element has been placed before the `source` element, which is not the order in which the elements are in the schema. After you have added the `synopsis` and `source` elements to the design, check the HTML preview to see the output. This is an important point to note: that the order in which nodes are placed in the main template is how you specify the structure of the output.

Another important point to note at this stage is the form in which a node is created in the design. In the HTML preview, you will see that all the nodes included in the design have been sent to the output as text strings. Alternatively to being output as a text string, a node can be output in some other form, for example, as a table or a combo box. In this section, you have, by creating all the nodes as `( contents)` , specified that the output form of all nodes are text strings. In the section, Using Conditions, you will learn how to create a node as a combo box, and in the section, Using Global Templates and Rest-of-Contents, how to create a node as a (dynamic) table.

Make sure to save the file before moving to the next section.

5.3 Inserting Static Content

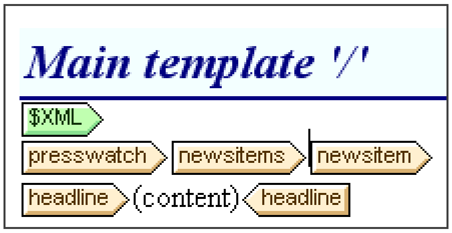
Static content is content you enter directly in the design—as opposed to content that comes from the XML source. A variety of static components can be entered in an SPS design. In this part of the tutorial, you will learn how to insert the following static components:

- An image
- A horizontal line
- Text

Inserting a static image

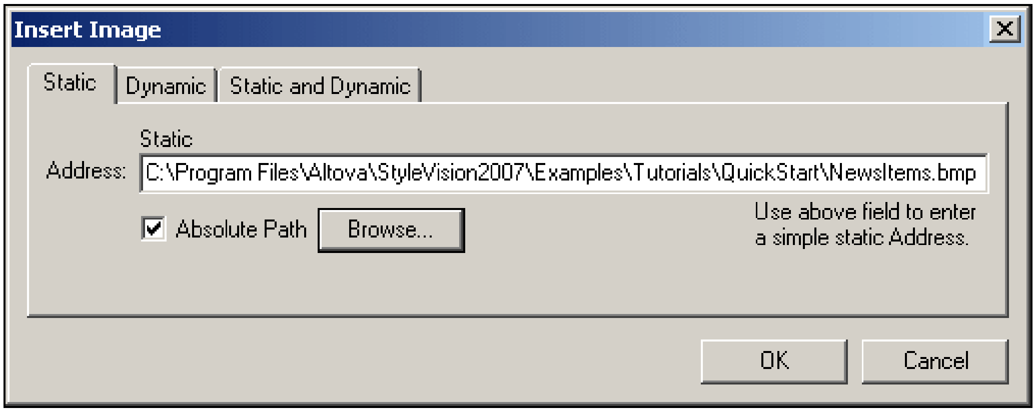
The static image to insert is `StyleVision2007/Examples/Tutorials/QuickStart/NewsItems.BMP`, which will be used as the header of the document. To insert this image at the head of the document, do the following:

1. Place the cursor between the start-tags of `newsitems` and `newsitem` (*screenshot below*).



Notice that the cursor is within the `newsitems` element but outside the `newsitem` element. It will therefore be inserted in the output once, at the start of processing of the `newsitems` element (because there is only one `newsitems` element defined in the schema).

2. Right-click, and select **Insert | Image**. The Insert Image dialog pops up (*screenshot below*).

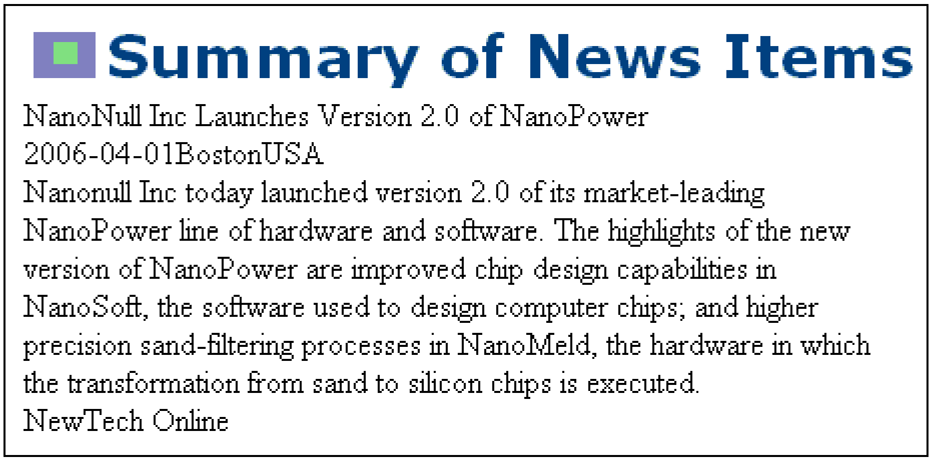


Address: C:\Program Files\Altova\StyleVision2007\Examples\Tutorials\QuickStart\NewsItems.bmp

☑ Absolute Path

3. In the Static tab, click the Absolute Path, then browse for the file `NewsItems.BMP` and select it.
4. Click **OK** to finish.

The HTML preview will look something like this:



Summary of News Items

NanoNull Inc Launches Version 2.0 of NanoPower
2006-04-01BostonUSA
Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.
NewTech Online

Inserting horizontal lines

The first horizontal line you will insert is between the document header and document body. Do this as follows:

1. Place the cursor immediately after the recently inserted static image.
2. Right-click, and select **Insert | Horizontal Line**. A horizontal line is inserted.

Set properties for the line as follows:

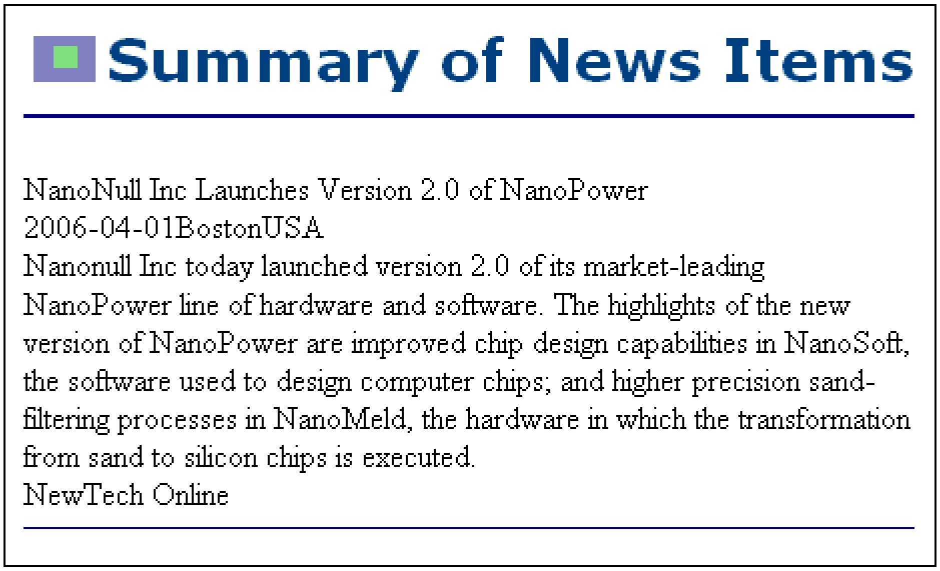
1. With the line selected in Design View, in the Properties entry helper, select the *line* component (in the Properties For column) and then the *HTML* group of properties.
2. Assign `color` and `size` properties for the line.
3. With the line selected in Design View, in the Styles entry helper, select the *line* component and then the *box* group of properties. Define a `margin-bottom` property of `12pt`.
4. Check the output in HTML Preview.

Now insert a horizontal line at the end of each news item. To do this the cursor would have to be placed immediately before the end-tag of the `newsitem` element. This will cause the line to be output at the end of each `newsitem` element.

Inserting static text

You have already added static text to your design. When you pressed the **Enter** key to obtain new lines (in the section Inserting Dynamic Content (from XML Source)), whitespace (static text) was added. In this section, you will add a few static text characters to your design.

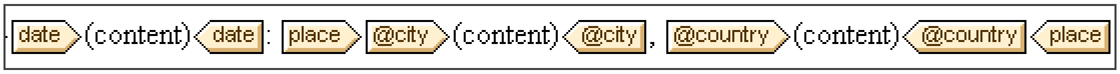
The SPS you have designed up to this point will produce output which looks something like this:



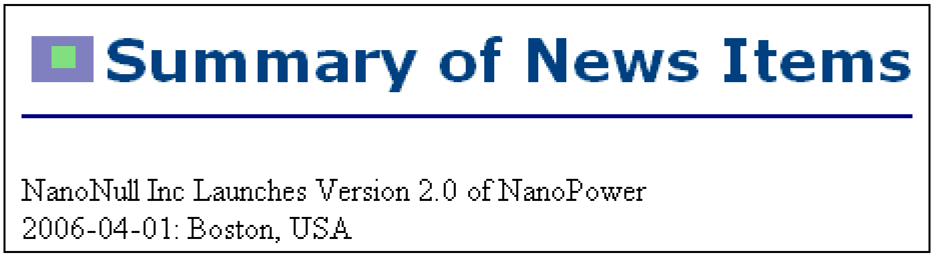
Summary of News Items

NanoNull Inc Launches Version 2.0 of NanoPower
2006-04-01BostonUSA
Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.
NewTech Online

Notice that in the output of the `dateline` element, the contents of the `date` element and `place/@city` and `place/@country` attributes are run together without spacing. You can add the spacing as static text. In the design, place the cursor after the `date` element and enter a colon and a space. Next, enter a comma and space after the `@city` attribute (*screenshot below*)


date (content) date : place @city (content) @city , @country (content) @country place

This part of the output will now look like this:

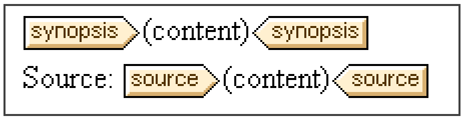


Summary of News Items

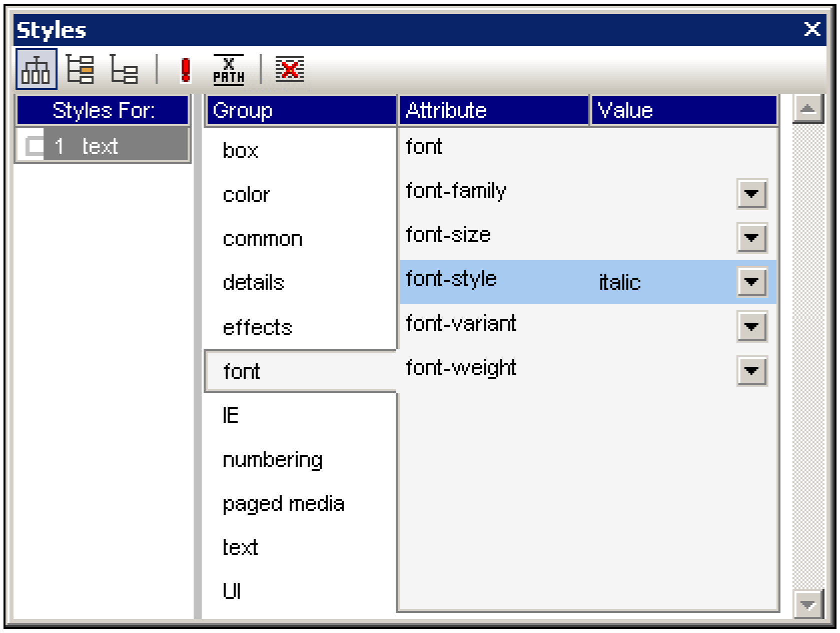
NanoNull Inc Launches Version 2.0 of NanoPower
2006-04-01: Boston, USA

Notice the colon, spacing and comma in the `dateline` output. All of these text items are static text items that were inserted directly in the design.

You will now add one more item of static text. In the design, type in the string "`Source: `" just before the start-tag of the `source` element (*screenshot below*).

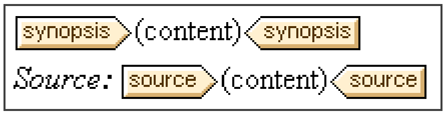


Formatting static text
To format static text, highlight the text to be formatted and specify local style properties. In the design, highlight the text "`Source:` " that you just typed. In the Styles entry helper (*screenshot below*), notice that the *1 text* component is selected. Now select the *font* group of properties, and, for the `font-style` property (*screenshot below*), select the `italic` option from the dropdown menu.



font-style italic

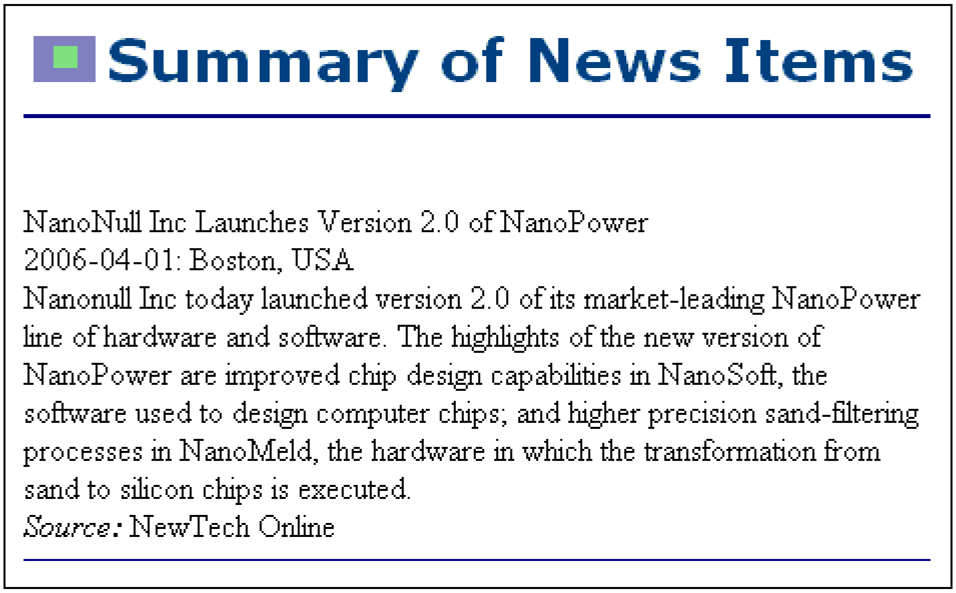
The static text (that is, the string "`Source: `") will be give an italic style in the design, and will look like this:



synopsis (content) synopsis

Source: source (content) source

The output will look like this in HTML Preview:



Summary of News Items

NanoNull Inc Launches Version 2.0 of NanoPower
2006-04-01: Boston, USA
Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.
Source: NewTech Online

If you think there is too little vertical space between the source item and the horizontal line separating two `newsitem` elements, then, in the design, insert a blank line between the source and the horizontal line (by pressing **Enter**).

After you are done, save the file.

In this section you have learned how to insert static content and format it. In the next section you will learn more about how design components can be formatted using CSS principles and properties.

5.4 Formatting the Content

StyleVision offers a powerful and flexible styling mechanism, based on CSS, for formatting components in the design. The following are the key aspects of StyleVision's styling mechanism:

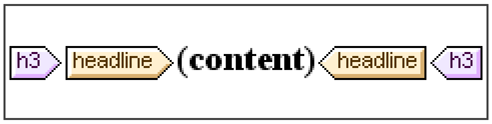
- CSS style rules can be defined for both block components and inline components.
- Predefined formats are block components that have inherent styles and can be used as wrappers for a group of adjacent components that need to be treated as a block. The inherent styles of these predefined formats can be overridden by styles you specify.
- Class attributes can be declared on components in the design, and the class can be used as a selector for external or global style rules.
- You can specify styles at three levels. These are, in increasing order of priority: (i) style rules in external stylesheets, (ii) global style rules, and (iii) local style rules.

In this section, you will learn how to:

- Assign predefined formats
- Assign a component a class attribute
- Define styles in an external CSS stylesheet and add this stylesheet to the style repository of the SPS
- Define global style rules
- Define local styles for a selection of multiple design components
- Define local styles for a single component

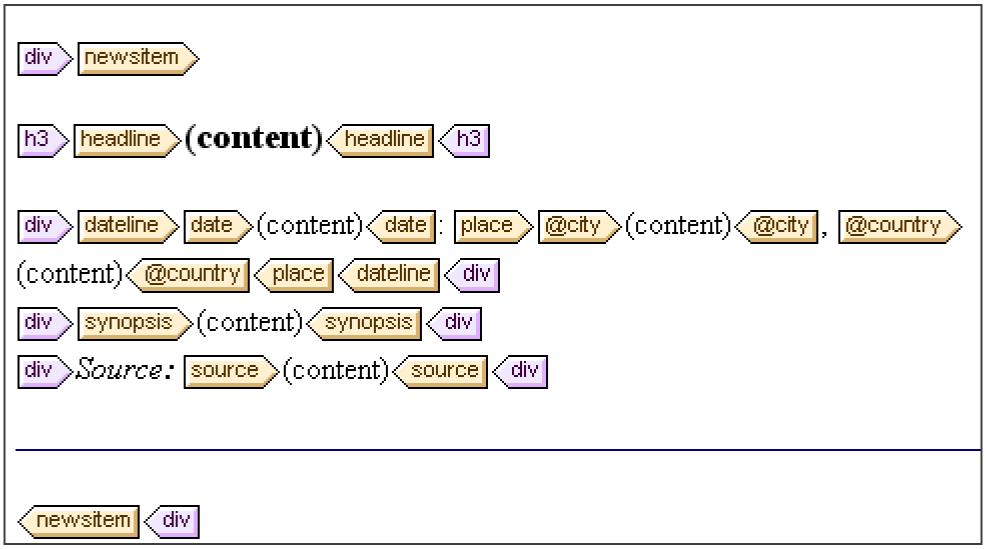
Assigning predefined formats

One reason to assign a predefined format is to give a component the inherent styling of that predefined format. In the design, select the `headline` element and then select **Insert | Format | Heading 3 (h3)** (alternatively use the Predefined Formats combo box in the toolbar). The predefined format tags are created around the `headline` element (*screenshot below*).



Notice that the font properties of the contents change and that vertical spacing is added above and below the predefined format. These property values are inherent in the `h3` predefined format.

Another use of predefined formats is to group design components in a block so that they can be formatted as a block or assigned inline properties as a group. The most convenient predefined property for this purpose is the `div` predefined format, which creates a block without spacing above or below. In your design, assign the `newsitem`, `dateline`, `synopsis`, and `source` nodes separate `div` components. Your design should look something like the screenshot below. Note that the static text "`Source:` " is also included in the `div` component that contains the `source` element, and that the entire `newsitem` element is inside a `div` component.

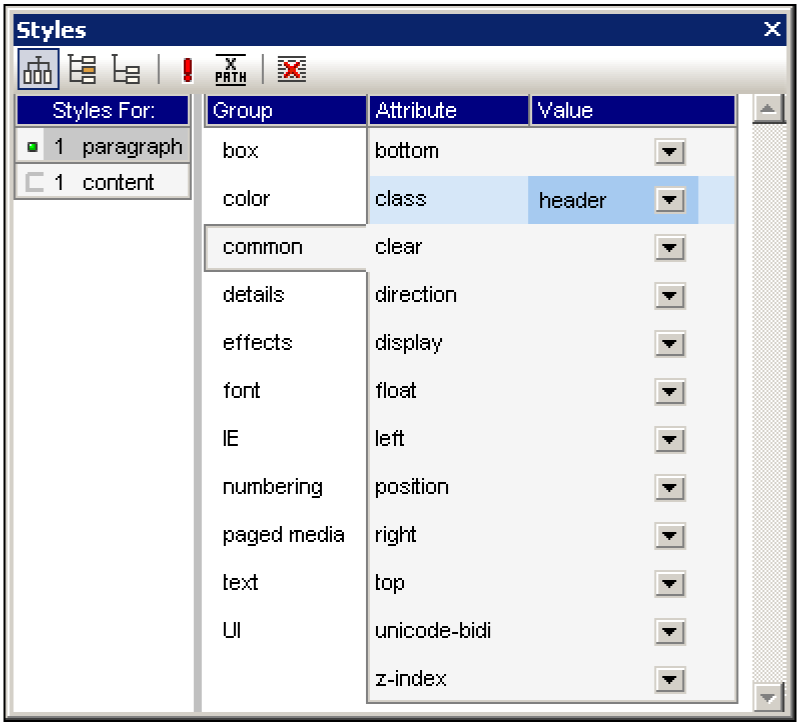


You have now grouped components together in different `div` blocks. Later in this section, you will learn how to assign styles to such blocks of grouped components.

Assigning components to class attributes

A style rule can be defined for a class of components. For example, all headers can be defined to have a set of common properties (for example, a particular font-family, font-weight, and color). To do this you must do two things: (i) assign the components that are to have the common properties to a single class; (ii) define the styling properties for that class.

In your design, select the `h3` tag, and in the Styles entry helper, select *1 paragraph* (to select the predefined format), and the *common* group of properties. Double-click in the Value field of the `class` property and enter `header`.



Styles For:
1 paragraph
1 content

Group	Attribute	Value
box	bottom	
color	class	header
common	clear	
details	direction	
effects	display	
font	float	
IE	left	
numbering	position	
paged media	right	
text	top	
UI	unicode-bidi	
	z-index	

This particular instance of the `h3` format is now assigned to a class named `header`. When you define styling properties for the `header` class (styles from an external stylesheet or global SPS styles), these properties will be applied to all components in the SPS that have the `header` class.

Adding an external CSS stylesheet to the style repository

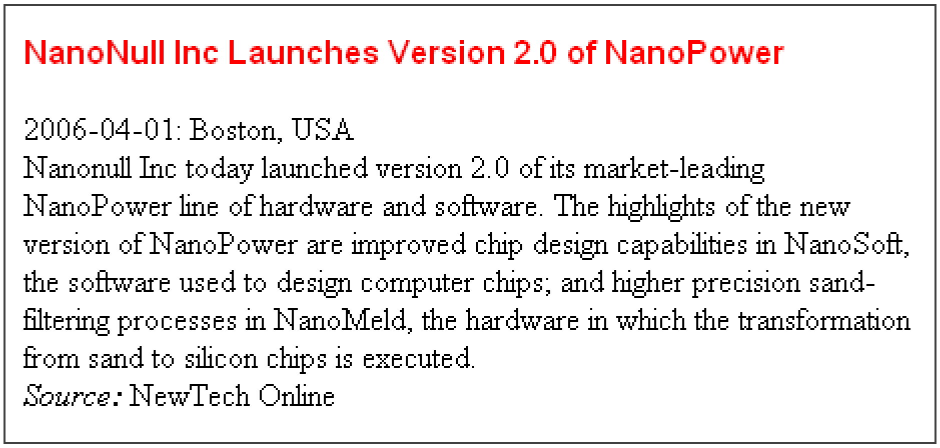
Style rules in an external CSS stylesheet can be applied to components in the SPS design. External stylesheets must, however, first be added to the style repository in order for rules in them to be applied to components. In the Style Repository entry helper (in Design View), do the following:

1. Select the `External` item.
2. Click the **Add** button in the toolbar. This pops up the Open dialog.
3. Browse for the file `StyleVision2007/Examples/Tutorials/QuickStart/QuickStart.css` and click **Open**.

The stylesheet is added to the style repository. It contains the following rules that are relevant at this stage:

```
.header    {
            font-family: "Arial", sans-serif;
            font-weight: bold;
            color: red;
           }

h3         {
            font-size: 12pt;
           }
```

The style rules for the `header` class and `h3` element are combined and produce the following HTML output for the `headline` element.



NanoNull Inc Launches Version 2.0 of NanoPower

2006-04-01: Boston, USA
Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.
Source: NewTech Online

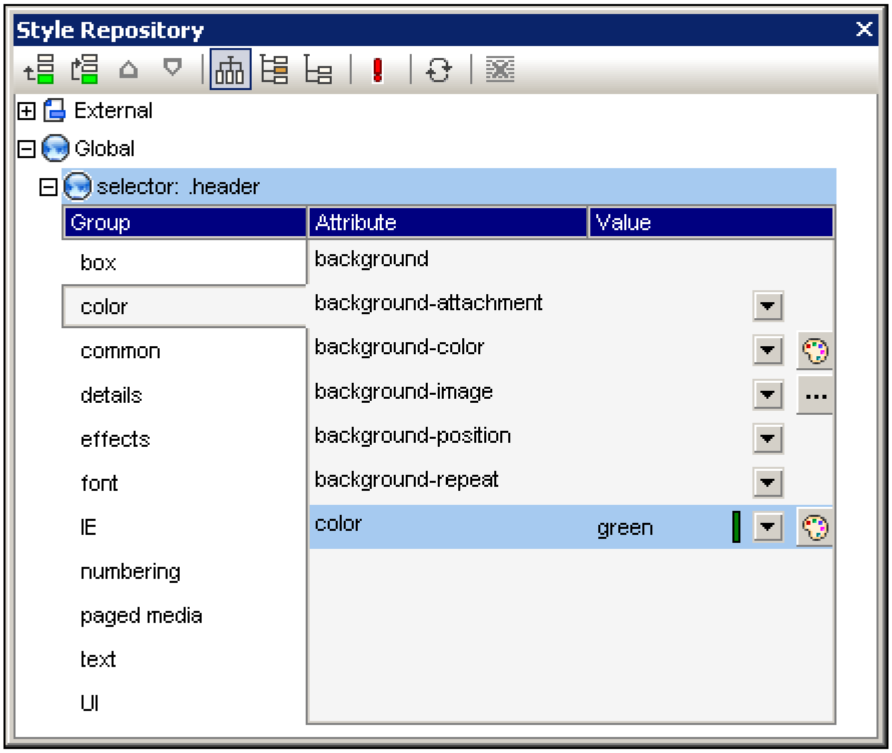
Defining global style rules

Global style rules can be defined for the entire SPS using CSS selectors. The rules are defined directly in the Style Repository entry helper. Create a global style rule for the `header` class as follows:

1. With Design View active, in the Style Repository entry helper, select the Global item.
2. Click the **Add** button in the toolbar. This creates an empty rule for the wildcard selector

(*), which is highlighted.

3. Type in `.header` to replace the wildcard as the selector.
4. In the *color* group of properties, select `green` from the dropdown list of the `color` property values (*screenshot below*).



Style Repository

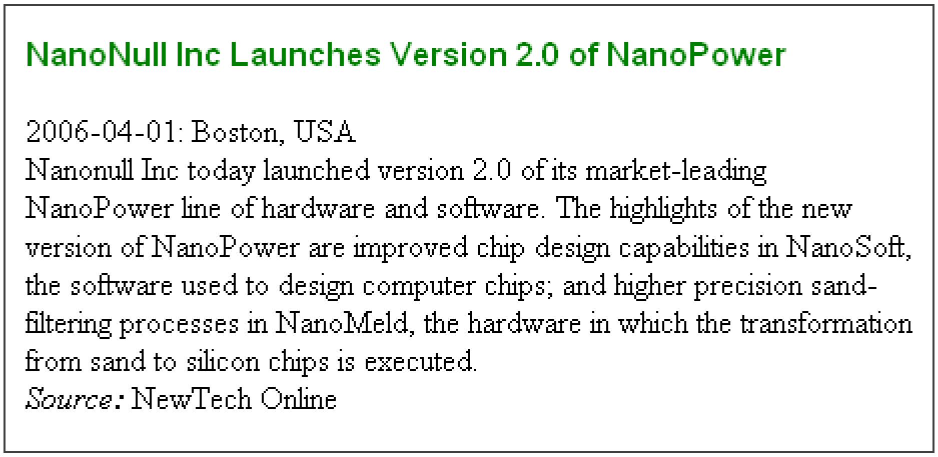
External

Global

selector: .header

Group	Attribute	Value
box	background	
color	background-attachment	
common	background-color	
details	background-image	
effects	background-position	
font	background-repeat	
IE	color	green
numbering		
paged media		
text		
UI		

Where the global style rule defines a property that is also defined in the external stylesheet (the `color` property), the property value in the global rule takes precedence. In the HTML preview, the contents of the headline will therefore be green. Other property definitions from the external stylesheet (not over-ridden by a property in a global style rule) are retained (in this case, `font-family` and `font-weight`).



NanoNull Inc Launches Version 2.0 of NanoPower

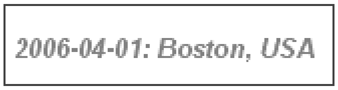
2006-04-01: Boston, USA
Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.
Source: NewTech Online

Defining local styles for multiple components at once

Local styles can be defined for multiple components at once. In your design, to specify that the entire text contents of a news item should have Arial as its font, click the `div` component surrounding the `newsitem` element and, in the Styles entry helper, in the Styles For column, select `1 paragraph`. Then, in the *font* group of properties, assign `Arial` as the `font-family`. This property setting will be inherited by all five descendant predefined formats.

Now, in the design, select the three `div` components surrounding the `dateline`, `synopsis`, and `source` nodes (by keeping the **Shift** key pressed as you click each `div` component). In the Styles entry helper, select `3 paragraphs`, then the *font* group of properties, and set a `font-size` of `10pt`. (The `h3` component was not selected because it already has the required `font-size` of `12pt`.)

Finally, in the design, select the `div` component surrounding the `dateline` element. In the Styles For column of the Styles entry helper, select `1 paragraph`. In the *font* group of properties, set `font-weight` to `bold` and `font-style` to `italic`. In the *color* group of properties, set `color` to `gray`. The output of the dateline will look like this



2006-04-01: Boston, USA

Notice that the styling defined for the `div` component has been applied to the static text within the `div` component as well (that is, to the colon and the comma).

Defining local styles for a single component

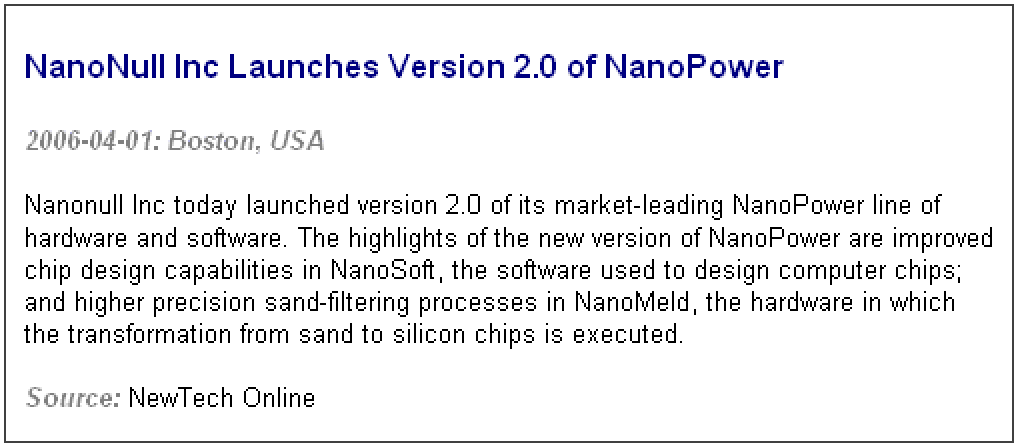
A local style defined on a single component overrides all other styles defined at higher levels of the SPS for that component. In the design, select the `headline` element and assign it a color of navy (`color` property in the *color* group of properties). The locally defined property (`color: navy`) overrides the global style for the `.header` class (`color: green`).

Select the `div` component surrounding the `source` element. In the Styles entry helper, with the `1 paragraph` item in the Styles For column selected, set the `color` property (in the *color* group of properties) to `gray`. In the *font* group of properties, set `font-weight` to `bold`. These values are applied to the static text. Remember that in the last section the static text "`Source:` " was assigned a `font-style` value of `italic`. The new properties (`font-weight: bold` and `color: gray` are additional to the `font-style: italic` property).

Now, in Design View, select the `( content)` placeholder of the `source` element. In the Styles For column, with *1 content* selected, set the `color` property (in the *color* group of properties) to `black`. In the *font* group of properties, set `font-weight` to `normal`. The new properties are set on the `contents` placeholder node of the `source` element and override the properties defined on the `div` component (*see screenshot below*).

Completing the formatting

To complete the formatting in this section, select the `div` component on the `synopsis` element and, in the Predefined Formats combo box in the toolbar, select `p`. This gives the block the inherent styles of the HTML's `p` element. The HTML preview should now look something like this:



NanoNull Inc Launches Version 2.0 of NanoPower

2006-04-01: Boston, USA

Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.

Source: NewTech Online

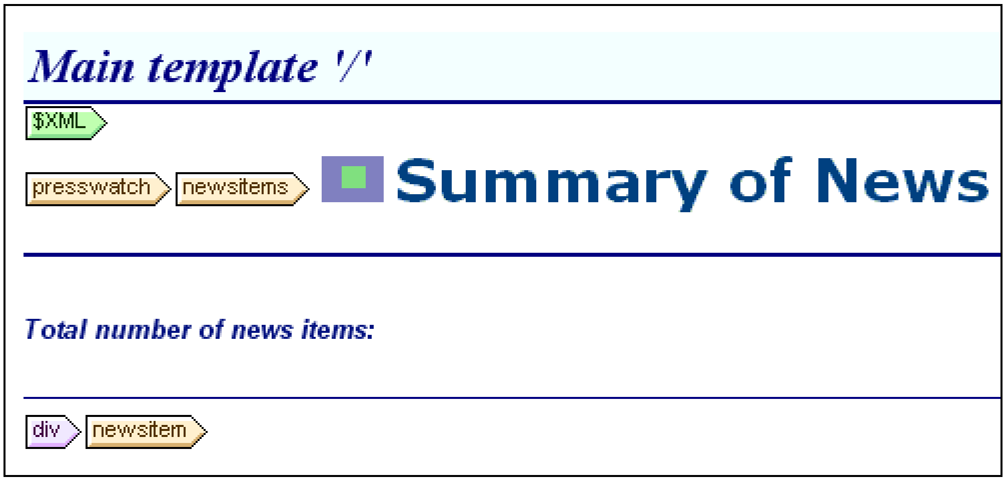
After you are done, save the file.

5.5 Using Auto-Calculations

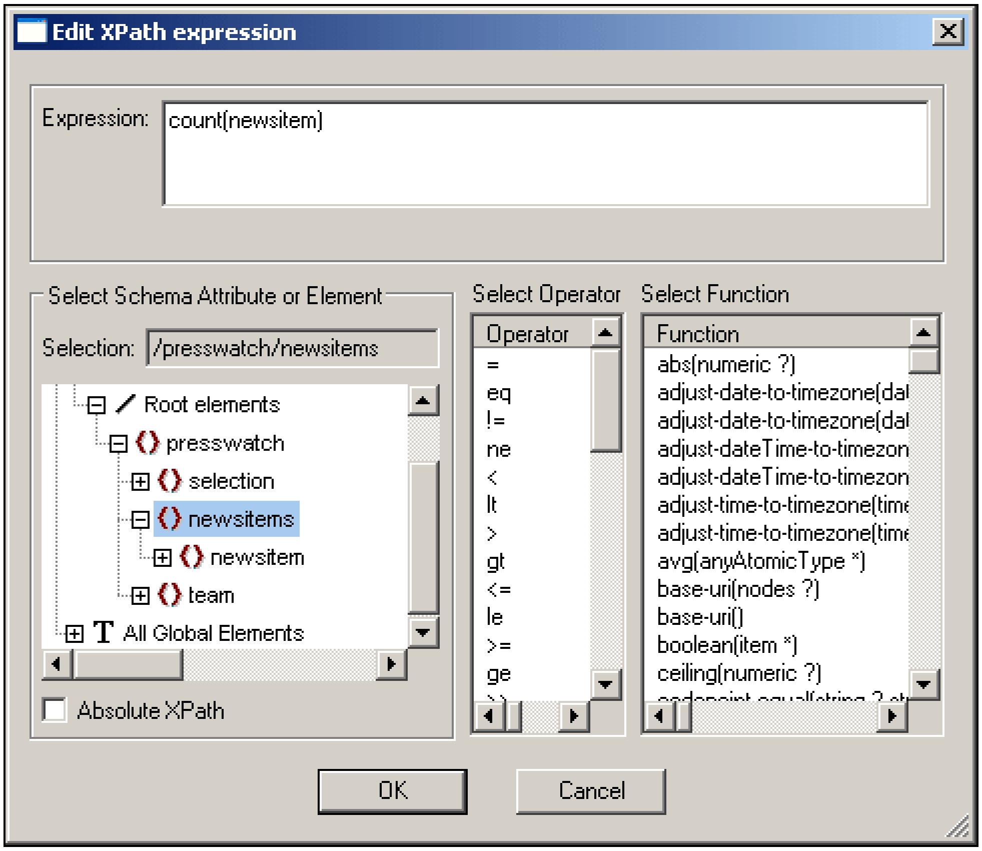
Auto-Calculations are a powerful mechanism for providing additional information from the available XML data. In this section you will add two pieces of information to the design: the total number of news items and the time period covered by the news items in the XML document. Neither piece of information is directly available in the XML document but has to be calculated or manipulated from the available data.

Counting the news item nodes
In the design, do the following:

1. Create space, as shown in the screenshot below, for a line of static text (on which the Auto-Calculation will also be placed). Use the **Return** key to add new lines and insert a horizontal line below the space you create (*see screenshot*).



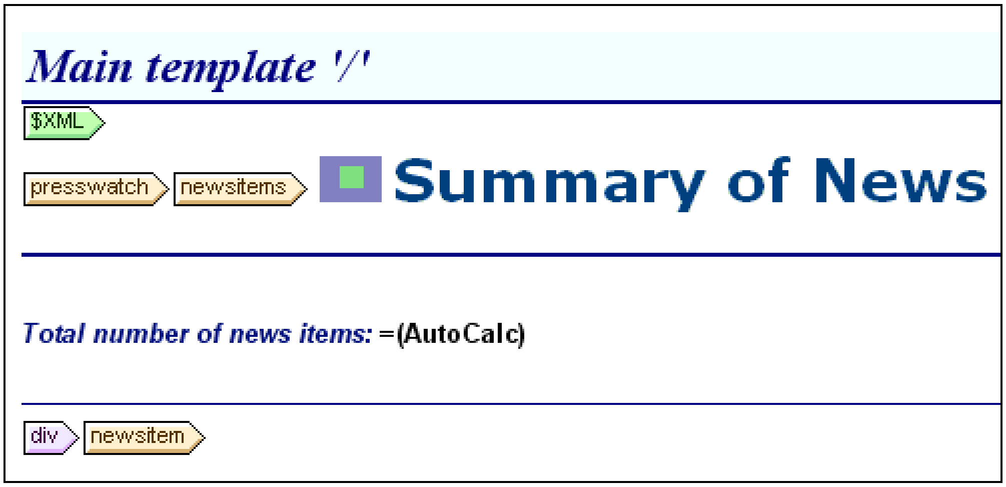
2. Type in the static text "`Total number of news items: `" as shown in the screenshot above.
3. Apply local styling of your choice to the static text. Do this as described in the section Formatting the Content.
4. Place the cursor after the colon and select **Insert | Auto-Calculation | Value**. This pops up the Edit XPath Expression dialog (*screenshot below*). (Alternatively, you can right-click and select the command in the context menu.)



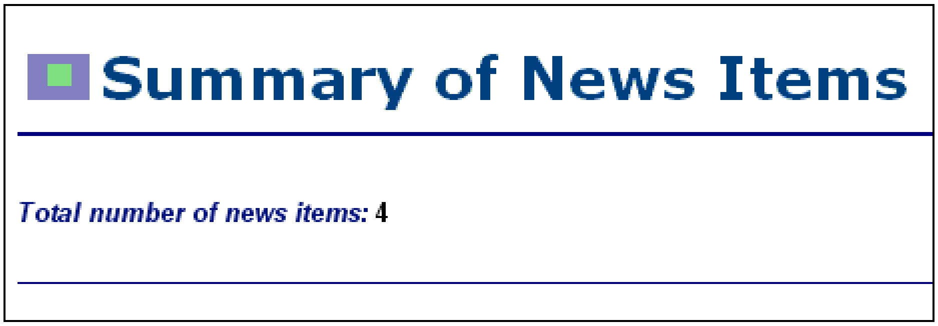
Expression: count(newsitem)

Selection: /presswatch/newsitems

5. In the schema tree, note that the context node is `newsitems`, which is highlighted. Now, in the Expression text box either type in the expression `count( newsitem)` or build the expression using the entry helpers. (Double-click the `count` function to enter it, then place the cursor within the parentheses of the function and double-click the `newsitem` node in the schema tree.
6. Click **OK** to finish. The Auto-Calculation is inserted in the design at the cursor location (*screenshot below*). Format the Auto-Calculation using local styles.



Your HTML output will look like this:



Summary of News Items

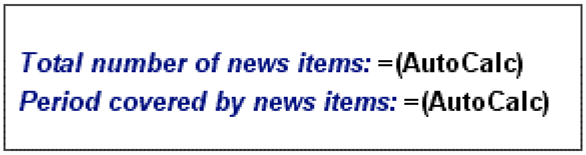
Total number of news items: **4**

Displaying the period covered by news items

The period covered by the news items can be obtained by getting the date of the earliest news item and the date of the latest news item. This can be achieved with XPath expressions like those given below. The first expression outputs the contents of the `date` node. The second expression is a refinement, outputting just the month and year values in the `date` node. You can use either of these.

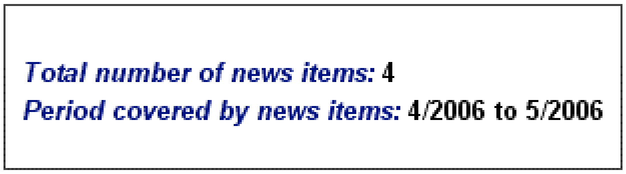
- `concat( min( //date), ' to ', max( //date)) .`
- `concat( month-from-date( min( //date)), '/', year-from-date( min( //date)), ' to ', month-from-date( max( //date)), '/', year-from-date( max( //date)))`

In the design, insert the static text and Auto-Calculation as shown in the screenshot below. Apply whatever local styling you like.



Total number of news items: =(AutoCalc)
Period covered by news items: =(AutoCalc)

The HTML preview will look something like this:



Total number of news items: **4**
Period covered by news items: **4/2006 to 5/2006**

After you are done, save the file.

5.6 Using Conditions

If you look at `QuickStart.xml`, you will see that each `newsitem` element has a `metainfo` child element, which in turn can contain one or more `relevance` child elements. In the SPS design, you can create a combo box that has a dropdown list which you can populate with unique `relevance` element values. When the Authentic View user selects an item from the dropdown list in the combo box, that item can be passed as a value to a node in the XML document. A condition can test what the user selection is (by looking up that node) and provide appropriate processing (displays) for each user selection. In this section, you will create a conditional template that displays those news items that have a `relevance` element that matches the user selection.

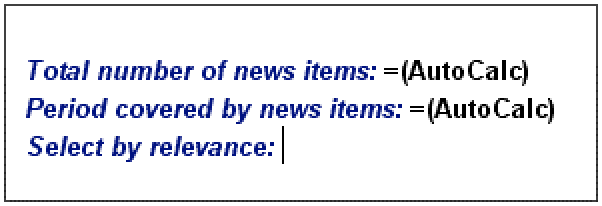
We will proceed as follows:

1. Create a combo box in which the Authentic View user can select the `byrelevance` value. The values in the dropdown list of the combo box are obtained by using an XPath expression, which dynamically compiles a list of all unique `relevance` node values.
2. Insert a condition around the `newsitem` element. This condition selects all `newsitem` elements that have a `relevance` element with content matching the content of the `byrelevance` node. The content that is surrounded by a branch of a condition is know as a conditional template.
3. Within the conditional template, list each `relevance` node of that news item.
4. Highlight the `relevance` element (in the list of `relevance` elements) that matches the `byrelevance` element. This is done by creating a condition to select such `relevance` elements and then applying special formatting to this conditional template.
5. In the condition for the `newsitem` element, insert a branch that selects all news items.

Creating the combo box to select unique node values

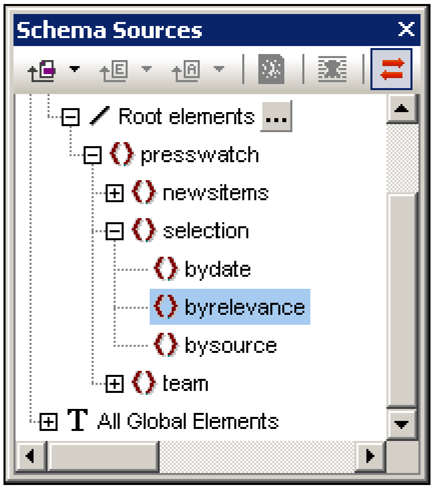
In the XML document, the node that will contain the user selection is `/presswatch/selection/byrelevance`. This is the node you will create as the combo box. Do this as follows:

1. Insert the static text "`Select by relevance:` " at the head of the document and just below the second Auto-Calculation (*screenshot below*).

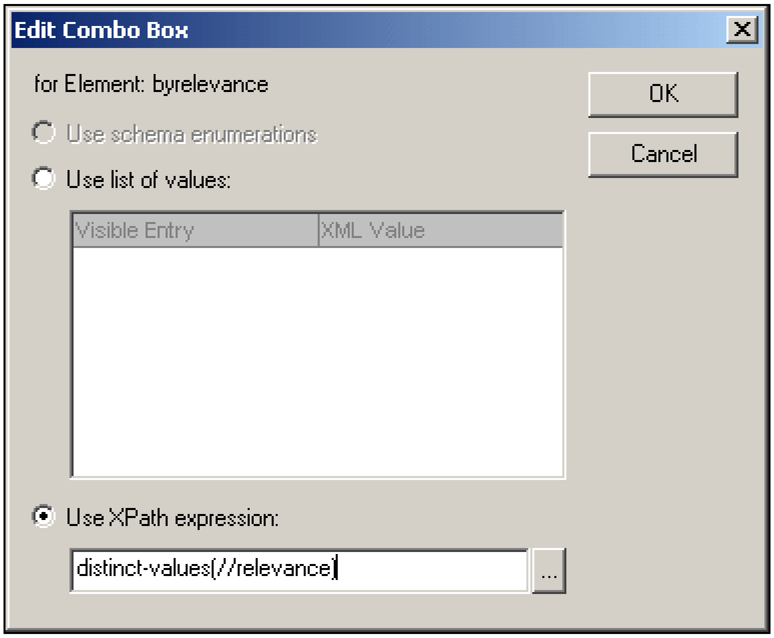


Total number of news items: =(AutoCalc)
Period covered by news items: =(AutoCalc)
Select by relevance:

2. Drag the `byrelevance` node from the Schema Sources entry helper (*screenshot below*), and drop it after the newly entered static text.



3. In the context menu that appears, select Create Combo Box. This pops up the dialog shown below.


Edit Combo Box

for Element: byrelevance

○ Use schema enumerations

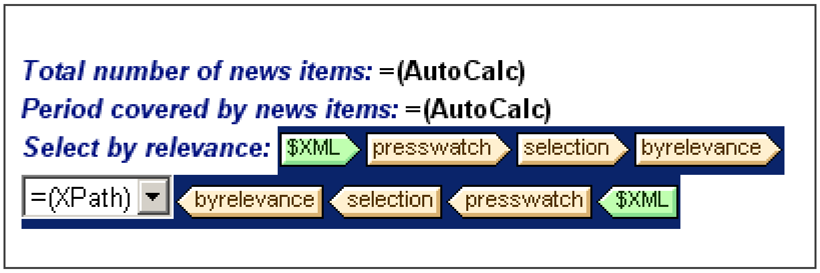
○ Use list of values:

Visible Entry	XML Value

● Use XPath expression:

distinct-values(//relevance)

4. In the Edit Combo Box dialog (*screenshot above*), select Use XPath Expression, and enter the XPath expression: `distinct-values(//relevance)` . This expression selects unique values of all `relevance` elements in the XML document.
5. Click **OK** to finish. The combo box is inserted and the design will look something like this:


Total number of news items: =(AutoCalc)

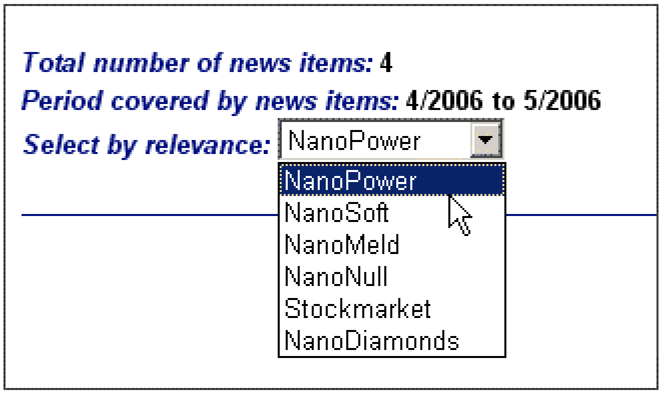
Period covered by news items: =(AutoCalc)

Select by relevance: $XML presswatch selection byrelevance

=(XPath) byrelevance selection presswatch $XML

6. Switch to Authentic View. When you click the dropdown arrow of the combo box, notice

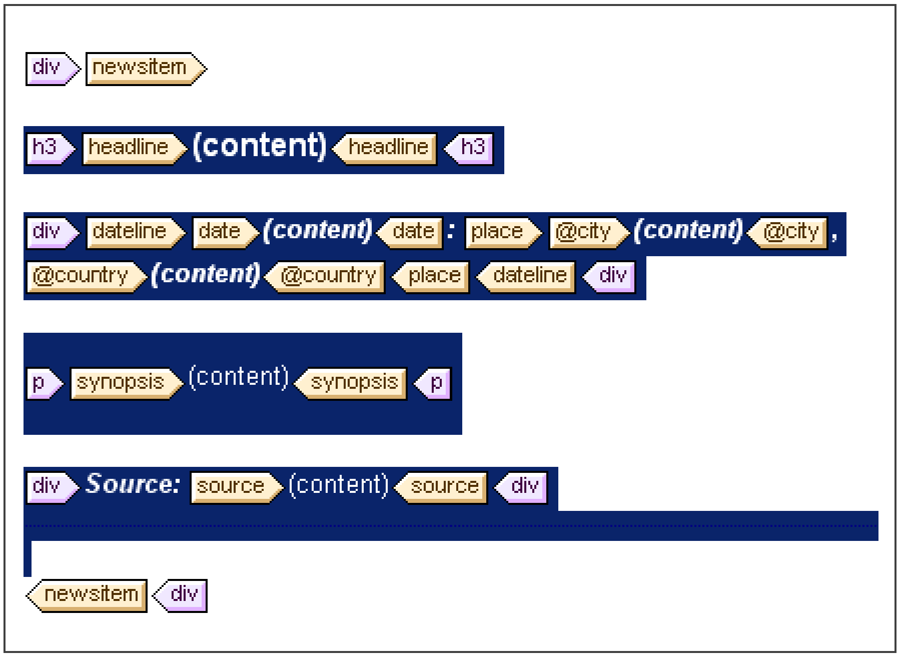
that the list contains the unique values of all `relevance` nodes (*screenshot below*). Check this against the XML document. This is a dynamic listing that will be augmented each time a new `relevance` value is added to the XML document.



Total number of news items: 4
Period covered by news items: 4/2006 to 5/2006
Select by relevance: NanoPower

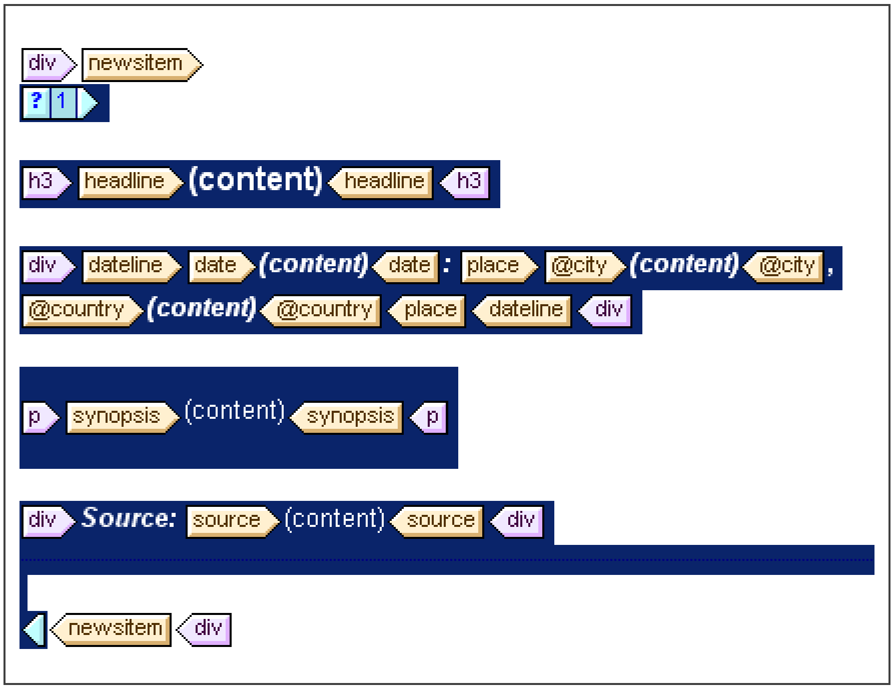
Inserting a condition to display news items having the selected `relevance`
The condition selects `newsitem` elements that have a `metainfo/relevance` element with a value that is the same as that selected by the user (and passed to the `/presswatch/selection/byrelevance` element). Insert the condition as follows:

1. Select the contents of the `newsitem` part of the design which is to be contained inside the condition (highlighted in the screenshot below).



2. Select the menu command (or context menu command) **Insert | Condition**. This pops up the Edit XPath expression.
3. Enter the expression `metainfo/relevance=/presswatch/selection/byrelevance`. This expression evaluates to true when the value of the `metainfo/relevance` descendant of the current `newsitem` is the same as the value of the `/presswatch/selection/byrelevance` element (the user selection).
4. Click **OK**. The condition is created around the contents of the `newsitem` element (

screenshot below).

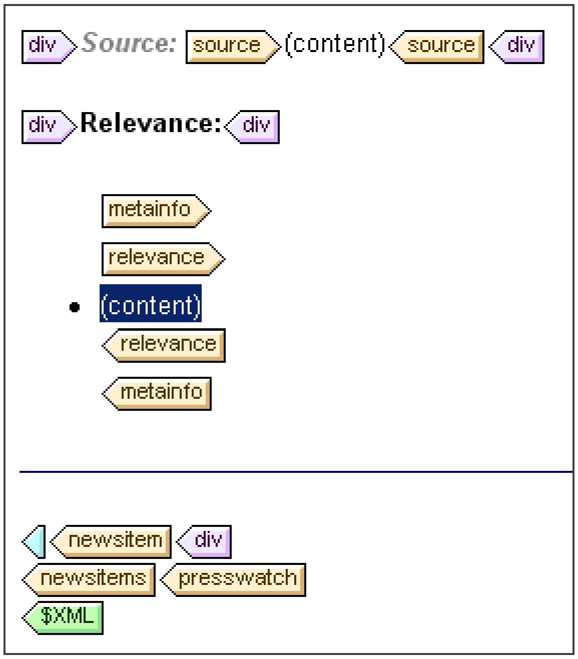


Note that there is a single branch in this condition. News items for which the condition test evaluates to true are displayed, those for which the condition test does not evaluate to true are not displayed. The condition in this case, therefore, works as a filter. Later in this section, you will add a second branch to this condition.

Inserting the `relevance` node as a list

In order to display the `relevance` nodes of each `newsitem` element, insert them in the design as follows (`see screenshot below`):

1. Create some vertical space below the `div` component for the `source` element and within the end-tag of the conditional template.
2. Type in the static text "`Relevance:` " and create a predefined format of `div` around it (highlight the static text and insert the predefined format).
3. Drag the `relevance` element from the Root elements tree in the Schema Sources entry helper and drop it into the design below the static text `Relevance:` .
4. Create it as a list. (In the context menu that pops up when you drop the node in the design, select Bullets and Numbering, and then select the desired list format.)
5. Apply text formatting to the contents of the list. When you are done, the design should look something like this:

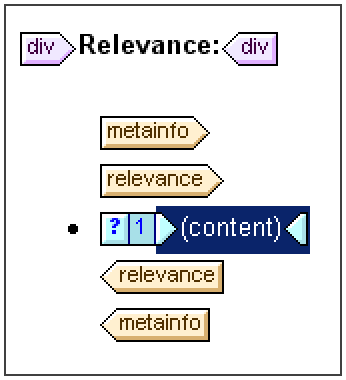


Now, in Authentic View, check the results for different selections of `relevance`; use the combo box to change the selection.

Making the selected `relevance` element bold

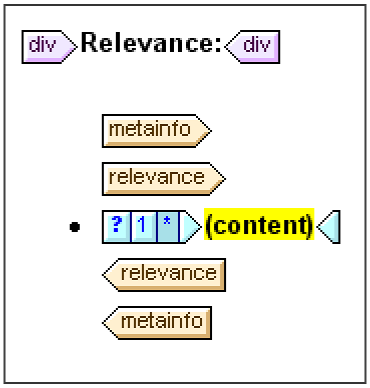
Some news items have more than one `relevance` element. In such cases, the design would be improved if the relevance that matches the user-selection were visually highlighted while the others were not. You can do this in the following way:

1. Select the `relevance` element in the design.
2. Insert a condition, giving it an XPath expression of: `. =/presswatch/selection/byrelevance`. This creates a condition with a single branch (*screenshot below*) that selects `relevance` elements that match the `byrelevance` element.



3. Select the `contents` placeholder and give it a local formatting (in the Styles entry helper) of bold (*font* group) and yellow background-color (*color* group).
4. Right-click the condition and, from the context menu, select **Condition | Copy Branch**.
5. In the Edit XPath Expression dialog that pops up, check the Otherwise check box (below the expression text box).
6. Click **OK** to finish. A new branch (`Otherwise`) is created (*screenshot below*). This

condition branch selects all `relevance` elements that do not match the `byrelevance` element.

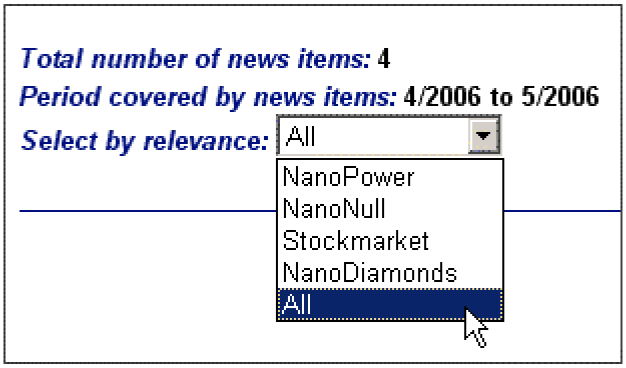


7. Notice that the contents of the `Otherwise` branch are a copy of the first branch; the `contents` placeholder is bold and has a yellow background. Remove this formatting (bold and background-color) from the `contents` placeholder.

You have put a condition with two branches (each with its conditional template) that carries out the following test on each `relevance` element: (i) if the contents of `relevance` match those of `/presswatch/selection/byrelevance`, then the contents of `relevance` are displayed bold and with a yellow background. Otherwise (the second branch) they are displayed normal. Check this in Authentic View.

Modifying the combo box and inserting a second condition branch
In the combo box where the Authentic View user selects a `byrelevance` value, there is no dropdown list option for selecting all news items. To include this option do the following:

1. In Design View, select the combo box.
2. In the Properties entry helper, with *combobox* selected in the Properties For column, click the **Edit** button of the `content origin` property (in the *combo box* group of properties).
3. In the Edit XPath Expression dialog that pops up, modify the XPath expression from `distinct-values(//relevance)` to `distinct-values(//relevance), 'All'`. This adds the string `All` to the sequence of items returned by the XPath expression.
4. Check the dropdown list of the combo box in Authentic View (*screenshot below*).



Total number of news items: 4

Period covered by news items: 4/2006 to 5/2006

Select by relevance: All

Now if the user selection is `All`, then this value (`All`) is passed to the node `/presswatch/selection/byrelevance`. The idea is that when the `byrelevance` node contains the value All, all news items should be displayed.

The condition that displays the news item template has a single branch with the expression `metainfo/relevance=/presswatch/selection/byrelevance`. Since no `metainfo/relevance` node has the value `All`, no news item will be displayed when `All` is te value of the `byrelevance` node. What you have to do is create a second branch for the condition, which will test for a value of `All`. By creating the news item template within this branch, you will be outputting the news item if the test is true. Do this as follows:

1. In Design View, select the news item condition.
2. Right-click the condition and, from the context menu, select **Condition | Copy Branch**.
3. In the Edit XPath Expression dialog that pops up, enter the expression: `/presswatch/selection/byrelevance='All'`.
4. Click **OK** to finish. A second branch is created.

The second branch has as its contents the same template as the first branch. What the second branch does is output the news item template if the user selection is `All`.

After you have completed this section, save the design.

5.7 Using Global Templates and Rest-of-Contents

Global templates are useful for specifying the processing of an element globally. This enables the rules of the global template (defined in one location) to be used at multiple locations in the stylesheet. A global template can be used in two ways:

- The rules of the global template can be copied to the local template.
- A local template (in the main template) can pass processing of that node to the global template, after completing which processing resumes in the main template; in this case, the global template is said to be invoked or used.

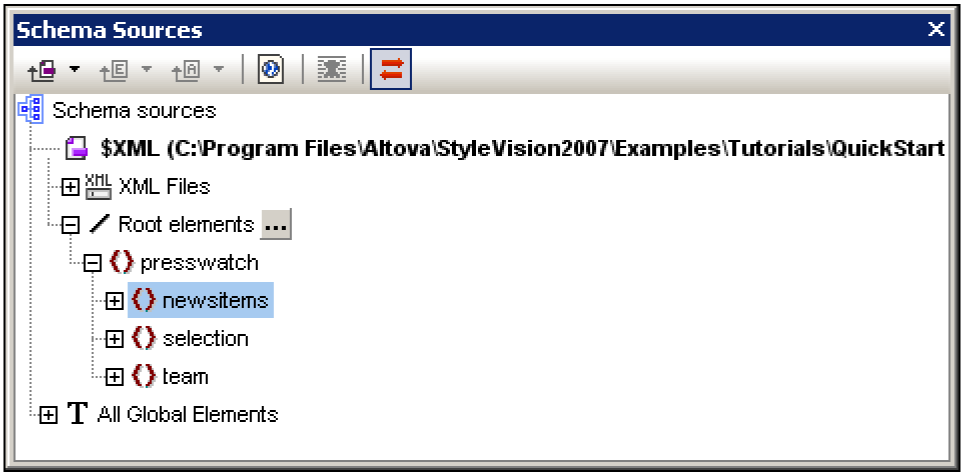
There are two mechanisms that are used to invoke a global template from the main template:

- A local template references a global template.
- A (rest-of-contents) instruction in the main template applies templates to the descendant elements of the current element (that is, to the rest-of-contents of the current element). If a global template exists for one of the descendant elements, the global template is applied for that element. Otherwise the built-in template for elements is applied.

In this section, you will create a design for the team-members' template using the rest-of-contents instruction and a global template for the global element member.

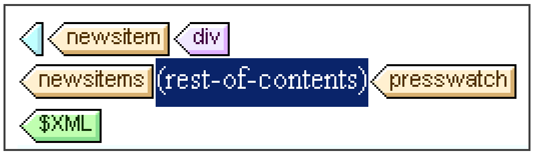
Inserting the rest-of-contents instruction

The broad structure of the schema is shown in the screenshot below.

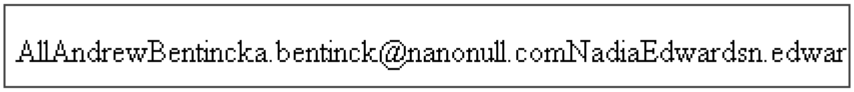


The document element presswatch contains three children: (i) selection; (ii) newsitems; and (iii) team. The main template you have created this far processes the /presswatch element. Within the presswatch element, only the newsitems element is processed. The selection and team elements are not processed within the presswatch element (although selection has been processed within the newsitems element). Inserting the rest-of-contents instruction within presswatch will therefore cause the selection and team elements to be processed.

Insert the rest-of-contents instruction in the design by placing the cursor between the end-tags of newsitems and presswatch, and selecting the menu command or context menu command **Insert | Rest of Contents**. The rest-of-contents placeholder is inserted (*screenshot below*).



If you look at the HTML preview, you will see a string of text (*screenshot below*):

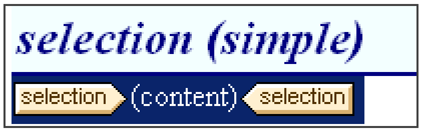

AllAndrewBentincka.bentinck@nanonull.comNadiaEdwardsn.edwar

This string is the result of the application of the built-in templates to the `selection` and `team` elements. The built-in template for elements processes child elements. The built-in template for text nodes outputs the text in the text node. The combined effect of these two built-in templates is to output the text content of all the descendant nodes of the `selection` and `team` elements. The text `All` comes from `selection/byrelevance`, and is followed by the text output of `team/member` descendant nodes, `first`, `last`, `email`, in document order. Note that the `id` attribute of `member` is not output (because, as an attribute, it is not considered a child of `member`).

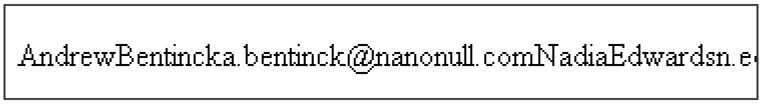
Creating a global template for `selection`

Since the content of `selection` is not required in the output, you should create an empty global template for `selection` so that its contents are not processed. Do this as follows:

1. In Design View, right-click `selection` in the Global Elements tree in the Schema Sources entry helper.
2. In the context menu that pops up, select **Make Global / Remove Global**. A global template for `selection` is created (*screenshot below*).



3. In the global template, click the `contents` placeholder.and press the **Delete** key of your keyboard. The contents placeholder is deleted.
4. Check the HTML preview. The text `All` is no longer present in the line of text output by the built-in templates (*screenshot below*).

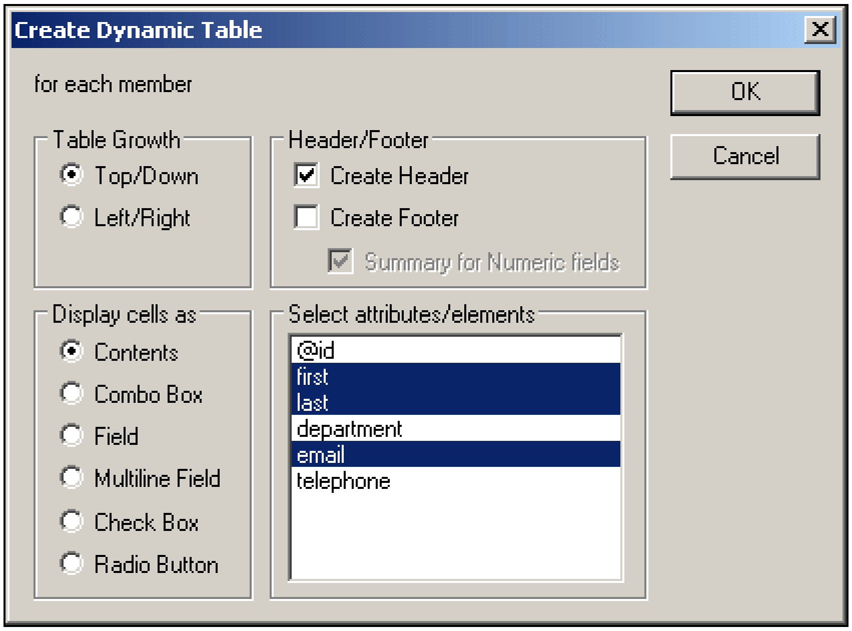

AndrewBentincka.bentinck@nanonull.comNadiaEdwardsn.e

Since the global template for `selection` is empty, the child elements of `selection` are not processed.

Creating a global template for `team/member`

The objective is to create a table to display details of the members of the press monitoring team. This table will be created in a global template for the `team` element. Do this as follows:

1. Create a global template for the element `team` (right-click `team` in the Global Elements list of the Schema Sources entry helper and select **Make Global / Remove Global**).
2. In the Global Elements list, expand the `team` element and drag its `member` child element into the global template of `team` (in the design).
3. In the context menu that pops up when you drop the element into the global template of `team`, select **Create Table**. This pops up the Create Dynamic Table dialog (*screenshot below*).



Create Dynamic Table

for each member

OK | Cancel

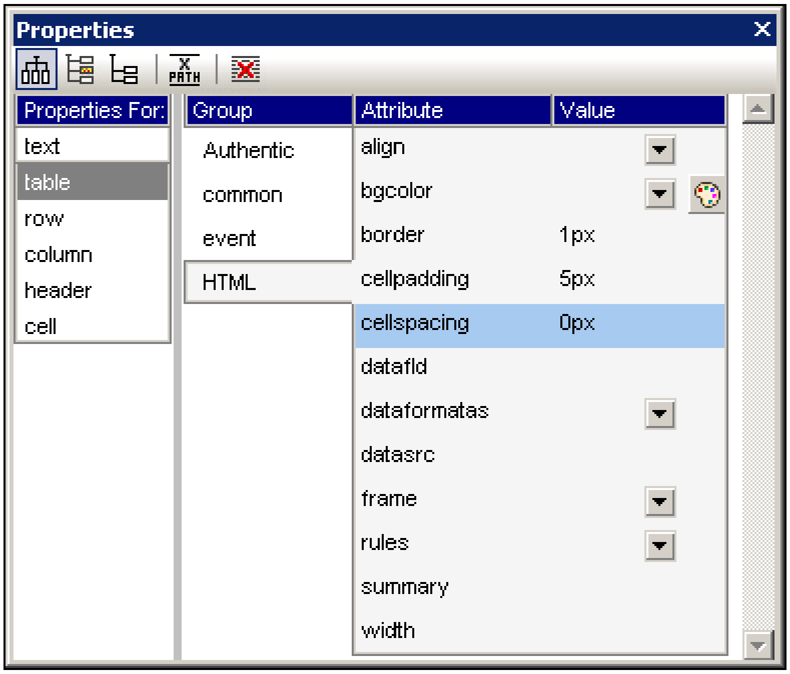
Table Growth
- ● Top/Down
- ○ Left/Right

Header/Footer
- ☑ Create Header
- □ Create Footer
- ☑ Summary for Numeric fields

Display cells as
- ● Contents
- ○ Combo Box
- ○ Field
- ○ Multiline Field
- ○ Check Box
- ○ Radio Button

Select attributes/elements
- @id
- first (selected)
- last (selected)
- department
- email (selected)
- telephone

4. In the attributes/elements list deselect `team`, `department` and `telephone` (*see screenshot*), and click **OK**. The dynamic table is created.
5. Place the cursor in the dynamic table, and in the Properties entry helper, with `table` selected in the Properties For column, specify table properties as shown in the screenshot below.



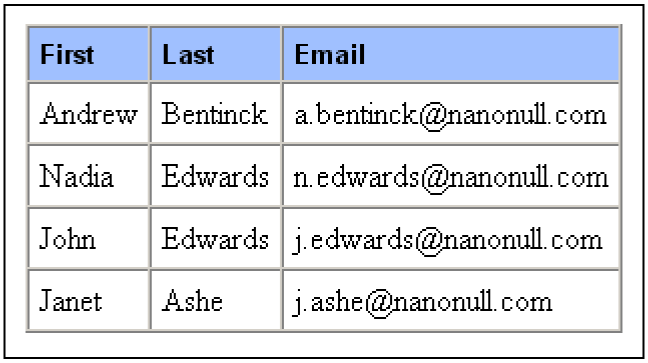
Properties

Properties For:	Group
text	Authentic
table	common
row	event
column	HTML
header	
cell	

Attribute	Value
align	
bgcolor	
border	1px
cellpadding	5px
cellspacing	0px
datafld	
dataformatas	
datasrc	
frame	
rules	
summary	
width	

6. Set additional properties as required in the Properties and Styles entry helpers. For example, a background color can be set for the header row by placing the cursor in the header row, and with `row` selected in the Properties For column of the Styles entry helper, specifying a value for the `background-color` property (*color* group). You can also edit the headers, which are strings of static text. Also, if the `content` placeholder of the `team` element is still present in the global template, delete it.

The HTML preview of the table will look something like this:



First	Last	Email
Andrew	Bentinck	a.bentinck@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com
John	Edwards	j.edwards@nanonull.com
Janet	Ashe	j.ashe@nanonull.com

5.8 That's It!

Congratulations for having successfully completed the tutorial. You have learned the most important aspects of creating an SPS:

- How to create the structure of the document (main template and global templates).
- How to insert dynamic and static content in the design, using a variety of dynamic and static SPS components..
- How to use CSS styles, in external stylesheets, in global style rules, and in local style rules.
- How to use Auto-Calculations to derive additional information from the available XML data.
- How to use conditions to filter the XML data and how to obtain different outputs depending on values in the XML data.
- How to use global templates and rest-of-contents.

For a more detailed description of these features, see the corresponding sections in the following four sections:

- High-Level Procedures
- Content Editing Procedures
- Presentation Procedures
- Additional Editing Procedures

These sections also contain descriptions of several other StyleVision features not encountered in the Quick Start tutorial.

Using the SPS

After completing the SPS, you should also try out the two main uses of SPS:

- Editing XML documents in the Authentic View of XMLSpy or Authentic Desktop. (The Enterprise and Professional editions contain an Authentic View preview tab, which does not have a few features such as Entry Helpers and Text State Icons.) These two products provide a full-feature Authentic View, in which you can try out the entry helpers and context menu. To edit `QuickStart.xml` in Authentic View in XMLSpy or Authentic Desktop, associate the XML file with `MyQuickStart.sps` and switch to Authentic View.
- Generating XSLT stylesheets for transforming the XML file to HTML output. The XSLT stylesheets can be generated using the File | Save Generated Files command or via the command line. Try generating XSLT stylesheets from `MyQuickStart.sps` and then using these stylesheets to transform `QuickStart.xml`.

Chapter 6

High-Level Procedures

6 High-Level Procedures

This section describes important high-level procedures involved when using StyleVision. These procedures are listed below and described in detail in the sub-sections of this section.

- General Usage Procedure provides a broad overview of how to build an SPS. It starts with the sources required by an SPS and goes on to explain how an SPS is used with Authentic View and to generate output.
- Creating the SPS Structure discusses the structure of an SPS and the factors affecting it.
- Working with Databases explains how an SPS can be used to edit databases and generate DB reports.
- Command Line Interface: StyleVisionBatch describes how StyleVision can be used from the command line to generate files.

6.1 General Usage Procedure

Objectives
SPS documents that you create in StyleVision can be used for two broad purposes:

- To control the display of XML source documents in Authentic View and to enable data to be entered in XML documents or DBs via the Authentic View interface.
- To generate XSLT stylesheets for HTML output.

In this way, the SPS can be used to enable XML document editing and to generate HTML output from the edited XML document. Additionally, the generated XSLT stylesheets can be used to transform other XML documents based on the same schema as the SPS.

Steps in creating an SPS
Given below is an outline of the steps involved in creating a new SPS.

1. Assign a schema to the newly created SPS. The schema may be: (i) a schema file (DTD or XML Schema); (ii) an XML Schema generated from a DB; or (iii) a user-defined schema (created directly in StyleVision). This is done in the Schema Sources entry helper.
2. Assign a Working XML File to the SPS. The Working XML File provides the XML data processed by the SPS when generating Authentic View and output previews. The Working XML File is assigned in the Schema Sources entry helper. The Working XML File enables you to preview output in StyleVision.
3. Select the required XSLT version.
4. The SPS document is designed in Design View using the various design components available to the designer. The design process consists of creating a document structure and defining presentation properties.
5. The Authentic View and outputs are tested. If modifications to the design are required, these are made and the SPSdocument is re-tested.
6. If XSLT files or output files are required, these are generated.
7. If required, assign a Template XML File. The Template XML File provides the starting data for a new XML document created with the SPS.
8. The SPS is deployed for use among multiple Authentic View users.

SPS and Sources

Creating a new SPS file

To create a new SPS document, select an option from under the **File | New (Ctrl+N)** command or click the **New from Schema** icon in the Standard toolbar. A new SPS document is created and is displayed in Design View. The new document is given a provisional name of `SPSX.sps`, where `X` is an integer corresponding to the position of that SPS document in the sequence of new documents created since the application was started.

After a new SPS document is created, the source files for the SPS must be assigned.

Assigning source files for the SPS

There are three types of source files that can be assigned to an SPS:

- Schema sources
- Working XML File
- Template XML File

These source file assignments are made in the Schema Sources entry helper, and how to make the assignments is described in the section, Schema Sources. The significant points about each type of source file are given below.

Schema sources

A schema source file must be assigned to an SPS so that a structure for the design document can be created. Schema sources are assigned in the Schema Sources entry helper. A schema may be an XML Schema file (`.xsd` file), an XML Schema generated from a DB file, a DTD, or a user-defined schema. For each schema, one optional Working XML File and one optional Template XML File can be assigned.

Working XML File

An SPS can, optionally, have a Working XML File associated with it. The function of the Working XML File is to provide the XML data source for output previews in StyleVision, and it must therefore be valid according to the schema with which it is associated. The Working XML File is assigned in the Schema Sources entry helper.

Template XML File

An SPS can have a Template XML File optionally associated with it. The function of the Template XML File is to provide the starting data of the new XML document that is created each time that SPS is opened in the Authentic View of a product other than StyleVision. The Template XML File must be valid according to the schema with which it is associated. It is assigned in the Schema Sources entry helper.

XSLT and XPath Versions

An SPS is essentially an XSLT stylesheet. For each SPS you must set the XSLT version: 1.0 or 2.0. You do this by clicking the appropriate toolbar icon: XSLT 1.0 or XSLT 2.0. The selection you make determines two things:

- Which of the two XSLT engines in StyleVision is used for transformations; StyleVision has separate XSLT 1.0 and XSLT 2.0 engines.
- What XSLT functionality (1.0 or 2.0) is displayed in the interface and allowed in the SPS . For example, XSLT 2.0 uses XPath 2.0, which is a much more powerful language than XPath 1.0 (which is used in XSLT 1.0). Additionally, some SPS features, such as the table-of-contents feature, is available only with XSLT 2.0.

XSLT transformations
XSLT transformations in StyleVision are used: (i) to generate output views in the interface; and (ii) to generate and save output files (HTML) from within the interface and from the command line. The XSLT engine used for transformations (Altova XSLT 1.0 Engine or Altova XSLT 2.0 Engine) corresponds to the XSLT version selected in the SPS.

XSLT functionality in GUI
The functionality appropriate for each XSLT version relates mostly to the use of the correct XPath version (XPath 1.0 for XSLT 1.0 and XPath 2.0 for XSLT 2.0). XPath expressions are widely used in StyleVision—most commonly in features such as Auto-Calculations and Conditional Templates—and there are interface mechanisms that require, and help you build, XPath expressions. The functionality of the correct XPath version is automatically made available in the interface according to the XSLT version you select.

Creating the Design

In the SPS design, you specify:

1. What content (from the XML document or DB) should go to the output; additionally content can be inserted directly in the SPS for inclusion in the output;
2. How the output should be structured; and
3. What presentation (formatting) properties are applied to the various parts of the output.

Content for output
The content for the output can come from:

1. The XML document or DB to which the SPS is applied. Content from the XML document is included in the SPS by dragging the required XML data node from the relevant schema tree in the Schema Sources Entry Helper and dropping this node at the desired place in the SPS.
2. An external XML document that is accessible to the application (that is, to StyleVision or an Authentic View product). By using the `doc()` function of XPath 2.0 in an Auto-Calculation, content from external XML document sources can be accessed. An XML document accessed via the `doc()` function in an XPath expression does not need to be referenced via the Schema Sources associations.
3. The SPS itself. Text and other content (such as images and tables) can be inserted directly in the SPS using the keyboard and other GUI features. Such input is independent of the XML document.
4. Manipulated dynamic (XML source) data, with the manipulations being achieved using XPath 1.0 and XPath 2.0 expressions. Manipulations are typically achieved with Auto-Calculations.
5. For the HTML output, JavaScript functions can be used to generate content.

Structure of output
In the SPS design, the structure of the output can be controlled by using either: (i) a procedural approach, in which the output structure is specified in an entry-level template (StyleVision's main template) and can be independent of the structure of the XML document; (ii) a declarative approach, in which template rules are declared for various nodes (StyleVision's global templates), thus generating an output that follows the structure of the XML document; or (iii) a combination of the procedural and declarative approaches. In Design View, you can use a mix of main template and global templates to obtain the desired structure for the output document.

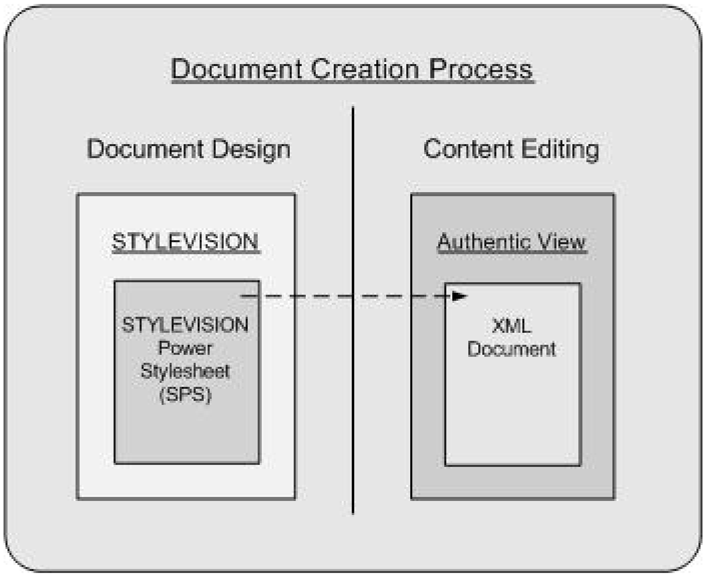
Presentation (or formatting) of the output
In Design View, presentation properties are applied to design components using CSS styles. Styles can be defined locally on the component, for HTML selectors declared at the document level, and for HTML selectors declared in an external CSS stylesheet. Additionally, certain HTML elements can be applied to components using predefined formats. Specifying presentation properties is described in detail in the section, Presentation Procedures.

SPS and Authentic View

One of the core uses of the SPS you create with StyleVision is to control the input of data and the display of an XML document in Authentic View, which is a document view available in Altova products. With Authentic View, users who are unfamiliar with XML can easily enter and edit XML document content correctly.

A document creation and editing process that involves Authentic View consists of two separate stages:

- ***Document design.*** The Authentic View of the XML document, which is graphical view, is designed in StyleVision. The design document is an SPS. The SPS not only processes the XML document for display in Authentic View and for final output; it also provides mechanisms, in Authentic View, for **inputting data into the XML file or DB.**



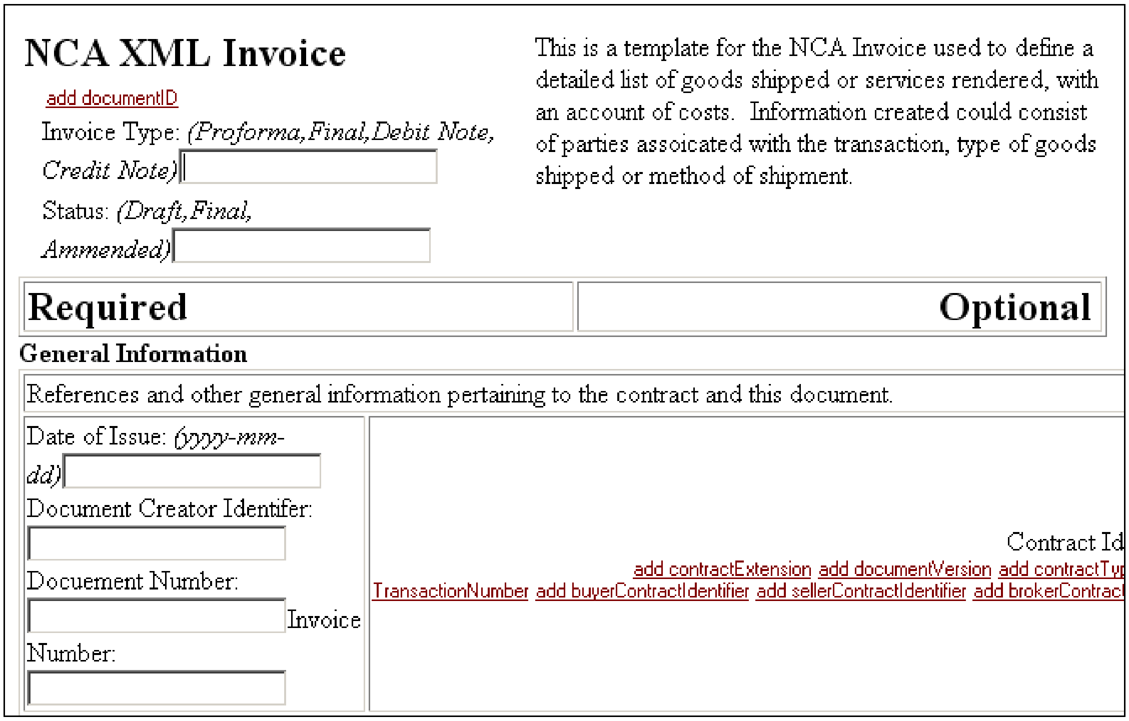
- ***Content editing.*** This SPS created in the document design stage is linked to the XML document to be edited. (The XML document must be valid according to the schema on which the SPS is based.) An XML document which is linked to an SPS is presented graphically in the Authentic View of an Altova product as the Authentic View of that XML document. When a new Authentic XML document is created, it can be assigned an SPS and then be edited in Authentic View using the document template (Template XML File) and controls specified in the SPS. If an existing XML document is opened and assigned an SPS, the existing data is displayed in Authentic View according to the design in the SPS, and the document can be edited in Authentic View.

The user of Authentic View is not expected to be knowledgeable about either XML or the schema being used for the document. The document display in Authentic View should make content editing as easy and non-technical as possible. It is, therefore, the task of the person who designs the SPS to produce a user-friendly Authentic View display. For detailed information about using Authentic View, see the Authentic View documentation in the user manual of XMLSpy or Authentic Desktop.

SPSs for standard industry schemas

Altova's Authentic View package includes SPSs for a number of standard industry schemas. Users can therefore immediately create an XML document based on a standard schema in

Authentic View. The screenshot below shows a partial Authentic View of the NCA Invoice standard.



NCA XML Invoice

add documentID

Invoice Type: *(Proforma,Final,Debit Note, Credit Note)*

Status: *(Draft,Final, Ammended)*

This is a template for the NCA Invoice used to define a detailed list of goods shipped or services rendered, with an account of costs. Information created could consist of parties assoicated with the transaction, type of goods shipped or method of shipment.

Required	Optional

General Information

References and other general information pertaining to the contract and this document.

Date of Issue: *(yyyy-mm-dd)*

Document Creator Identifer:

Docuement Number:

Invoice Number:

Contract Id

add contractExtension add documentVersion add contractTy

TransactionNumber add buyerContractIdentifier add sellerContractIdentifier add brokerContrac

You can easily customize any of the supplied standard industry SPSs, which are available in the **`Examples/IndustryStandards`** folder of your application folder.

Generated Files

In StyleVision, XSLT stylesheets and output files can be generated using the **File | Save Generated Files** command or the command line utility, `StyleVisionBatch.exe`. The following files can be generated:

- XSLT stylesheets based on the SPS design.
- Output files generated by processing the Working XML File assigned in the SPS with the XSLT stylesheets generated from the SPS. The command line utility offers the option of specifying XML files other than the Working XML File as the XML input.

The markup for the output is contained in the SPS. The data for the output is contained in the XML document or DB. It is the XSLT stylesheet that brings markup and data together in the output. Both the XSLT stylesheets as well as the actual output can be previewed in StyleVision in the Output Views.

Given below are important points to note about the generated documents:

- ***HTML output and stylesheets:*** (1) The formatting and layout of the generated HTML document will be identical to the HTML Preview of StyleVision and near-identical to the Authentic View of the XML document. (2) Data-input devices (text input fields, check boxes, etc) in the HTML file do not allow input. These data-input devices are intended for XML data input in Authentic View and, though they are translated unchanged into the graphical HTML equivalents, they cannot be used for data-entry in the HTML document.

6.2 Creating the SPS Structure

The structure of the SPS is defined in its main template. The main template consists of components which may be dynamic (content taken from the XML document) or static (content inserted directly in the SPS). A dynamic component is most commonly the text content of a node or nodes, but may also have a structure in the SPS, as when an element from the XML document is inserted as a dynamic table or a dynamic list. Static components are usually single items, such as a text string or a horizontal line, but some components, such as static tables and static lists, may have a substructure.

The main factors that influence the structure of the SPS are:

- The schema sources and what schema trees they make available for use in the design.
- The use of, and interaction between, main template and global templates.

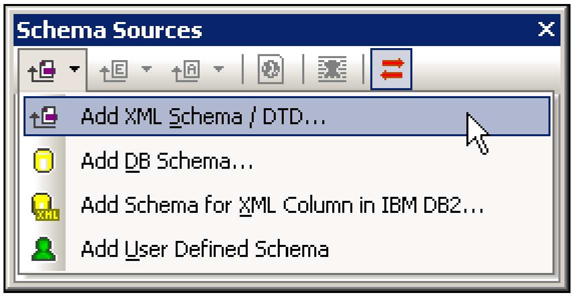
These factors are discussed in the sub-sections of this section.

Schema Sources

The schema sources are the starting point of the design, and design structure can be influenced by: (i) choices you make during schema selection, and (ii) the root elements you select in the schema.

Schema selection

The selection of the schema is done in the Schema Sources entry helper using the **Add Schema** button (*screenshot below*).

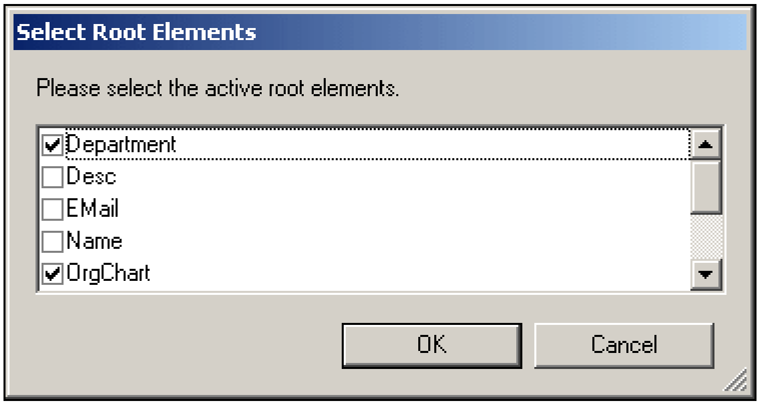


The schema source can be selected from a file, from a DB, or be user-defined. An important point to consider is whether you will be using global templates, and whether elements you wish to create as global templates are defined as global elements in the schema. When adding a DTD from file, remember that all elements defined in the DTD are global elements. When adding an XML Schema from file, it is worth checking what elements are defined as global elements and, should you wish to make any change to the schema, whether this is permitted in your XML environment. When a DB is selected, during the import process, you can select what tables from the DB to import. This selection determines the structure of the XML Schema that will be generated from the DB.

Root elements

If a schema source has multiple global elements, then multiple root elements (document elements) can be selected for use in the design. This enables the SPS design to have templates that match multiple document elements. The advantage of this is that if an SPS, say `UniversalSPS.sps`, based on `UniversalSchema.xsd` has one template each for its two root elements, `Element-A` and `Element-B`, then this one SPS can be used with an XML instance document which has `Element-A` as its document element as well as with another XML instance document which has `Element-B` as its document element. For each XML instance, the relevant template is used, while the other is not used. This is because for the document element of each XML instance document, there is only one template in the SPS which matches that document element. For example, the document element `/Element-A` will be matched by the template which selects `/Element-A` but not by that which selects `/Element-B`. In this connection, it is important to remember that if multiple global elements are defined in the schema, an XML document with any one of these global elements as its document element is valid (assuming of course that its substructure is valid according to the schema).

To set up the SPS to use multiple root elements (document elements), click the ... button to the right of the `/Root elements` entry of the schema. The following dialog pops up.

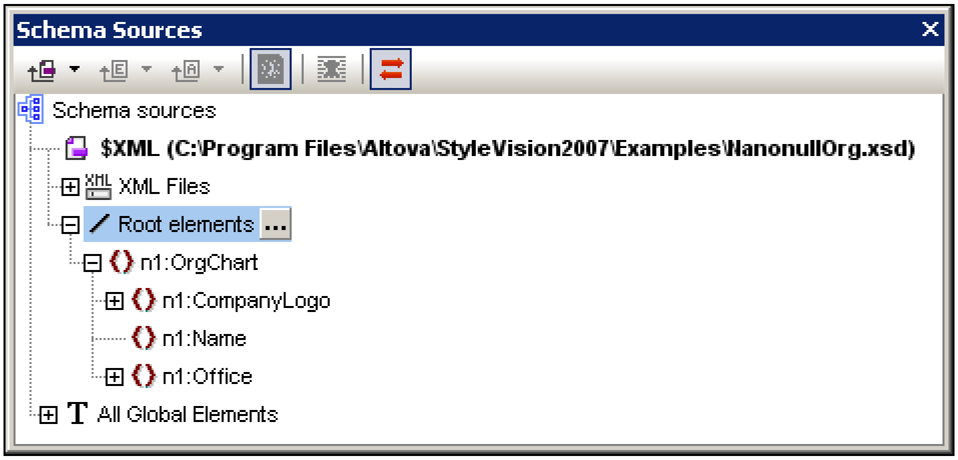


Select Root Elements

Please select the active root elements.

- ☑ Department
- □ Desc
- □ EMail
- □ Name
- ☑ OrgChart

The dialog lists all the global elements in the schema. Select the global elements that should be available as root elements (document elements), and click **OK**. The schema tree would then look something like this:



Schema Sources

```
Schema sources
  $XML (C:\Program Files\Altova\StyleVision2007\Examples\NanonullOrg.xsd)
    XML Files
    Root elements ...
      n1:OrgChart
        n1:CompanyLogo
        n1:Name
        n1:Office
  All Global Elements
```

For the SPS represented in the screenshot above, three templates, to match each of the three document elements, can now be created in the design. The SPS can then be used with different XML instances that are valid according to `NanonullOrg.xsd`, some with `Department` as document element, others with `OrgChart` as document element, and still others with `Person` as document element. The appropriate template will be used in the case of each instance document.

DTDs and XML Schemas

An SPS can be based on an XML Schema or DTD. An XML Schema or DTD can be created as a schema source in one of the following ways:

- The XML Schema or DTD is is created as a schema source directly when the SPS is created (**File | New | New from XML Schema / DTD**).
- The XML Schema or DTD is added to an empty SPS (in Schema Sources entry helper (toolbar icon, or context menu when a Schema Sources item is selected), **Add Schema | Add XML Schema / DTD**).

The respective commands prompt you to browse for the XML Schema or DTD. If the schema is valid, it is created as a schema source in the Schema Sources tree of the Schema Sources entry helper.

DB Schemas

An SPS can be based on a schema that is generated from a DB. A DB schema can be created as a schema source in one of the following ways:

- The DB schema is generated for a new SPS when the SPS is created directly from a DB (**File | New | New from DB**).
- The DB schema is added to a new empty SPS that has no schema sources (in Schema Sources entry helper (toolbar icon or context menu), **Add Schema | Add DB Schema**).

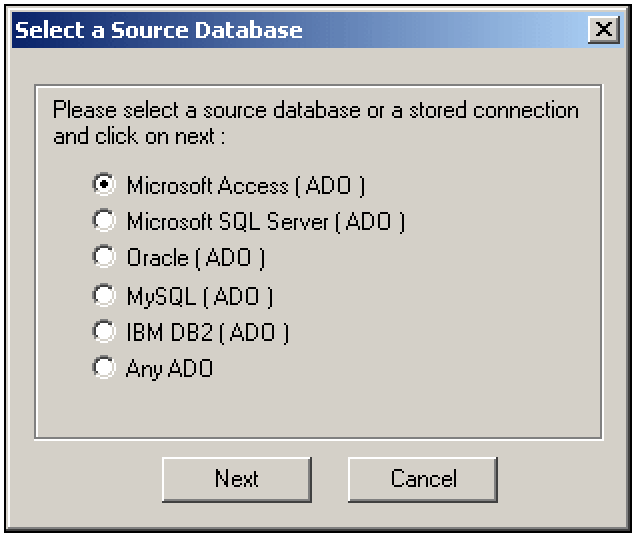
The respective DB-schema-generation commands generate a temporary XML Schema from the selected DB and creates the schema as a schema source in the Schema Sources entry helper. An element extraneous to the DB, called `DB`, is created as the document element, and the DB structure is created within this document element. During the schema creation process, you will be prompted to select which tables from the database you wish to import. These database tables will be created in the XML Schema as children of the `DB` element and also as items in the Global Templates list.

Creating the XML Schema from the DB consists of two steps:

- Browsing for the DB file (if it is a MS Access DB), or building a **connection string** (all other DBs except MS Access). Note that **only ADO connections are supported**. This is because ADO's shape command support enables data to be generated hierarchically.
- Selecting the **Base Table information** (which enables you to select what tables from the database to import). The temporary XML Schema that is generated and the XML file that is created will be based on the selected data tables.

Given below are the steps for creating the schema:

1. Click either of the DB-schema-generation commands (create new SPS from DB or add DB schema; *see above*). The following dialog is displayed:

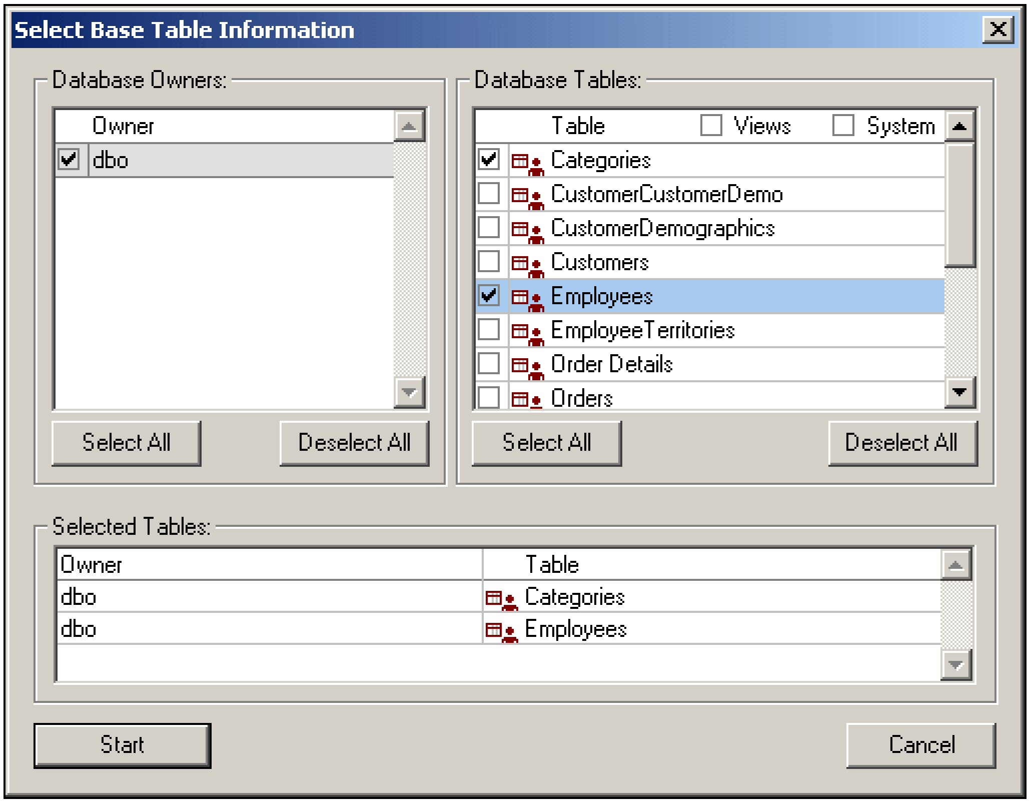


Select a Source Database

Please select a source database or a stored connection and click on next :

● Microsoft Access (ADO)
○ Microsoft SQL Server (ADO)
○ Oracle (ADO)
○ MySQL (ADO)
○ IBM DB2 (ADO)
○ Any ADO

2. Select the source DB type and click **Next**. If you selected a DB type of MS Access, then a dialog appears in which you can browse for the DB. If you selected any other DB type option, a dialog appears prompting you to build a connection string (see Connect to a DB and set up the SPS for details about building a connection string). After you have selected the MS Access DB or built a connection string to the DB, click **Next**.
3. The next dialog is the Select Base Table Information dialog (shown below), which lists the data tables in the selected DB by owner.



	Owner
☑	dbo

	Table □ Views □ System
☑	Categories
□	CustomerCustomerDemo
□	CustomerDemographics
□	Customers
☑	Employees
□	EmployeeTerritories
□	Order Details
□	Orders

Owner	Table
dbo	Categories
dbo	Employees

4. Select the database owner, if any. This causes a list of data table belonging to that owner to be displayed in the Database Tables pane.
5. In the Database Tables pane, select the DB tables that you wish to include in the schema. These tables are displayed in the Selected Tables pane.
6. Click **Start**. An XML Schema is created, with the selected tables as children of the document element DB. The schema is displayed in the Schema Sources entry helper. The Global Templates list also contains the selected DB tables.

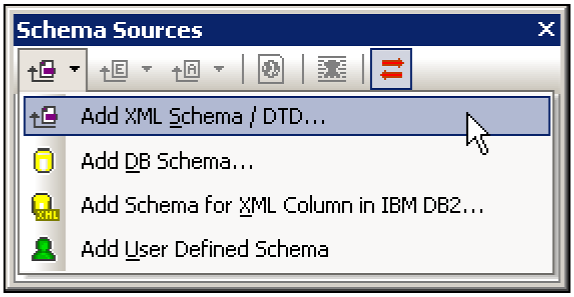
You can now start designing the SPS. A temporary XML File is created each time output preview tab is clicked. The XML file is structured according to the generated XML Schema and contains data from the selected DB tables.

User-Defined Schemas

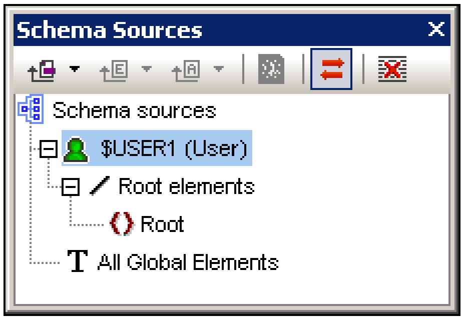
You can quickly create a user-defined schema in the Schema Sources entry helper. This is useful if you have an XML document that is not based on any schema and you wish to create an SPS for this XML document.

To add and create a user-defined schema, in the Schema Sources entry helper, do the following:

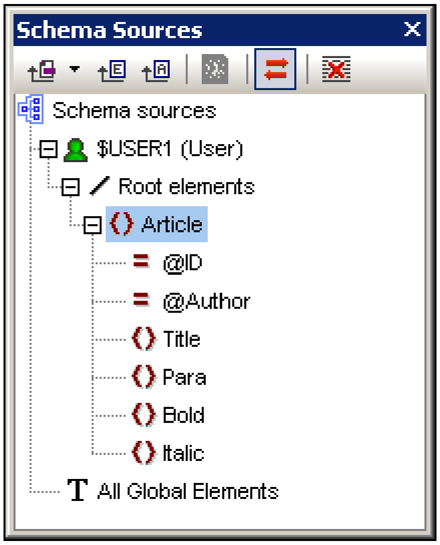
1. Click **File | New | New (empty)**. In the Schema Sources entry helper, click the dropdown arrow of the Add Schema icon and select **Add User-Defined Schema** (*screenshot below*).



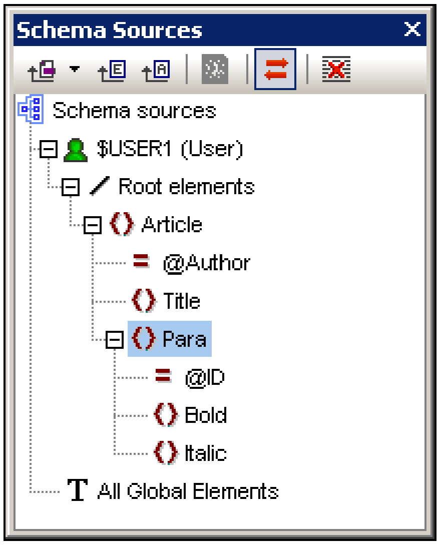
The new schema is created and is indicated with the parameter `$USER1` (*screenshot below*).



2. Now expand the Root Elements item. There is a single root element (document element) called `Root`.
3. Double-click `Root` and rename it to match the document element of the XML document for which you are building this schema.
4. To assign a child element or an attribute to the document element, click, respectively, the Append New Element icon or the Append New Attribute icon in the toolbar of the Schema Sources entry helper. Alternatively, you can right-click and select the required command from the context menu. After the new element or attribute is added to the tree, type in the desired name. Note that the Append New Element icon and append New Attribute icon have dropdown menus that can be accessed by clicking the dropdown arrow in the respective icon. The dropdown menus contain items that enable you to add nodes at alternative levels relative to the selected node. You can also drag nodes to the desired location (described in the next step). In the screenshot below, the `Article` element is the document element. The elements `Title`, `Para`, `Bold`, and `Italic`, and the attributes `ID` and `Author` have been added at the child level of `Article`.



5. To move the elements `Bold` and `Italic`, and the attribute `ID` to the level of children of `Para`, select each individually and drag to the `Para` element. When a bent downward-pointing arrow ↴ appears, drop the dragged node. It will be created as a "child" of `Para` (*screenshot below*).



6. When any element other than the document element is selected, adding a new element or attribute adds the new node at the same level as the selected element. Drag a node (element or attribute) into an element node to create it as a "child" of the element node.

Editing node names and deleting nodes
To edit the name of an element or attribute, double-click in the name and edit the name. To delete a node, select it and click the Remove icon ☒ in the toolbar. Alternatively, select **Remove** from the context menu.

Output Structure

The design document consists of one main template and, optionally, one or more global templates. In this section, we describe the role that these two kinds of templates play in the structure of the design. We are not concerned here with the presentation properties in the design, only the structure.

This section describes the three kinds of templates that are used in an SPS:

- Main Template
- Global Templates
- Design Fragments

Main Template

The main template determines the structure of the output. That is, the sequence in which the main template is laid out in the design is the sequence in which Authentic View and the output is laid out. In programming jargon, this is procedural processing. Processing starts at the beginning of the template and proceeds in sequence to the end. Along the way, nodes from the XML document are processed. The templates which process these nodes are called local templates. After a local template is processed, the processor moves to the next component in the main template, and so on. Occasionally, a node may reference a global template for its processing. In such cases, after the global template is executed for that node, the processor returns to the position in the main template from which it branched out and continues in sequence from the next component onwards.

The entry point for the main template is the document node of the schema. StyleVision offers the option of selecting multiple root elements (document elements). This means that within the main template, there can be local templates for each of the active document elements. The one that is executed during processing will be that for the element which is the document element of the XML instance document being processed.

Global Templates

A global template is defined for a global element. It specifies instructions for the selected schema element and is invoked by a call from the main template. The processing model is similar to that of declarative programming languages, in that a single template is defined and invoked multiple times. Global templates are invoked in two situations:

- When an element in the main template has been set to reference its global template (done by right-clicking and selecting Make Global Template).
- When a `(rest-of-contents)` is inserted within an element in a local template, and the rest of the contents of that element includes an element for which a global template exists.

Global templates are useful if an element occurs within various elements or in various locations, and a single set of instructions is required for all occurrences. For example, assume that a `para` element must be formatted the same no matter whether it occurs in a `chapter`, `section`, `appendix`, or `blockquote` element. An effective approach would be to define a global template for `para` and then ensure, that in the main template the global template for the `para` element is processed wherever required (for example, by including `//chapter/para` in the main template and specifying that `para` reference its global template; or by including `//chapter/title` and then including `(rest-of-contents)`).

Note:

- Global templates are of two types: simple and complex. Complex global templates are available for reasons of backward-compatibility. If a global template in an SPS created with a version of StyleVision prior to version 2006 contains a table or list, then that global template will typically be opened in StyleVision 2006 and later versions as a complex global template.
- A complex global template is different than a simple global template in the way the node for which the global template was created is processed. When the first instance of the node is encountered in the document, the complex global template processes all subsequent instances of that node immediately afterwards. A simple global template, on the other hand, processes each node instance only when that node instance is individually encountered.
- It is important to note that a simple global template will be automatically converted to a complex global template if a predefined format or newline is created **around** the element node for which the global template was created. This will result in the processing behaviour for complex global templates (described in the previous list item). To revert to the simple global template, the predefined format should be removed (by dragging the node outside the predefined format and then deleting the predefined format), or the newline should be removed (by deleting the item in the Design Tree entry helper), as the case may be. To avoid the automatic conversion from simple global template to complex global template, make sure that the predefined format or newline is added within the node tags of the element for which the simple global template was created.

Design Fragments

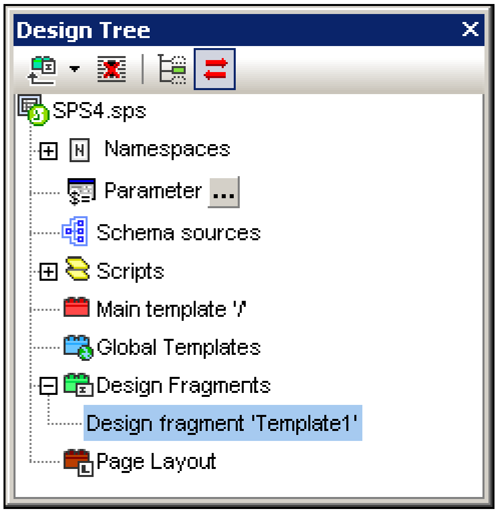
Design Fragments are useful for creating parts that can be re-used at different locations in the document, similar to the way functions are re-used.The usage mechanism is as follows:

1. Create the Design Fragment in the design
2. Fill out the contents of the Design Fragment
3. Insert the Design Fragment at a location in a template.

Creating a Design Fragment

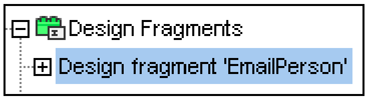
To create a Design Fragment do the following:

1. In the Design Tree, click the Add Design Fragment toolbar icon . This adds a Design Fragment item in the Design Fragments list of the design tree (*screenshot below*).



Notice that a Design Fragment template is also created in the design tree. This template is appended to the templates already in the design. (If you wish to see only the Design Fragments that are in the design, hide the main template and global templates by clicking their Show/Hide icons in StyleVision's Template Filter toolbar.)

2. Double-click the Design Fragment item so as to edit its name. Name the Design Fragment as required and press **Enter**. The edited name is entered in the Design Tree (*screenshot below*) and in the template in the design.

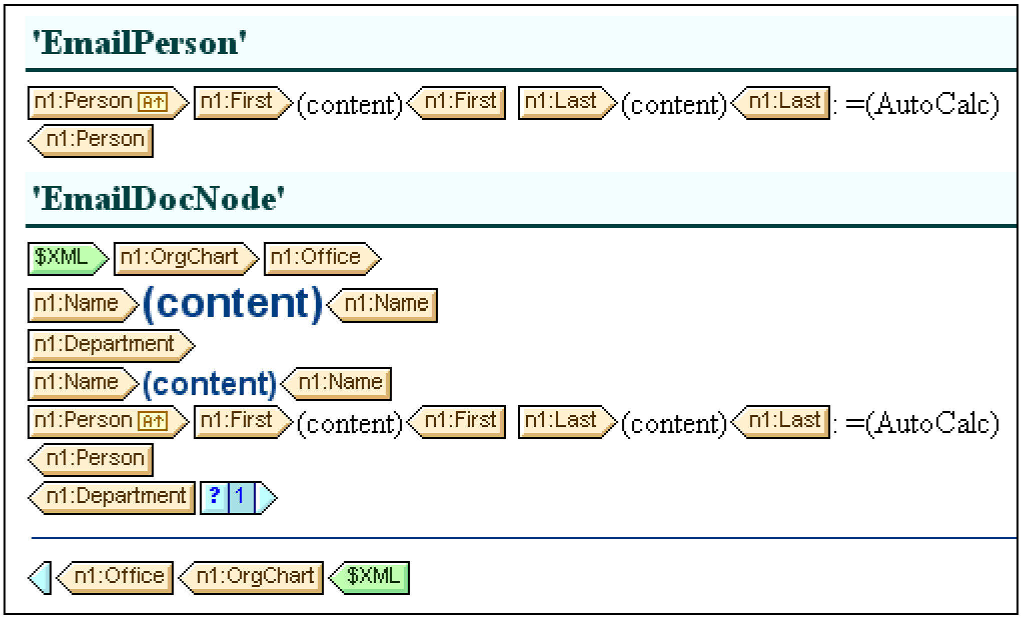


3. In the design, create the contents of the Design Fragment template. How to do this is described in the next section.

Creating the contents of a Design Fragment

The contents of the Design Fragment template are created as for any other template. To insert static content, place the cursor in the Document Fragment template and insert the required static content. To insert dynamic content, drag the required schema node into the Design Fragment template.

When dragging a node from the schema source you can drag the node either: (i) from the Global Elements tree, or (ii) from the Root Elements tree. The difference is significant. If a node is dragged from the Global Elements tree, it is created without its ancestor elements (in the screenshot below, see the `EmailPerson` Design Fragment) and, therefore, when used in a template, it will have to be used within the context of its parent. On the other hand, if a node is dragged from the Root Elements tree, it is created within a structure starting from the document node (in the screenshot below, see the `EmailDocNode` Design Fragment), and can therefore be used anywhere in a template.



The screenshot above shows two Design Fragment templates that produce identical output. In the `EmailPerson` Design Fragment template, the `Person` node has been created by dragging the global element `Person` into the `EmailPerson` template. In the `EmailDocNode` Design Fragment template, the `Person` node has been dragged from the Root Elements tree, and is created with an absolute path (from `$XML`, the document node).

When these Design Fragment templates are inserted in the main template, care must be taken that the `EmailPerson` template is called from within a context that is the parent of the `Person` node. You can experiment with these Design Fragments. They are in the example file `Email.sps`, which is in the folder `StyleVision2007/Examples/Tutorials/DesignFragments`.

After you have completed the design, notice that the components of the design are also graphically depicted in the Design Tree.

Inserting a Design Fragment in a template

To insert a Design Fragment, drag the Design Fragment from the Design Tree to the required location. The location at which the Design Fragment is dropped should provide a correct context. If the contents of the Design Fragment were created from a global element, then the correct context in the main template would be the parent of the node dragged into the Design Fragment. See Creating the contents of a Design Fragment above.

Note: If a Design Fragment is referenced in the main template and if the name of the Design Fragment is changed subsequently, then the reference in the main template will no longer be correct and an XSLT error will result. In order to correct this, delete the original reference in the main template and create a fresh reference to the newly named Design Fragment.

Alternatively, right-click at the location where the Design Fragment is to be inserted and select **Insert Design Fragment** from the context menu.

Deleting a Design Fragment
To delete a Design Fragment, select it in the Design Tree and click the Remove toolbar icon of the Design Tree ☒.

Example file
For an example SPS, go to the folder `StyleVision2007/Examples/Tutorials/DesignFragments`.

6.3 Working with Databases

When a DB is used as the basis of an SPS—that is, as the main schema of an SPS—the SPS can be used in the following ways:

- To edit the DB in Authentic View.
- To generate an XML Schema having a structure based on the DB (if the DB does not contain a schema; only XML DBs, such as IBM DB2 version 9 upwards, contain schemas).
- To generate an XML file with data from the DB (if the required DB data is not already in XML format).
- To design and generate XSLT stylesheets for HTML output.
- To generate DB reports (based on the SPS design) in HTML format. These reports can be previewed in StyleVision

When a DB is the source of a subsidiary schema in an SPS, then data from the DB can be included in the design document, but the DB itself cannot be edited in Authentic View. It is the XML document or DB associated with the main schema that can be edited.

General procedure

This section describes the procedure for working with DBs in StyleVision. After an introductory sub-section, which provides an overview of how DBs work in StyleVision, the sub-sections of this section describe the various steps in the work procedure. Note that we distinguish between two broad types of DBs: non-XML DBs and XML DBs. The term DB is used in two senses: generically, it refers to all DBs; specifically, to non-XML DBs. XML DBs are always referred to as XML DBs. The distinction should be borne in mind because the connection procedure is different for these two types of DB.

- Connecting to a DB and Setting up the SPS: Describes how to connect to non-XML DBs, including IBM DB2 versions below 9..
- Connecting to an XML DB and Setting Up the SPS: Describes how to connect to XML DBs—currently IBM DB2 version 9 and above.
- The DB Schema and DB XML file: When DB tables (from non-XML DBs) are loaded, StyleVision generates and works with temporary XML Schema and XML data files based, respectively, on the DB structure and data. For XML DBs, the schema and XML files are not generated by StyleVision but referenced directly from the DB or, in the case of schemas, from another file location.
- DB Filters: Filtering DB Data: DB data that is loaded into the temporary XML file can be filtered.
- SPS Design Features for DB: In the SPS, special DB functionality, such as DB controls and DB Queries, are available.
- Generating Output Files: A wide range of DB report-related files can be generated by StyleVision.

DBs and StyleVision

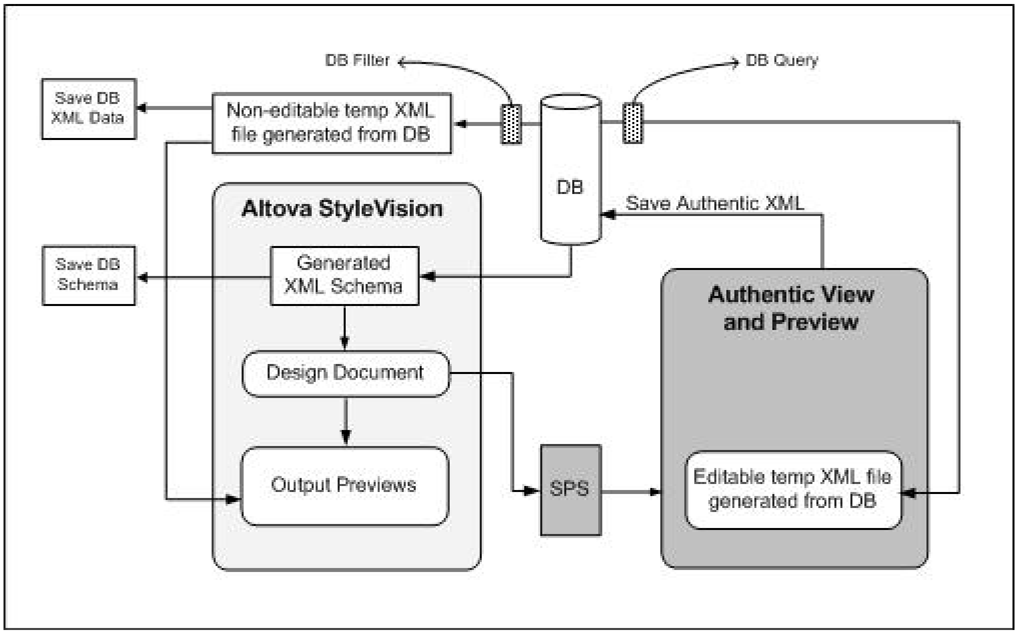
In StyleVision, you can create DB-based SPSs. These stylesheets enable you to do two things:

- Edit DBs in Authentic View, and
- Generate reports from DBs.

After you have created the SPS, you can view reports in StyleVision and generate report files in HTML format. You can also save the following DB-related XML files that StyleVision generates:

- XML Schema based on DB structure (not applicable for XML DBs, where a schema is already available)
- XML file having structure defined in the generated schema and content from the DB (not applicable for XML DBs, where the data is already available in XML format)
- SPS that you design, and which is based on the generated schema
- XSLT stylesheet for HTML output (based on design of SPS)
- HTML output

The saved XML file can then be processed with the required XSLT stylesheet/s. This provides more flexible report-generating capabilities.



Note: The XML Schema and XML files are generated from non-XML DBs by StyleVision, and you cannot alter their structure or content for use in Authentic View. This is because the structure of these files is related to the structure of the non-XML DB. Editing the DB and creating reports from the DB depend on the unique XML structure generated by StyleVision from the DB.

Broad mechanism for working with DB-based SPSs
Given below are the steps involved in creating and using DB-based SPSs. These steps cover the two uses of DB-based StyleVision Power Stylesheets: editing the DB and creating HTML reports from the DB.

- Connect to the database with StyleVision. During the connection process you can specify what data tables in the DB should be filtered out from the XML Schema..
- When the connection is made, a temporary XML Schema is generated based on the structure of the DB and that schema is displayed in the Schema Window of StyleVision in tree form. In the case of XML DBs, a pre-existing schema (either in the DB or at a file location) is referenced.
- Temporary StyleVision-internal XML files are also created. One is non-editable (*see screenshot above*) and is used for the previews and as the source of the generated XML data file. The other is an editable XML file, which is displayed in Authentic View (*see screenshot above*). When changes made to this file in Authentic View are saved (with the **File | Save Authentic XML Data** command), the modifications are written back to the DB. The non-editable XML file is updated if necessary each time an output view is newly accessed or when the XML data is saved.
- In StyleVision, you can define top-level filters to restrict the data imported into the non-editable XML File, i.e. for the output views and the reports.
- A DB Query is used within Authentic View to restrict the list of records displayed in Authentic View. It is used only during editing.
- If editing changes have been saved to the DB, then the next time an output view window is accessed, the non-editable XML file is updated with the modified contents of the DB and the refreshed file is displayed in the preview.
- A DB-based SPS is created in the same way as the standard schema-based SPS: by dragging-and-dropping nodes into the Design Window, inserting static stylesheet components, assigning display properties, etc. These mechanisms are described in this documentation.

Connecting to a DB and Setting Up the SPS

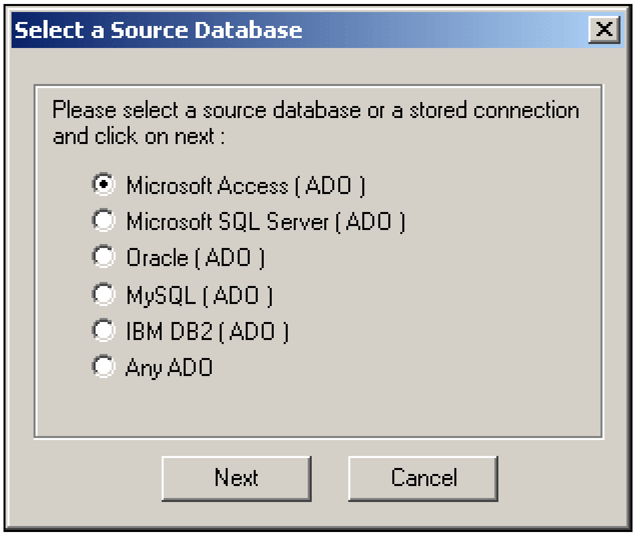
The procedure for setting up an SPS based on and connected to a DB is as follows:

1. Connect to the DB. Note that **only ADO connections are supported**. This is because ADO's shape command support enables data to be generated hierarchically. For an MS Access DB, the connection is made when the DB is opened. For all other DBs—that is, except Microsoft Access (MS Access) DBs—a connection string must be built explicitly. Note also that connections to IBM DB2 version 9 DBs and upwards are made using the command **File | New | New from DB XML**—not **File | New | New from DB**.
2. Select the tables in the DB that are to be imported. Based on these tables, an XML Schema is generated for the SPS.

These steps are described in detail below.

Selecting the DB type

On clicking **File | New | New from DB** or the **Add DB Schema** command in the Schema Sources entry helper, the Select A Source Database dialog (*screenshot below*) appears.



Select a Source Database

Please select a source database or a stored connection and click on next :

● Microsoft Access (ADO)
○ Microsoft SQL Server (ADO)
○ Oracle (ADO)
○ MySQL (ADO)
○ IBM DB2 (ADO)
○ Any ADO

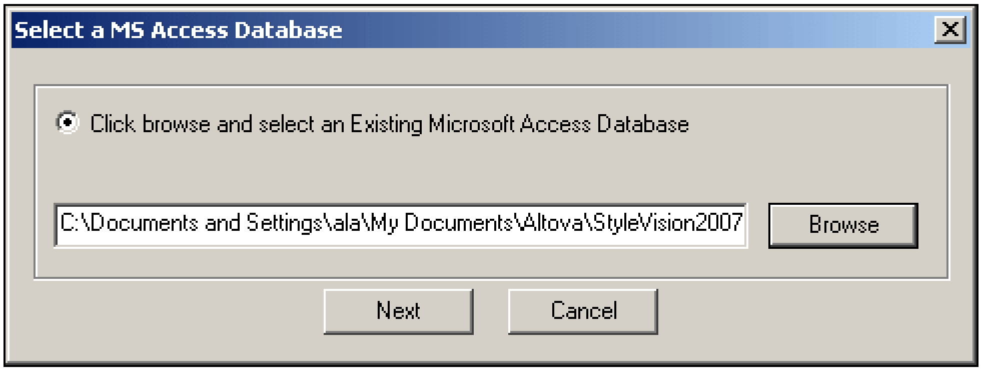
Select the required source DB type and click **Next**.

If you selected MS Access, the Select a MS Access Database dialog appears that prompts for the location of the MS Access DB. If you selected any other DB type, the ADO Connection String dialog appears; this dialog is the starting point for a series of dialogs that help you build a connection string for that DB.

Note: For **IBM DB2 databases from version 9 onwards** (which have support for XML columns), there is a separate connection mechanism. For connecting to and building an SPS based on such a DB, use StyleVision's **File | New | New from DB XML** command or the **Add DB XML Schema** command in the Schema Sources entry helper.

Connecting to an MS Access DB

To connect to an MS Access DB, browse for the required MS Access DB (*see screenshot below*). Note that StyleVision supports MS Access 2000 and MS Access 2003 DBs.



● Click browse and select an Existing Microsoft Access Database

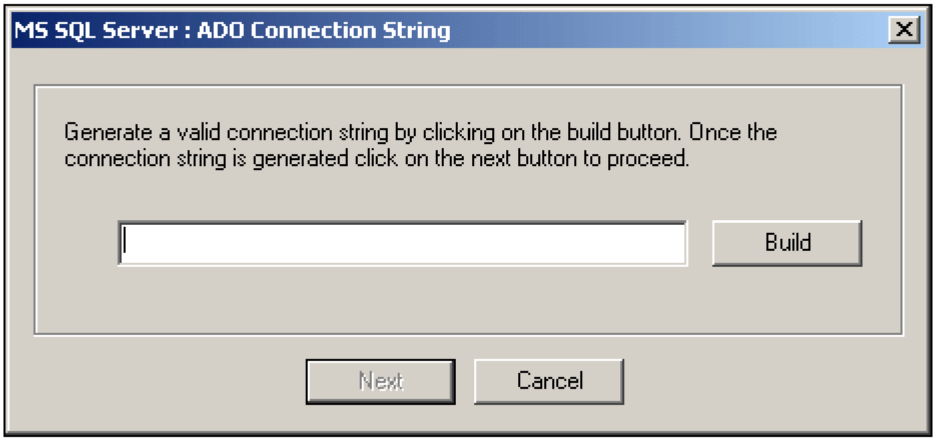
C:\Documents and Settings\ala\My Documents\Altova\StyleVision2007

When you are done, click **Next**. The Select Base Table Information dialog will be displayed, in which you select the DB tables from which the XML Schema structure will be generated; how to do this is described in the section Select Base Table Information below.

Note: If the SPS you are creating will be used by several users to access or edit a single DB, then the path to the DB should be a UNC path. This ensures that the SPS is correctly set up to point to the MS Access database via a network share. Without a UNC path in the SPS and a network share mechanism, it will not be possible to point clients correctly to the DB. For an overview of network shares and UNC paths, see Network shares and UNC paths below.

Connecting to a DB via ADO

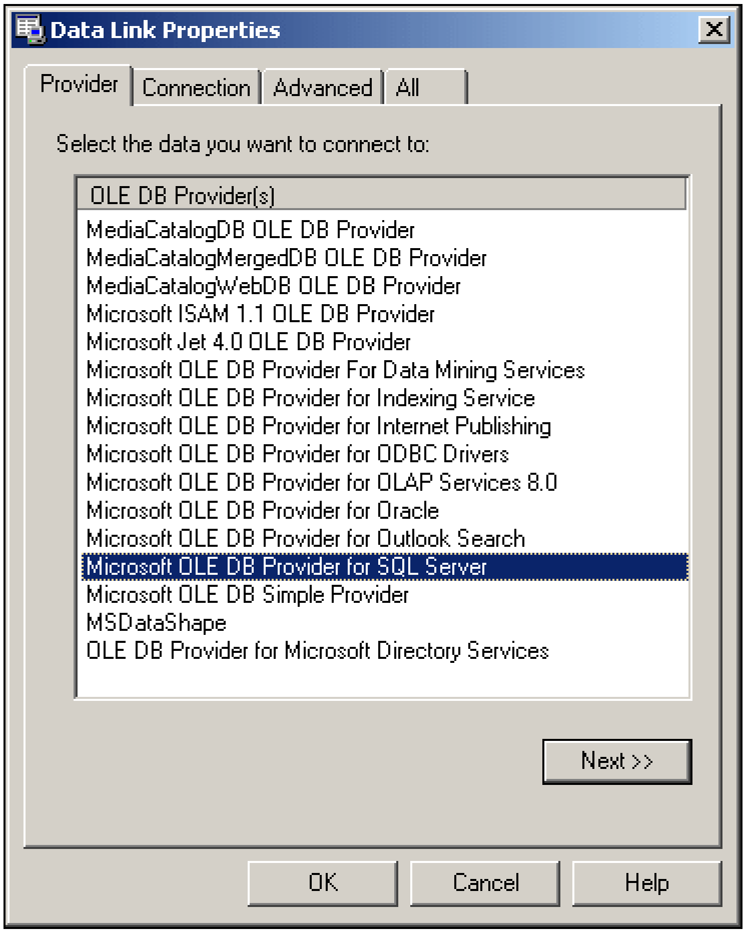
To connect to any DB other an MS Access DB, you must explicitly build a connection string to that DB. The starting point for building the connection string is the ADO Connection String dialog (*screenshot below*), which appears after you click the **Next** button in the Select a Source Database dialog.



Generate a valid connection string by clicking on the build button. Once the connection string is generated click on the next button to proceed.

Build the ADO connection string as follows:

1. In the ADO Connection String dialog, click **Build**. This pops up the Data Link Properties dialog.

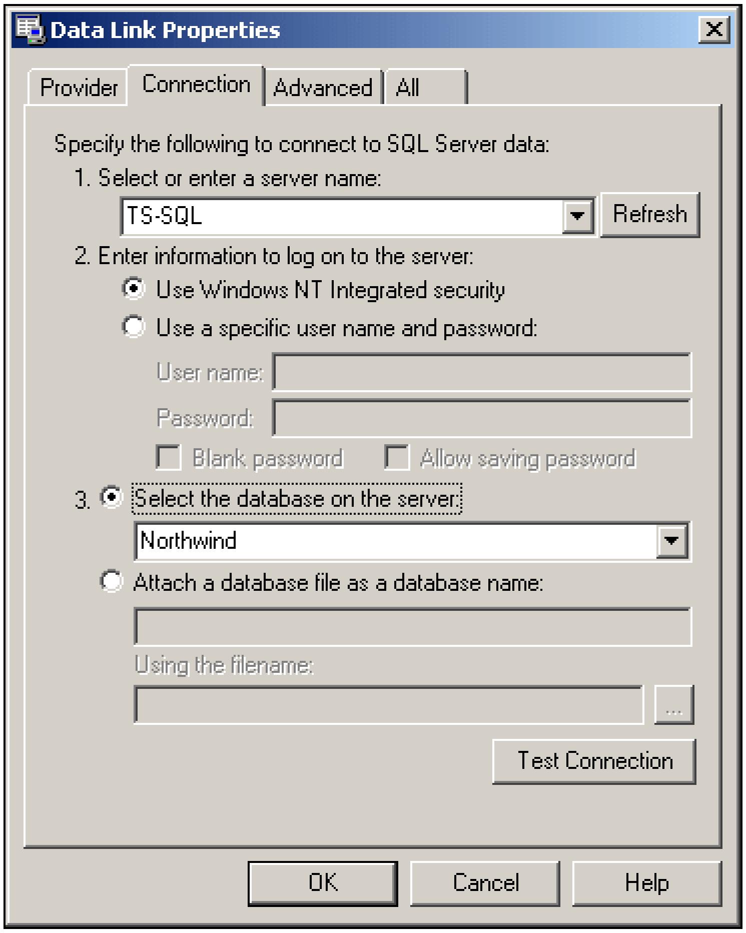


Select the data you want to connect to:

OLE DB Provider(s)

MediaCatalogDB OLE DB Provider
MediaCatalogMergedDB OLE DB Provider
MediaCatalogWebDB OLE DB Provider
Microsoft ISAM 1.1 OLE DB Provider
Microsoft Jet 4.0 OLE DB Provider
Microsoft OLE DB Provider For Data Mining Services
Microsoft OLE DB Provider for Indexing Service
Microsoft OLE DB Provider for Internet Publishing
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services 8.0
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for Outlook Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Services

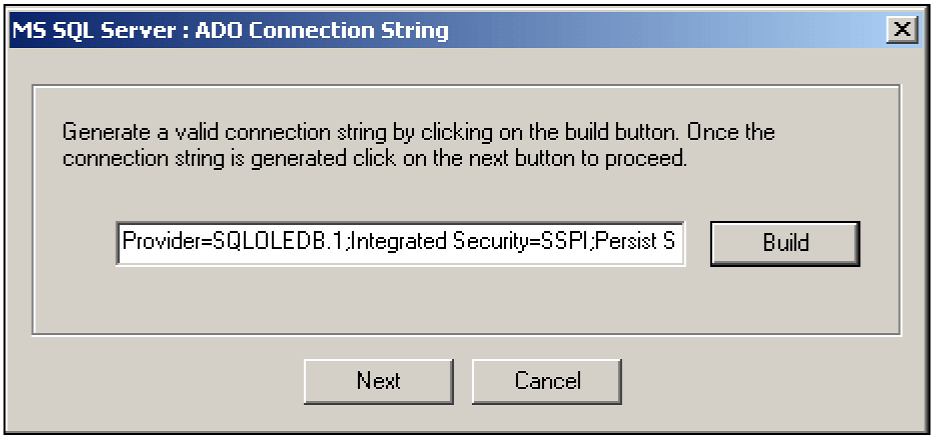
2. Select the provider (driver) to connect to the DB. (For SQL DBs, we recommend that you set up and use **Microsoft OLE DB Provider for SQL Server**. For Oracle, MySQL, Sybase, and IBM DB2 DBs (lower than version 9), we recommend that you set up and use **Microsoft OLE DB Provider for ODBC Drivers**.) Then click **NEXT>>**. This takes you to the Connection tab.



Specify the following to connect to SQL Server data:

1. Select or enter a server name: TS-SQL
2. Enter information to log on to the server:
   - ● Use Windows NT Integrated security
   - ○ Use a specific user name and password:
   - User name:
   - Password:
   - □ Blank password □ Allow saving password
3. ● Select the database on the server: Northwind
   - ○ Attach a database file as a database name:
   - Using the filename:

3. Enter the name of the server, server log-on information, and the DB name, all of which are user-specific. If, for the log-on information, you enter a required password, note that **you must check "Allow saving password"** to save the password in the connection string. If a password is required, it must be saved in the connection string. Otherwise the connection will fail.
4. Click **Test Connection** to test whether a connection can be successfully made with the connection string you have built. If a password has been entered, it will be used for testing the connection. However, the password will be saved in the connection string only if the "Allow saving password" option has been checked (*see Step 3*) .
5. If the connection test fails, rebuild the connection string correctly. If the connection test is successful, click **OK** to complete. The connection string you build is entered in the ADO Connection String dialog. A password will be entered in the connection string only if the "Allow saving password" option is checked.



Generate a valid connection string by clicking on the build button. Once the connection string is generated click on the next button to proceed.

Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist S

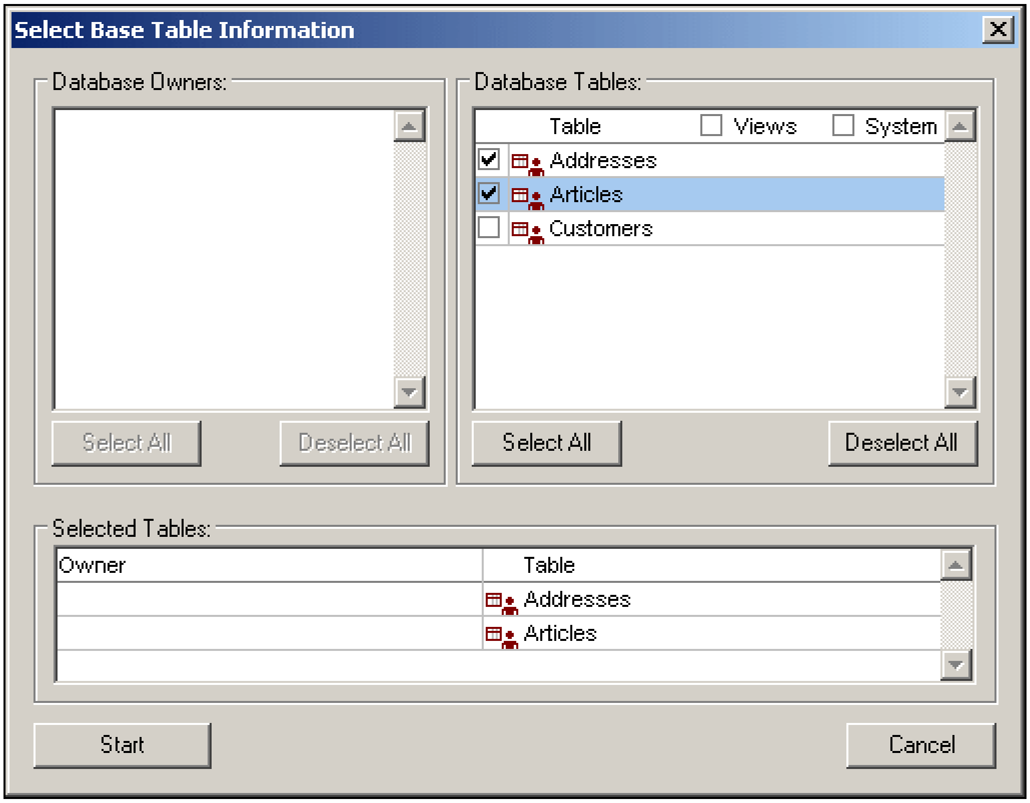
6. Click **Next**. The connection to the DB is made, and the Select Base Table Information dialog is displayed.

Note:

- If a password is required to access the DB, it must be saved in the connection string.
- All the connection information is stored in the SPS in the form of a connection string. It is therefore important that all clients that use this SPS must have the same driver installed so that they will correctly use the connection string.
- Since an ADO connection is used, hierarchical relationships in Oracle and Sybase DBs are ignored and only flat schemas are generated from these DBs.

Selecting Base Table Information

The Select Base Table Information dialog is displayed when StyleVision connects successfully to a DB. This dialog displays DB tables by owner.



Select Base Table Information

Database Owners:

Select All   Deselect All

Database Tables:

Table	□ Views	□ System
☑ Addresses		
☑ Articles		
□ Customers		

Select All   Deselect All

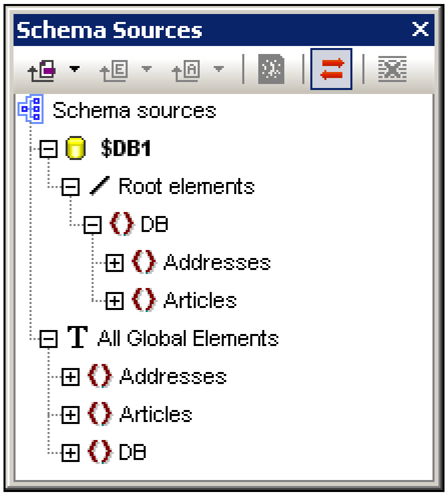
Selected Tables:

Owner	Table
	Addresses
	Articles

Start   Cancel

To select DB tables, do the following:

1. In the DB Owners pane, check the owners (if any), for which you want DB tables displayed in the DB Tables pane.
2. In the DB Tables pane, select the tables you wish to use in the SPS. The selected tables are shown in the Selected Tables pane. Note that some tables contain other tables. If the contained tables are not selected, they will not be created in the generated XML Schema.
3. Click **Start** to generate and load the XML Schema. An XML Schema with a structure corresponding to that of the DB with the selected tables is displayed in the Schema Window.



Schema Sources

```
Schema sources
⊟ $DB1
   ⊟ Root elements
      ⊟ DB
         ⊞ Addresses
         ⊞ Articles
   ⊟ All Global Elements
      ⊞ Addresses
      ⊞ Articles
      ⊞ DB
```

Note that all the selected DB tables are created in the XML Schema as children of the

DB document element and as items in the Global Templates list. For a complete description of the structure of the generated XML schema, see The DB Schema and DB XML files.

After you have connected to the DB and generated the XML Schema, you can use the full range of StyleVision features to design an SPS for the DB.

Note: The XML Schema that is generated from the DB will not be altered by any DB Filter that may be built subsequently.

Network shares and UNC paths

Any folder on your computer's hard drive can be marked as a 'share'. When this has been done, any other computer on the network can access it and even write to it depending on how the share has been set up. A share can therefore be used on a remote machine exactly as if it were a local folder.

When working with an SPS based on a DB which is on a hard drive (such as an MS Access DB), the folder in which the DB is located must be marked as a share. A UNC path is then used in the connection string to point to the DB. This enables the SPS to connect correctly to the DB when used by clients (such as Authentic Browser) on other machines.

There are two parts to setting up the network share mechanism for a DB-based SPS.

- The folder in which the DB is located must be set up for sharing. (In Windows 2000 and Windows XP, the sharing settings are accessed by right-clicking the folder and selecting **Sharing and Security...**.) You must also enable Advanced File Sharing (**My Computer | Tools | Folder Options**, then uncheck Simple File Sharing).
- Accessing the share and the DB from the remote location. To do this in a DB-based SPS, you have to set up the connection string with a UNC path. The format of a UNC path is: `\\servername\sharename\path\file.mdb`, where `servername` is the name of the server, `sharename` is the name of the share, `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.

Note: Network shares and UNC paths are to be used for DBs, such as MS Access, which do not require any driver and are located on a server on a network.

Known issues with DB editing

You should be aware of the following issues:

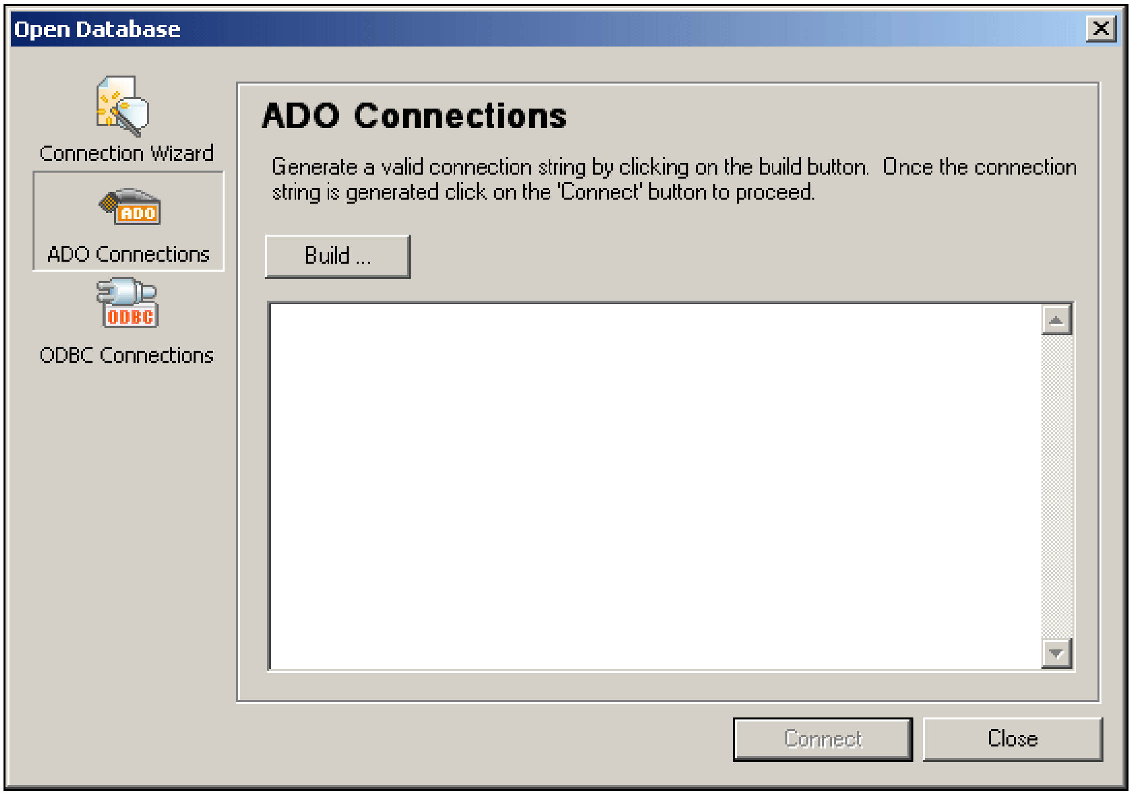
- If IBM DB2 table or field names contain blanks or non-ASCII characters, DB2's ADO driver is known to cause an Invalid Cursor Name error to be displayed.
- The date value `0001-01-01` is defined as a `NULL` value for some DBs, and could result in an error message.

Setting Up an IBM DB2 SPS

StyleVision supports DBs that contain XML fields (or columns). Currently, only IBM DB2 XML databases (version 9 and above) are supported. The procedure for setting up an SPS based on and connected to an XML DB is different than for DBs without XML fields. It consists of two broad parts: (i) connecting to the XML DB; and (ii) selecting the XML data and schema for the SPS.

Connecting to the XML DB

On clicking **File | New | New from XML Column in IBM DB2** or the **Add Schema from XML Column in IBM DB2** command in the Schema Sources entry helper, the Open Database dialog (*screenshot below*) appears.



Generate a valid connection string by clicking on the build button. Once the connection string is generated click on the 'Connect' button to proceed.

The options for connecting to the database are:

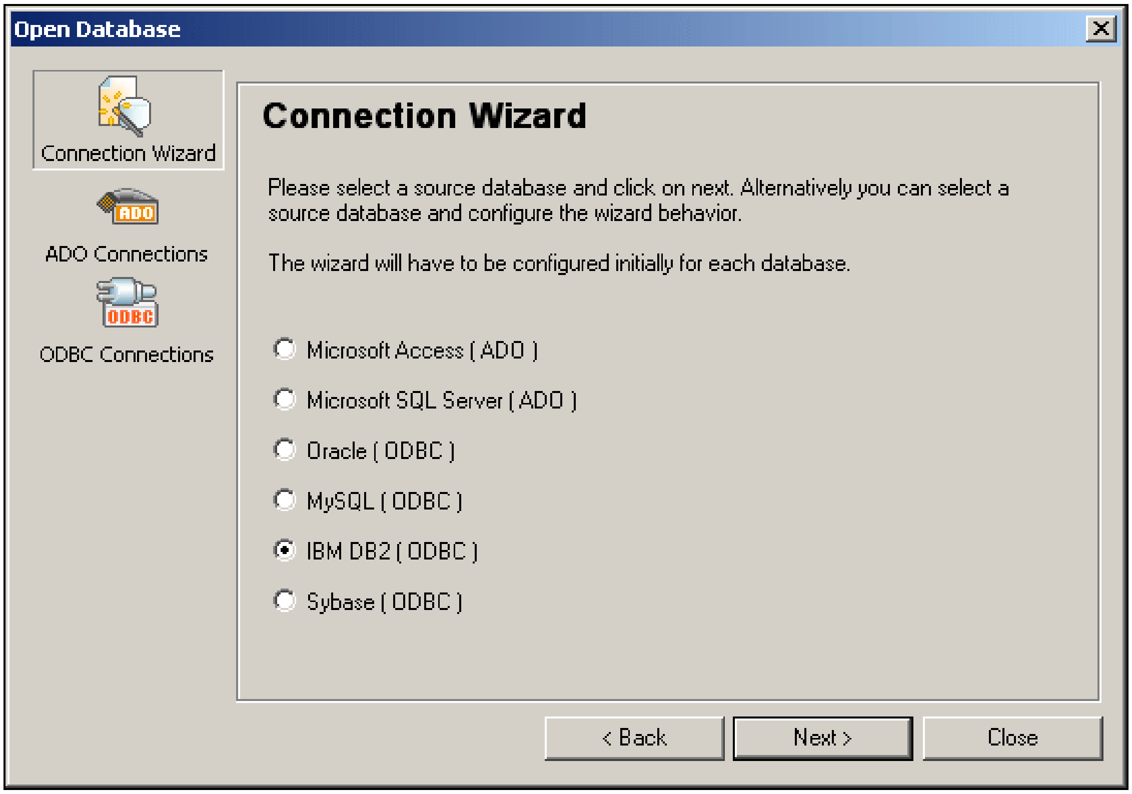
- Using a Connection Wizard that guides you through the connection process.
- Using an ADO connection.
- Using an ODBC connection.

Selecting the XML data and schema for the SPS

After the connection to the database has been made, you must select the XML cell of the DB in which the required XML data is stored. This XML data is then loaded as the Working XML File of the SPS. The final step in the setup procedure is to select the schema on which the SPS will be based. How to select the XML data and schema is explained in the section, Selecting the XML Data and Schema.

Connection Wizard

When the Connection Wizard button is selected, the Connection Wizard (*screenshot below*) pops up. Select IBM DB2 and click **Next**.



Connection Wizard

Please select a source database and click on next. Alternatively you can select a source database and configure the wizard behavior.

The wizard will have to be configured initially for each database.

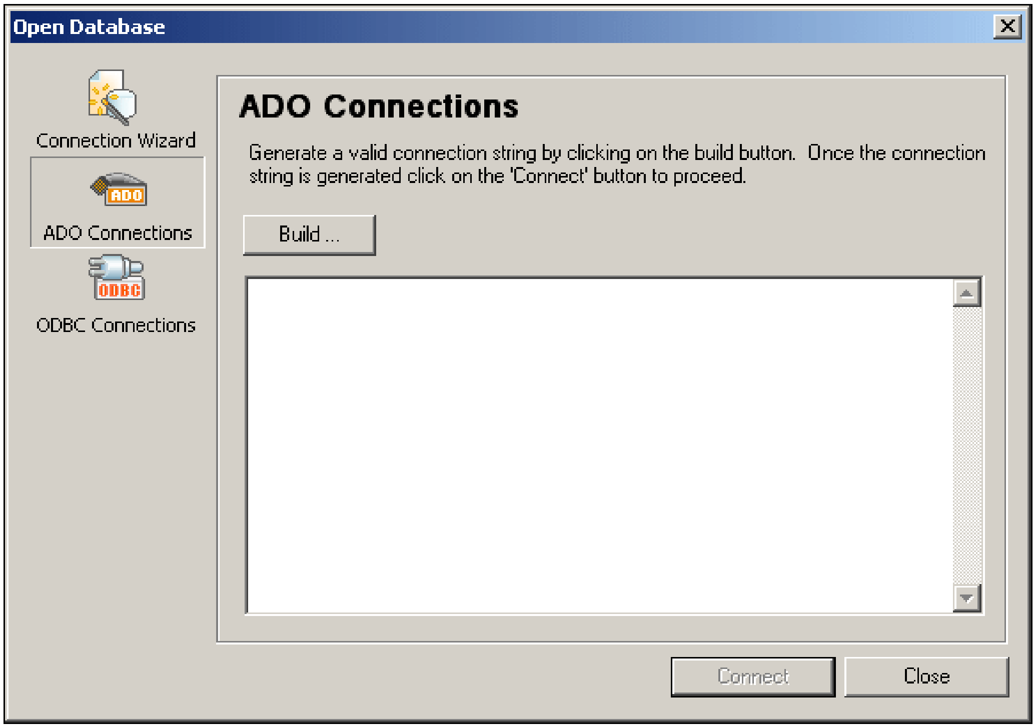
○ Microsoft Access (ADO)
○ Microsoft SQL Server (ADO)
○ Oracle (ODBC)
○ MySQL (ODBC)
● IBM DB2 (ODBC)
○ Sybase (ODBC)

The wizard will then guide you in configuring the connection to the IBM DB2 database and making the connection. These steps consist essentially of selecting the appropriate driver to connect to the DB, then locating the DB, and entering user information that enables you to connect. After the connection is made, the Select XML Table dialog pops up, in which you select the table that contains the required XML data.

ADO Connections

To connect to an XML DB via ADO, do the following:

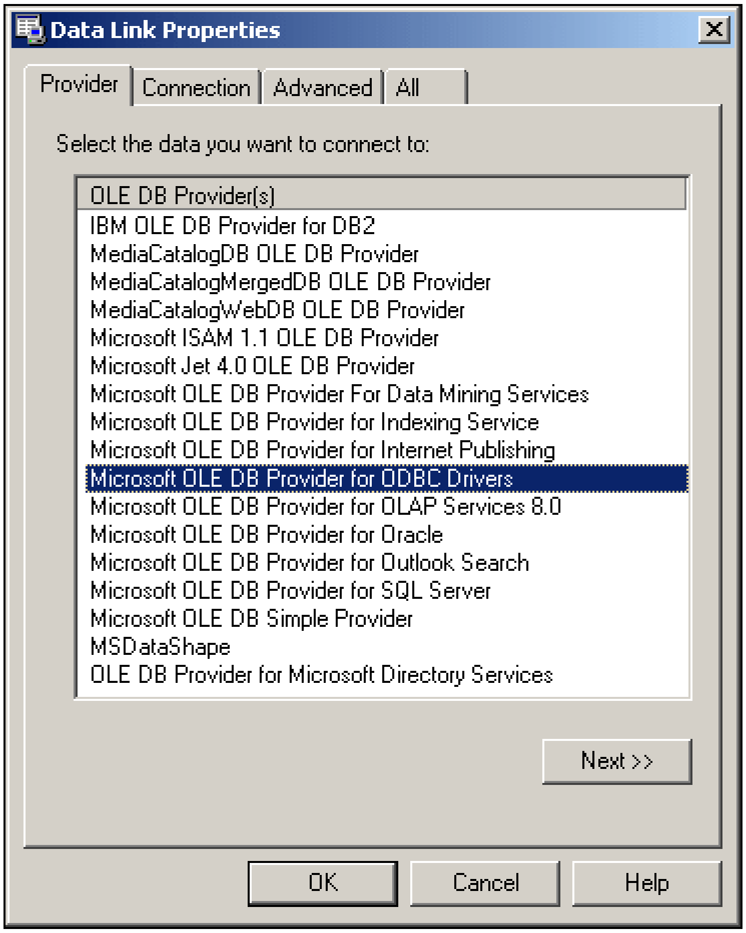
1. Select **ADO Connections**.



ADO Connections

Generate a valid connection string by clicking on the build button. Once the connection string is generated click on the 'Connect' button to proceed.

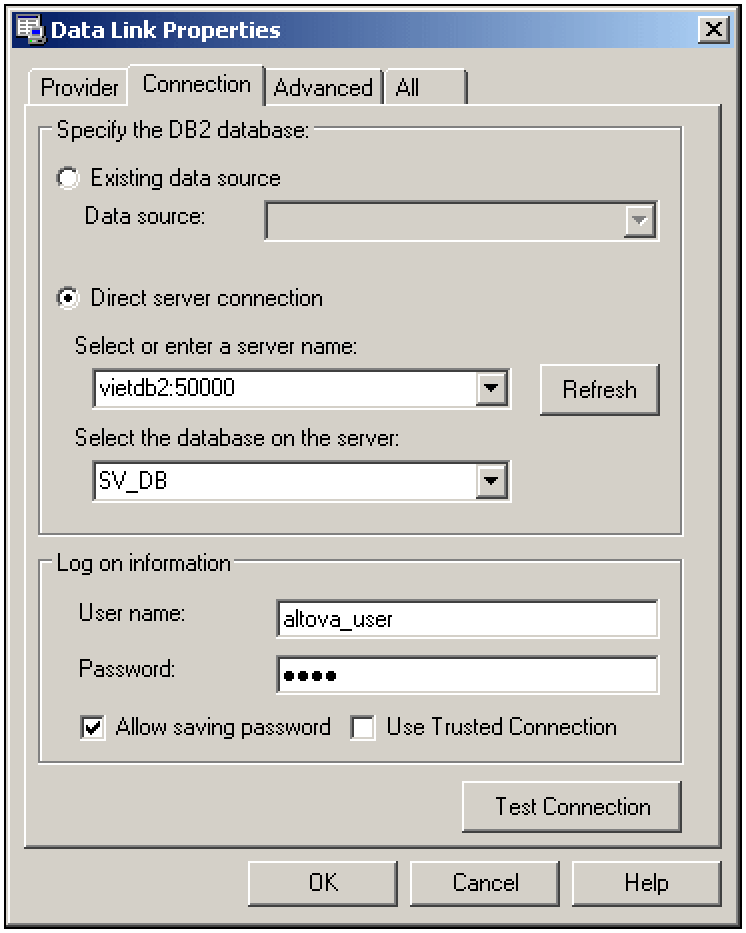
2. Click the **Build** button. The Data Link Properties dialog pops up.
3. In the Provider tab of the Data Link Properties dialog, select Microsoft OLE DB Provider for ODBC Drivers (*screenshot below*), and click **Next**.



Select the data you want to connect to:

OLE DB Provider(s)
IBM OLE DB Provider for DB2
MediaCatalogDB OLE DB Provider
MediaCatalogMergedDB OLE DB Provider
MediaCatalogWebDB OLE DB Provider
Microsoft ISAM 1.1 OLE DB Provider
Microsoft Jet 4.0 OLE DB Provider
Microsoft OLE DB Provider For Data Mining Services
Microsoft OLE DB Provider for Indexing Service
Microsoft OLE DB Provider for Internet Publishing
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services 8.0
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for Outlook Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Services

4. In the Connection tab (*screenshot below*), select Direct Server Connection, and in the combo box for the server, select the server on which the DB is located. Then select the DB on that server.



Specify the DB2 database:

○ Existing data source

Data source:

● Direct server connection

Select or enter a server name: vietdb2:50000

Select the database on the server: SV_DB

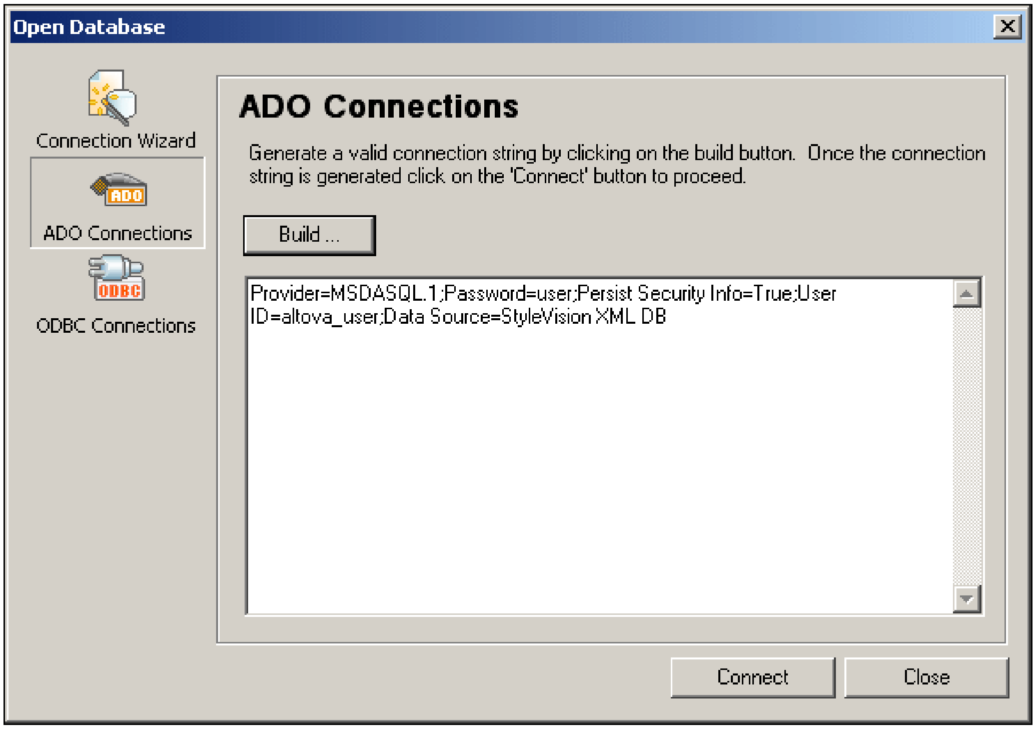
Log on information

User name: altova_user

Password: ●●●●

☑ Allow saving password ☐ Use Trusted Connection

5. Enter your user name and password.
6. Test the connection to the DB by clicking the **Test Connection** button. If the test fails, you will have to correct the connection data.
7. After the connection has been successfully tested, check the Allow Saving Password check box. This step is necessary to save the password information in the connection string. The Connection tab of the Data Link Properties should look something like this when you are done.
8. Click **OK**. The connection string you have built in the Data Link Properties dialog is entered in the Open Database dialog.



Generate a valid connection string by clicking on the build button. Once the connection string is generated click on the 'Connect' button to proceed.

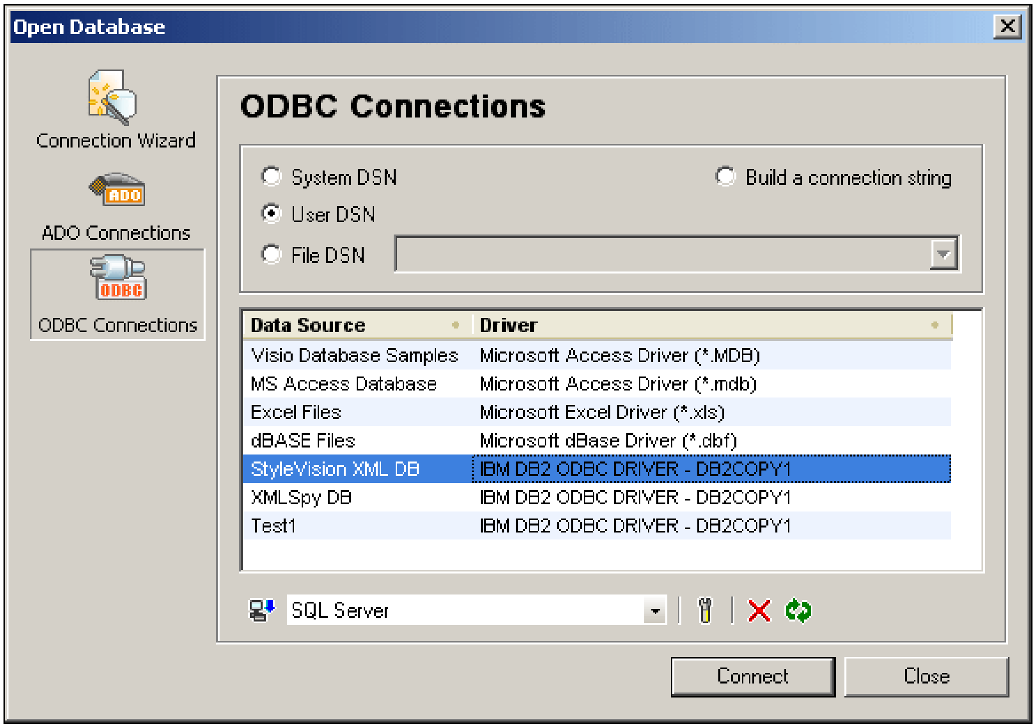
Provider=MSDASQL.1;Password=user;Persist Security Info=True;User ID=altova_user;Data Source=StyleVision XML DB

9. Click **Connect** to make the connection to the database. The connection is made and the Select XML Table dialog appears. How to select the XML tables is described later in this section.

ODBC Connections

To connect to an XML DB via ODBC, do the following:

1. Select **ODBC Connections**.



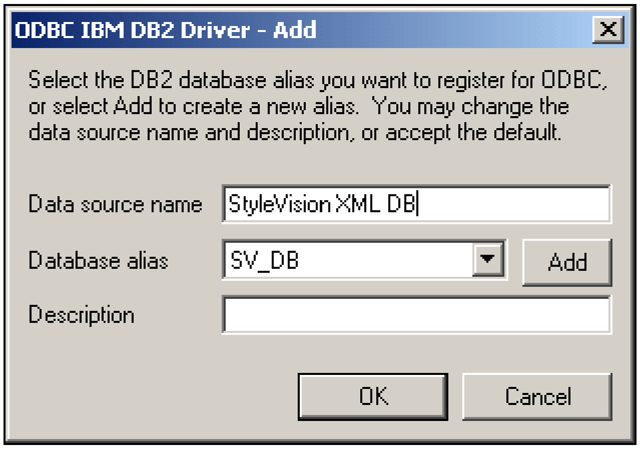
ODBC Connections

○ System DSN
● User DSN
○ File DSN
○ Build a connection string

Data Source	Driver
Visio Database Samples	Microsoft Access Driver (*.MDB)
MS Access Database	Microsoft Access Driver (*.mdb)
Excel Files	Microsoft Excel Driver (*.xls)
dBASE Files	Microsoft dBase Driver (*.dbf)
StyleVision XML DB	IBM DB2 ODBC DRIVER - DB2COPY1
XMLSpy DB	IBM DB2 ODBC DRIVER - DB2COPY1
Test1	IBM DB2 ODBC DRIVER - DB2COPY1

SQL Server

2. Select one of the options for specifying a Data Source Name (DSN). If you select System DSN or User DSN, the available DSNs are displayed in the Data Source pane. If you select File DSN, you are prompted to select the folder which contains the File DSNs. Alternatively, you can build a connection string to the DB by selecting the Build a Connection String option.
3. If you wish to add a DSN to those in the Data Source pane, do the following: (i) click the dropdown arrow of the Create a New Data Source icon ; (ii) select the required driver; (iii) in the dialog that appears (*screenshot below*), select the DB alias and give it a DSN.



Select the DB2 database alias you want to register for ODBC, or select Add to create a new alias. You may change the data source name and description, or accept the default.

Data source name: StyleVision XML DB

Database alias: SV_DB

Description:

(iv) click **OK** to finish. The DB is added as a Data Source to the list in the Data Source pane.

4. Select the Data Source in the Data Source pane and click Connect.
5. When you are prompted for your user ID and password, enter these and then click **OK**. The connection is made and the Select XML Table dialog (*screenshot below*) appears. How to select the XML tables is described below.

Selecting the XML Data and Schema

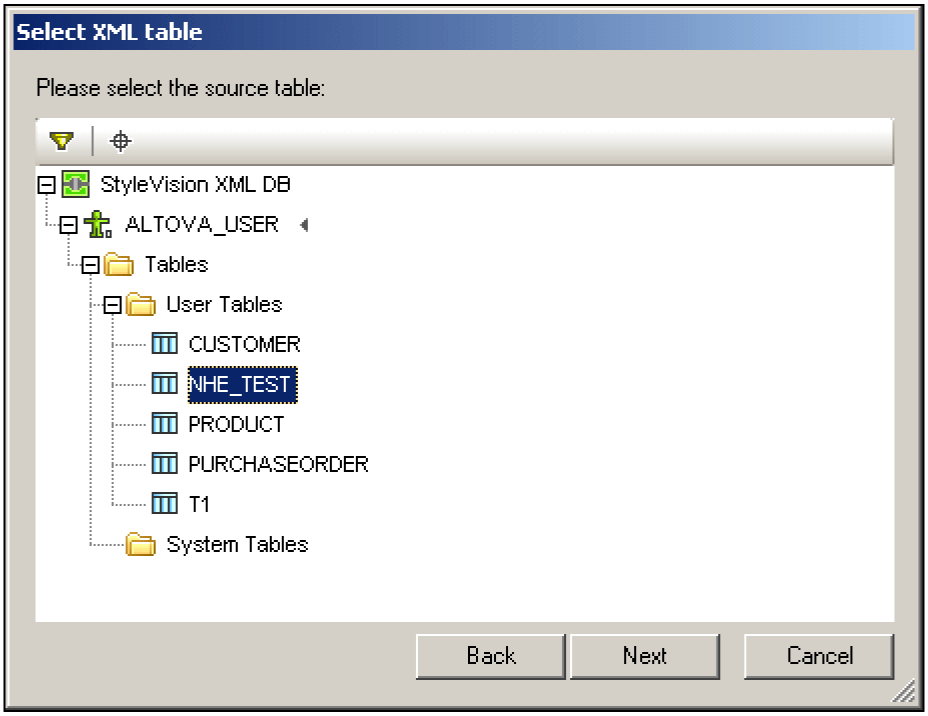
After having made the connection to the IBM DB2 database via the Open Database dialog (see previous sections), you will need to do two things:

- Select the cell in the DB that contains the required XML document. The XML document will be loaded automatically as the Working XML File.
- Select the XML Schema for the SPS can be generated will be generated for use as the XML Schema on which the SPS is based. The procedure for doing this as follows.

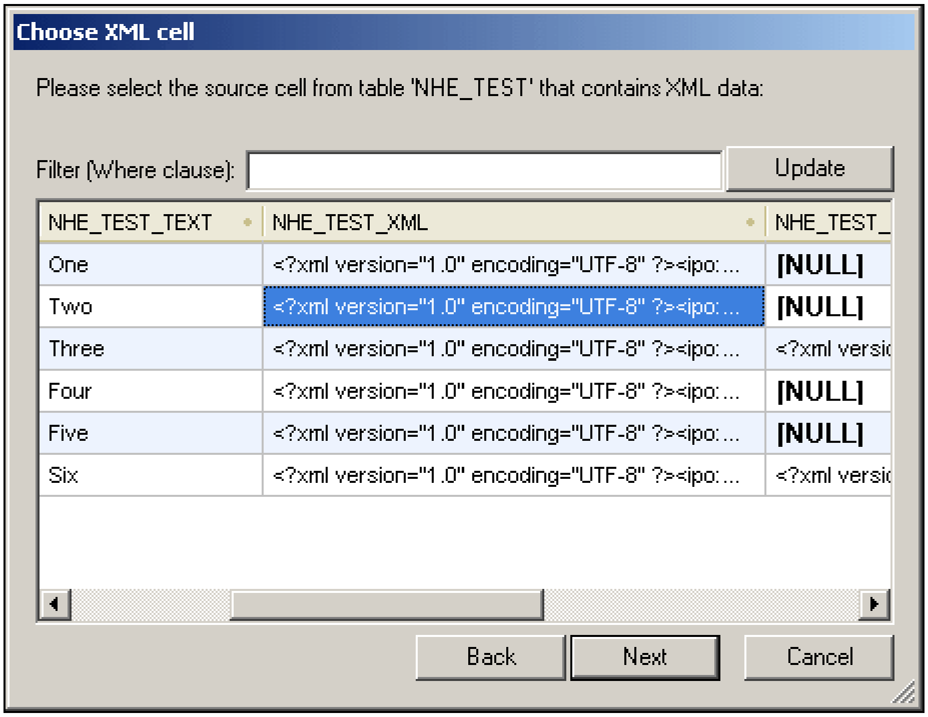
Selecting the XML Cell and Working XML File

After making the connection to the IBM DB2 database, the Select XML Table dialog (*screenshot below*) appears.

1. In the Select XML Table dialog, select the table that contains the XML data you wish to create as the Working XML File. In the screenshot below, the table `NHE_TEST` has been selected.



2. Click **Next**. This pops up the Choose XML Cell dialog (*screenshot below*). If you wish to filter the selection displayed in the pane, enter an SQL `WHERE` clause and click **Update**. Note that the `WHERE` clause should be just the condition (without the `WHERE` keyword, for example: `NHE_TEST_TEXT= 'Two'`)



Please select the source cell from table 'NHE_TEST' that contains XML data:

Filter (Where clause):

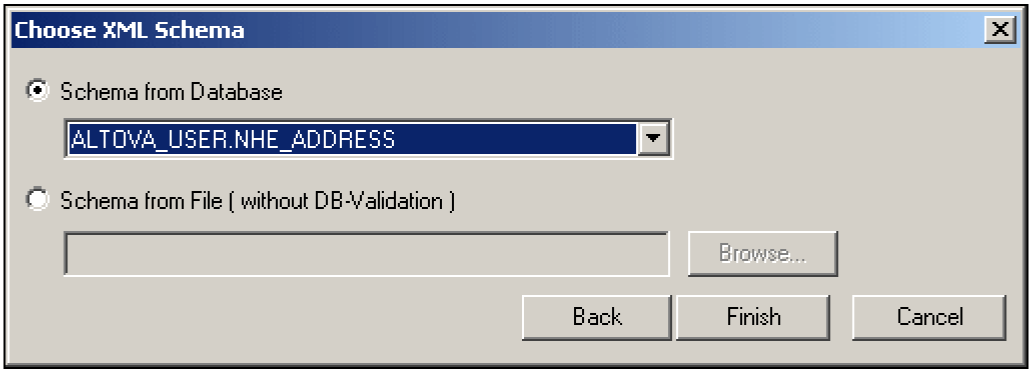
NHE_TEST_TEXT	NHE_TEST_XML	NHE_TEST_
One	<?xml version="1.0" encoding="UTF-8" ?><ipo:...	[NULL]
Two	<?xml version="1.0" encoding="UTF-8" ?><ipo:...	[NULL]
Three	<?xml version="1.0" encoding="UTF-8" ?><ipo:...	<?xml versic
Four	<?xml version="1.0" encoding="UTF-8" ?><ipo:...	[NULL]
Five	<?xml version="1.0" encoding="UTF-8" ?><ipo:...	[NULL]
Six	<?xml version="1.0" encoding="UTF-8" ?><ipo:...	<?xml versic

3. Select the cell containing the XML data you wish to create as the Working XML File. In the screenshot above, the selected cell is highlighted in blue.
4. Click **Next**. This pops up the Choose XML Schema dialog, in which you select the schema to be used for the SPS. *See next section*.

Selecting the XML Schema for the SPS

The schema that will be used for the SPS can be either an XML Schema contained in the DB or a schema at a file location that can be accessed by StyleVision. To select the schema, do the following:

1. In the Choose XML Schema dialog (*screenshot below*), select the appropriate radio button according to whether you wish to select the schema from among those stored in the DB or from a file location. Note that if a non-DB schema is selected—that is, a schema from an external file—then no DB validation will be carried out.



● Schema from Database

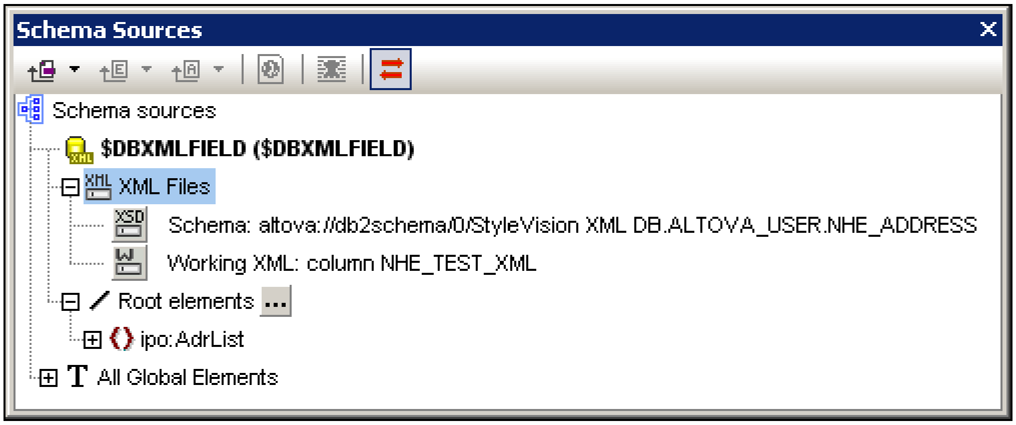
ALTOVA_USER.NHE_ADDRESS

○ Schema from File (without DB-Validation)

2. Select the schema. Schemas stored in the DB are listed in the dropdown list of the Schemas from Database combo box, and can be selected from there. An external schema can be selected by browsing for it.
3. Click **Finish** to complete.

The Schema Sources tree

After completing the process to select the XML data and the schema, the selected XML data is created as the Working XML File and the schema is loaded into the SPS. Both are displayed in the Schema Sources window (*screenshot below*).



Schema sources
$DBXMLFIELD ($DBXMLFIELD)
XML Files
Schema: altova://db2schema/0/StyleVision XML DB.ALTOVA_USER.NHE_ADDRESS
Working XML: column NHE_TEST_XML
Root elements ...
ipo:AdrList
All Global Elements

The SPS can now be built using the usual StyleVision mechanisms. Note that the data in the Working XML File can be edited in Authentic View and saved to the DB.

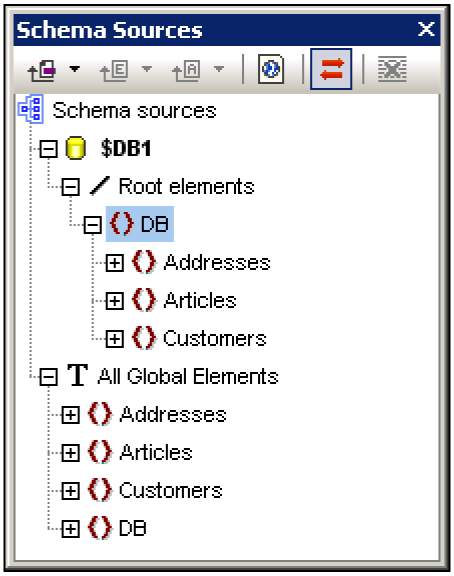
Note: The Working XML File should be valid against the schema selected for the SPS. Also ensure that the schema's root element (document element) corresponds to the root element of the XML document.

The DB Schema and DB XML files

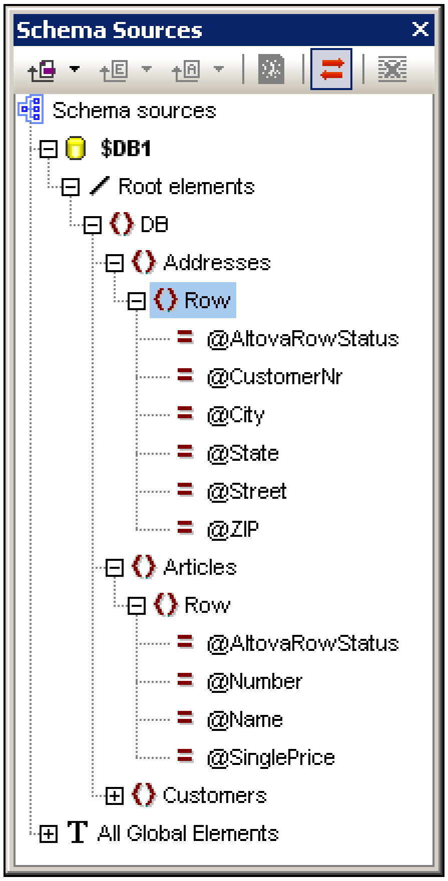
The DB XML Schema

When you load a non-XML database (non-XML DB) into StyleVision, an XML Schema with a structure based on that of the DB is generated by StyleVision and displayed in the Schema Window. (In the case of XML DBs, an existing schema (either stored in the DB or at a file location) is specified as the schema to be used in the SPS.) This section on schemas, therefore, refers only to non-XML DBs.

The XML Schema is created with a document element called `DB`. The `DB` element contains child elements which correspond to the top-level tables in the DB. These top-level table elements are also created as entries in the Global Templates list in the Schema Window. The top-level elements in the screenshot below are: `Addresses`, `Articles`, and `Customers`; they correspond to tables in the DB.



Each top-level table element may have an unlimited number of rows (*see screenshot below*). Each row corresponds to a record in the DB. In the schema tree the rows are represented by a single `Row` element. Each `Row` element has attributes which correspond to the fields of the table. One of these attributes is generated by StyleVision for every row of every table: `AltovaRowStatus`, which holds the current status of the row: added, updated, and/or deleted. The remaining attributes are the fields of the respective DB table.



Note: The structure of the generated XML Schema is as outlined above. Whatever tables are selected during the connection step are included in the structure. The construction of a DB Filter does not affect the structure of the XML Schema.

New DB Schema Structure

The structure of the XML Schema generated from DBs starting with the 2005 version of StyleVision is different than the structure generated in previous versions of StyleVision. The new structure enables the editing of databases in the Authentic View of Altova products—a feature which was not available with earlier versions. As a result, any SPS generated with earlier versions of StyleVision will generate an error when opened in versions of StyleVision starting from the 2005 version. To be able to use the DB editing and reporting features of <%SV-PROD%>, you should recreate the SPS in the current version of StyleVision.

DB XML files

Two **temporary** XML files are generated from the DB (see DBs and StyleVision for an illustration):

- A temporary editable XML file, which can be edited in Authentic View
- A temporary non-editable XML File, which is used as the Working XML File (for previews and output generation)

The temporary **editable XML file** is generated when the DB is loaded into StyleVision. It can be edited in Authentic View after the SPS has been created. The display in Authentic View can be filtered by using the Query mechanism available in Authentic View. Any modification made in

Authentic View to the editable data is written to this temporary XML File. Clicking **File | Save Authentic XML Data** saves the information in the temporary editable XML file to the DB.

The temporary **non-editable XML file** is generated when the DB is loaded into StyleVision. It is used as the Working XML File and for generating HTML output. The editable XML file must be saved before changes made in Authentic View can be viewed in a preview.

Note:

- In the Authentic View of other Authentic View products only one temporary (editable) XML file is created when a DB-based SPS is opened. Modifications made in Authentic View are written to this file. When the file is saved, the information in the XML file is written to the DB.
- You can filter the data that goes into the non-editable temporary XML File for report-generation. (See Edit DB Filter for details.)
- You do not have to specifically assign a Working XML File in order to see HTML previews. The automatically generated (non-editable) temporary XML file is used for this purpose.

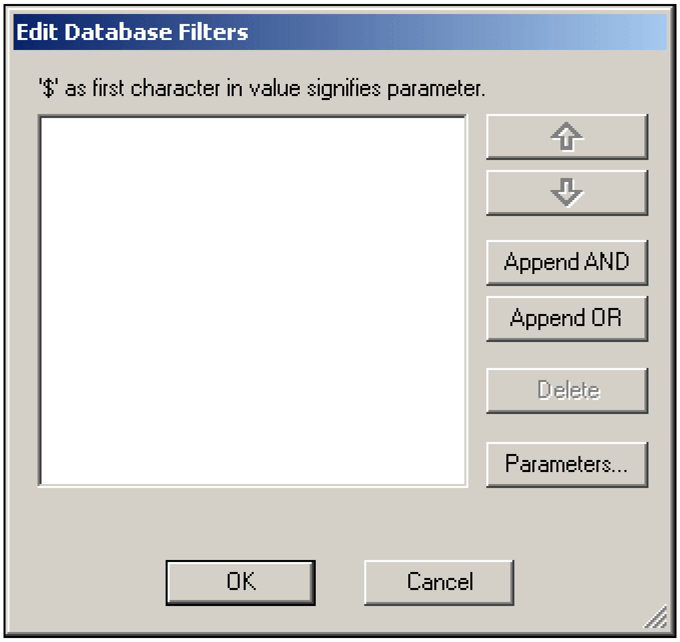
DB Filters: Filtering DB Data

The data that is imported into the temporary **non-editable** XML file from the database (DB) can be filtered. (Note that the non-editable XML file is used for report generation, and the effect of a DB filter will therefore be seen only in the HTML preview; not in Authentic Preview, which displays the temporary **editable** XML file, and not in Authentic View.) The DB filter (DB Filter) can be created either within the DB itself (if this is supported in your DB application), or it can be created within the SPS (SPS file). In the SPS, one DB Filter can be created for each top-level data table in the XML Schema (i.e. for the data tables that are the children of the `DB` element). Each time a DB Filter is created or modified, the data from the DB is re-loaded into the temporary non-editable XML file that is generated for the DB. In this way, DB Filters help you to keep the XML file down to an optimal size and to thus make processing for report generation more efficient.

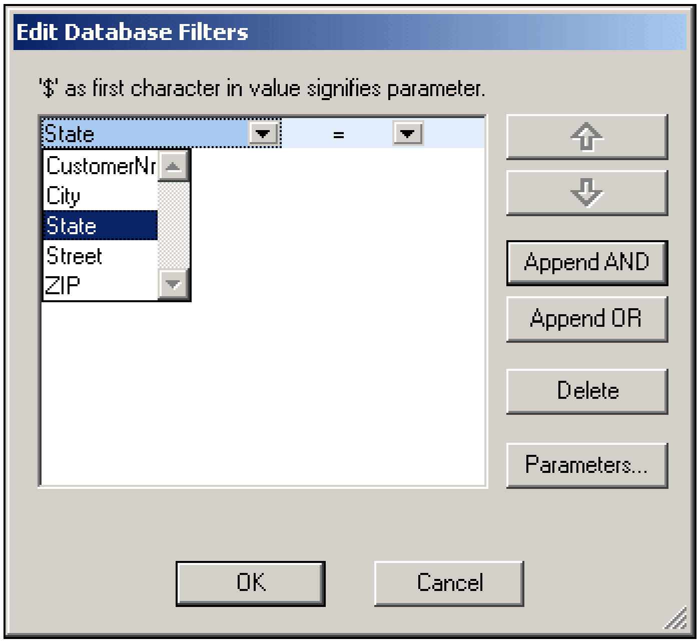
Note: Using a DB Filter modifies the data that is imported into the temporary non-editable XML File. If you save an SPS with a DB Filter and then generate an XML File from the SPS, the generated XML File will be filtered according to the criteria in the DB Filter.

Creating a DB Filter

1. In the Design Document or Schema Tree, select the data table element for which you wish to create a DB Filter (either by clicking the start or end tag of the element, or by selecting the element in the schema tree).
2. Select **Edit | Edit DB Filters** or click the icon in the toolbar. The following dialog is displayed:



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



'$' as first character in value signifies parameter.

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

Expressions in criteria

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

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

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

Using parameters with DB Filters

You can also enter the name of a parameter as the value of an expression. This causes the parameter to be called and its value to be used as the value of that expression. The parameter you enter here can be a parameter that has already been declared for the stylesheet, or it can be a parameter that you declare subsequently.

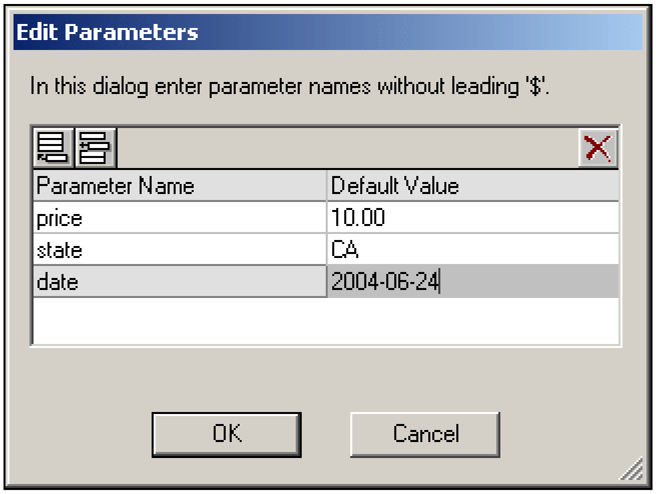
Parameters are useful if you wish to use a single value in multiple expressions, or if you wish to pass a value to a parameter from the command line (see StyleVision from the command line for details).

To enter the name of a parameter as the value of an expression, type $ into the value input field followed (without any intervening space) by the name of the parameter. If the parameter has already been declared (see Parameters), then the entry will be colored green. If the parameter has not been declared, the entry will be red, and you must declare it.

Declaring parameters from the Edit DB Filter dialog

To access the Edit Parameters dialog (in order to declare parameters), do the following:

1. Click the **Parameters...** button in the Edit DB Filters dialog. This pops up the Edit Parameters dialog shown below.



Edit Parameters

In this dialog enter parameter names without leading '$'.

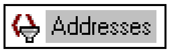
Parameter Name	Default Value
price	10.00
state	CA
date	2004-06-24

2. Type in the name and value of the parameter in the appropriate fields.

Alternatively, you can access the Edit Parameters dialog and declare or edit a DB Parameter by selecting **Edit | Edit Stylesheet Parameters**.

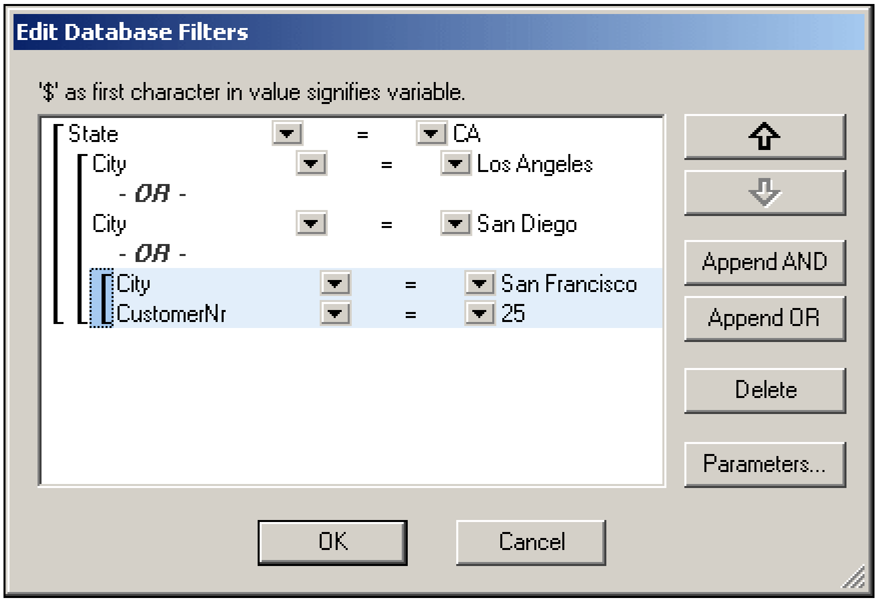
Note: The Edit Parameters dialog contains **all** the parameters that have been defined for the stylesheet. While it is an error to use an undeclared parameter in the SPS, it is not an error to declare a parameter and not use it.

After a DB Filter is created for a data table element, that element in the Schema Tree is displayed with the filter symbol, as shown for the Addresses element in the screenshot below.



Re-ordering criteria in DB Filters

The logical structure of the DB Filter and the relationship between any two criteria or sets of criteria is indicated graphically. Each level of the logical structure is indicated by a square bracket. Two adjacent criteria or sets of criteria indicate the AND operator, whereas if two criteria are separated by the word `OR` then the OR operator is indicated. The criteria are also appropriately indented to provide a clear overview of the logical structure of the DB Filter.



Edit Database Filters

'$' as first character in value signifies variable.

```
State         =  CA
  City        =  Los Angeles
    - OR -
  City        =  San Diego
    - OR -
    City        =  San Francisco
    CustomerNr  =  25
```

The DB Filter shown in the screenshot above may be represented in text as:

```
State=CA AND (City=Los Angeles OR City=San Diego OR (City=San
Francisco AND CustomerNr=25))
```

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

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

The following points should be noted:

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

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

Modifying a DB Filter

To modify a DB Filter, click **Edit | Edit DB Filters**. This pops up the Edit DB Filters dialog box. You can now edit the expressions in any of the listed criteria, add new criteria, re-order criteria, or delete criteria in the DB Filter. After you have completed the modifications, click OK. The data from the DB is automatically re-loaded into StyleVision so as to reflect the modifications to the DB Filter.

Clearing (deleting) a DB Filter

To clear (or delete) a DB Filter, select the element for which the DB Filter has to be cleared either in the Design Window or the Schema Tree. (There is one DB Filter for each (top-level) data table element.) Then click **Edit | Clear DB Filter**. The filter will be cleared, and the filter symbol will no longer appear alongside the name of the element in the Schema Tree.

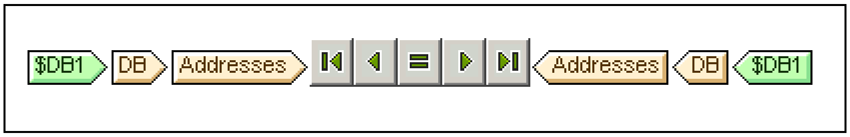
SPS Design Features for DB

You can design a DB-based SPS just as you would design any other schema-based SPS, that is by dragging-and-dropping schema nodes from the schema window into the design document; by inserting static content directly in the design document; and by applying suitable formatting to the various design components. The following points, however, are specific to DB-based SPSs.

Creating a dynamic table for a DB table

To create a dynamic table for a DB table, do the following:

1. In the schema tree, select the top-level DB table to be created as a dynamic table, drag it into the design.
2. When you drop it, create is as `contents` and delete the `contents` placeholder. If the Auto-Add DB Controls feature is on, your design will look something like this:

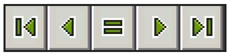


3. In the schema tree, select the `Row` element of the DB table you wish to create as a dynamic table.
4. Drag it to a location inside the `Addresses` element.
5. When you release the element, select Create Table from the menu that pops up, and select the DB fields you wish to create as columns of the dynamic table. The DB table is created as a dynamic table.

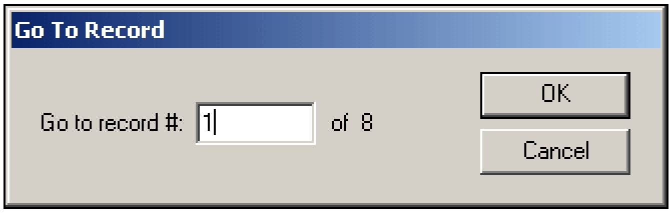
Note: You can also create a DB table in any other format, such as `( contents)` .

Auto-add DB Controls

The **Authentic | Auto-add DB Controls** menu command or the toolbar icon toggles the auto-insertion of navigation controls for DB tables on and off. When the toggle is switched on this toolbar icon has a black border; when the toggle is off, the toolbar icon has no border. If the Auto-insert toggle is on, DB Controls (*see screenshot below*) are automatically inserted when a DB table is dropped into the Design document. It is dropped, by default, immediately before the `Row` start tag.



These controls enable the Authentic View user to navigate the records of the DB table in Authentic View. The first (leftmost) button navigates to the first record; the second button navigates to the previous record; the third button is the Goto Record button, which pops up the Goto Record dialog (*screenshot below*); it pops up a dialog that prompts you for the number of the record to which you wish to go; the fourth button navigates to the next record; and the fifth button navigates to the last record.



Go to record #: 1 of 8

To manually insert the navigation controls in the Design document—which is useful if you wish to insert the controls at some other location than the default location—then do the following:

1. Turn the Auto-insert DB Controls toggle off.
2. Place the cursor at the location where you wish the navigation controls to appear (but within the DB table element's start and end tags).
3. Right-click, and from the popup menu, select **Insert | DB Control | Navigation** or **Insert | DB Control | Navigation+Goto**

Inserting a DB Query button for Authentic View

The DB Query button enables the Authentic View user to submit a DB query. This helps the user to build conditions for the records to be displayed in Authentic View. Query buttons can be inserted for individual DB tables anywhere between:

- The start tag of the DB table element and the start tag of the DB table element's (child) Row element.
- The end tag of the DB table element and the end tag of that table element's (child) Row element.

To insert a DB Query button in your Design document, do the following:

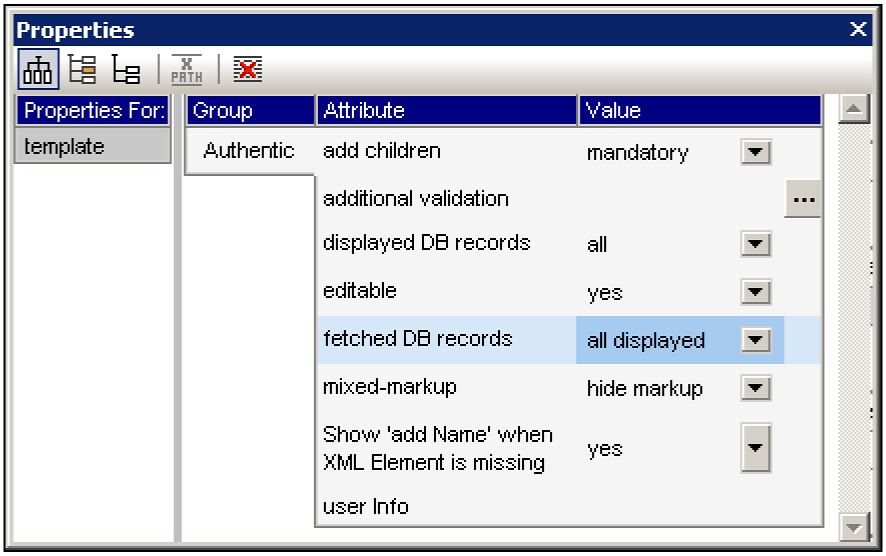
1. Place the cursor at the allowed location (*see above*) where you wish the Query button to appear. (in the screenshot below, the cursor is placed between the end `Row` tag and the end `Customer` tag.
2. Right-click, and from the context menu (or **Insert** menu), select **Insert | DB Control | Query Button**. The Query Button is inserted at that point in the Design document.

When it is clicked in Authentic View, the Query button will pop up the Edit Database Query dialog. This dialog is described in the Authentic View documentation.

Records displayed and records fetched

You can control the number of records displayed in Authentic View and the number of records fetched when the file is loaded. To make these settings, do the following:

1. In Design View, select the `Row` element that corresponds to the records to be displayed/fetched.
2. In the Properties entry helper, select `template` in the Properties For column, and the Authentic group of properties.



Properties For:	Group	Attribute	Value
template	Authentic	add children	mandatory
		additional validation	
		displayed DB records	all
		editable	yes
		fetched DB records	all displayed
		mixed-markup	hide markup
		Show 'add Name' when XML Element is missing	yes
		user Info	

3. For the property `displayed DB Records`, select `all` from the dropdown list or enter

the number of records to be displayed.

4. For the property `fetched DB Records`, select `all` or `all displayed` from the dropdown list, or enter the number of records to be fetched. The value `all displayed` is the default. This property determines how many records are fetched when the DB data is loaded. If the value is less than that of the `displayed DB Records` property, then the value of the `displayed DB Records` is used as the value of `fetched DB Records`. The `fetched DB Records` property enables you to reduce the number of records initially loaded thus speeding up the loading and display time. Additional records are loaded and displayed when the row (record) navigation buttons in Authentic View are used.

Note:

- The number of records displayed is applied to Authentic View.
- If a large number of tables is open in Authentic View, then the user will get an error message saying that too many tables are open. On the design side, you can reduce the number of tables that are open by reducing the number of records displayed (because a record can contain tables). On the user side, the user can use queries to reduce the number of tables loaded into Authentic View.

AltovaRowStatus

In the XML Schema that is created from a DB, each table has an **`AltovaRowStatus`** attribute. The value of this attribute is automatically determined by StyleVision and consists of three characters, which are initialized to `---`. If a row is modified, or a new row is added, the value is changed using the following characters.

`A`	The row has been added (but not yet saved to the DB).
`U, u`	The row has been updated (but not yet saved to the DB).
`D, d, X`	The row has been deleted (but not yet saved to the DB).

These values can be used to provide users with information about rows being edited. The status information exists up to the time when the file is saved. After the file is saved, the status information is initialized (indicated by `---`).

Formatting design document components

When records are added, modified, or deleted, StyleVision formats the added/modified/deleted records in a certain way to enable users to distinguish them from other records. Datatype errors are also flagged by being displayed in red. If you wish to maintain this differentiation, make sure that the formatting you assign to rows in a table do not have the same properties as those assigned by StyleVision. The formatting assigned by StyleVision is as follows:

Added	`A`	Bold, underlined
Modified (Updated)	`U, u`	Underlined
Deleted	`D, d, X`	Strikethrough
Datatype error		Red text

Generating Output Files

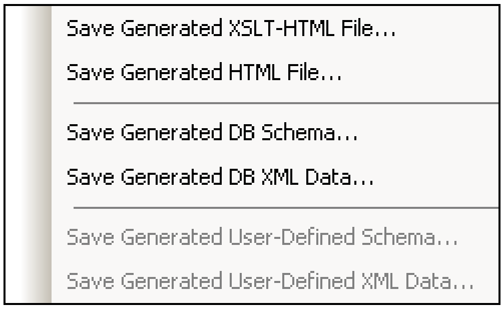
After you have created a DB-based SPS, you can generate files related to it and save them. The following files can be generated and saved:

- The XML Schema based on the DB structure
- The XML file having the structure of the generated XML Schema and content from the DB
- The XSLT file for HTML output
- The HTML output file

The files can be generated and saved from within the GUI or from the command line.

From within the StyleVision GUI

1. In the **File** menu, select the **Save Generated Files** item. This pops up the following submenu.


Save Generated XSLT-HTML File...
Save Generated HTML File...
Save Generated DB Schema...
Save Generated DB XML Data...
Save Generated User-Defined Schema...
Save Generated User-Defined XML Data...

2. Select the file you wish to generate. This pops up the Save As dialog.
3. Browse for the desired folder, enter the desired filename, and click **OK**.

From the command line

From the command line, you can call StyleVision so that it generates and saves files associated with a DB-based SPS. You can save not only the XML Schema and XSLT files, but also an XML file with data from the DB, and HTML output files based on the design in the SPS. For a description of how to use the command line, see StyleVision from the command line.

6.4 HTML Import

In StyleVision you can import an HTML file and create the following documents based on it:

- An SPS document based on the design and structure of the imported HTML file.
- An XML Schema, in which HTML document components are created as schema elements or attributes. Optionally, additional elements and attributes that are not related to the HTML document can be created in the user-defined schema.
- An XML document with: (i) a structure based on the XML Schema you have created, and (ii) content from the HTML file.
- XSLT stylesheets based on the design in Design View.

HTML-to-XML: step-by-step

The HTML Import mechanism, which enables the creation of XML files based on the imported HTML file, consists of the following steps:

1. Creating New SPS via HTML Import. When an HTML file is imported into StyleVision, a new SPS document is created. The HTML document is displayed in Design View with HTML markup tags. A user-defined XML Schema with a document element called `Root` is created in the Schema Sources window. This is the schema on which the SPS is based. The HTML document content and markup that is displayed in Design View at this point is included in the SPS as static content.
2. Creating the Schema and SPS Design. Create the schema by (i) dragging components from the HTML document to the required location in the schema tree (in the Schema Sources window); and, optionally, (ii) adding your own nodes to the schema tree. In the Design Window, HTML content that has been used to build nodes in the schema tree will now be displayed with schema node tags around the content. HTML content that has no corresponding schema node will continue to be displayed without schema node tags.
3. In the Design Document, assign formatting to nodes, refine processing rules, or add static content as required. These modifications will have an effect only on the SPS and the generated XSLT. It will not have an effect on either the generated schema or XML file.
4. After you have completed the schema tree and the design of the SPS, you can generate and save the following:

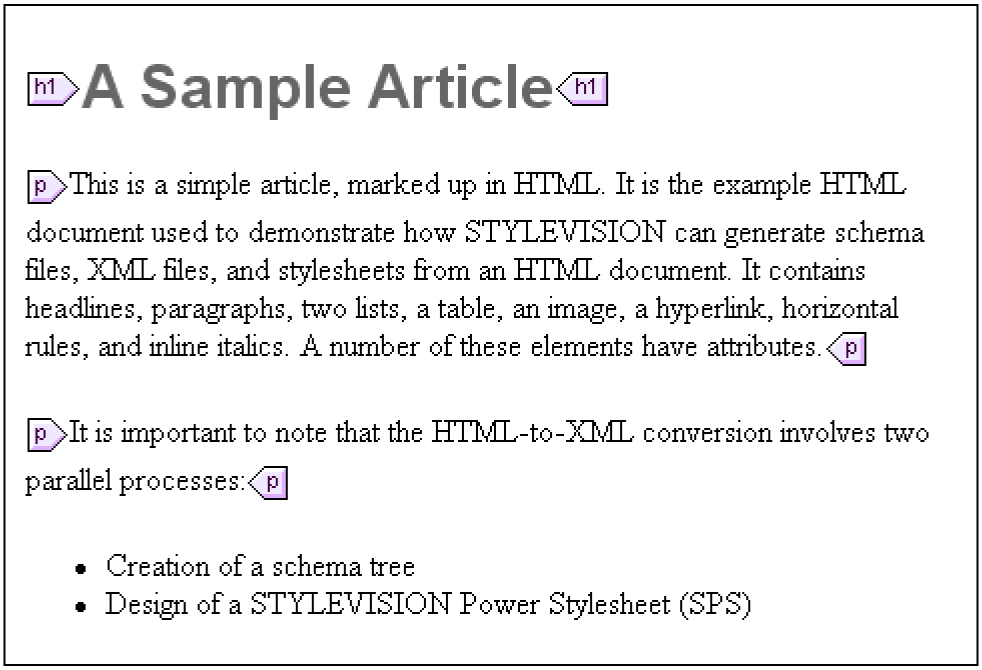
   - an XML Schema corresponding to the schema tree you have created;
   - an XML data file with a structure based on the schema and content for schema nodes that are created with the `(content)` placeholder in the SPS design;
   - a SPS (`.sps` file) and/or XSLT stylesheet based on your design.

Creating New SPS via HTML Import

To create a new SPS file from an HTML document, do the following:

1. Select the menu command **File | New | New from HTML File**.
2. In the Open dialog that pops up, browse for the HTML file you wish to import. Select it and click **Open**.

A new SPS document is created. The document is displayed in Design View and is marked up with the predefined HTML formats available in StyleVision (*screenshot below*).



A Sample Article

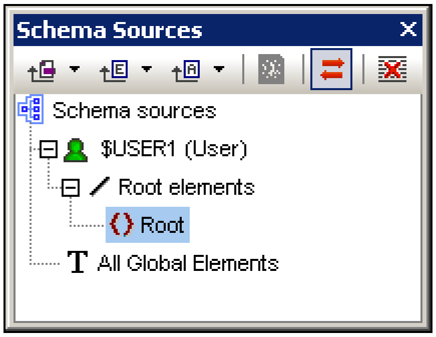
This is a simple article, marked up in HTML. It is the example HTML document used to demonstrate how STYLEVISION can generate schema files, XML files, and stylesheets from an HTML document. It contains headlines, paragraphs, two lists, a table, an image, a hyperlink, horizontal rules, and inline italics. A number of these elements have attributes.

It is important to note that the HTML-to-XML conversion involves two parallel processes:

- Creation of a schema tree
- Design of a STYLEVISION Power Stylesheet (SPS)

Note that the HTML document is displayed within the main template. There is no global template.

In the Schema Sources entry helper, a user-defined schema is created (*screenshot below*) with a root element (document element) called `Root`.



Note that there is no global element in the All Global Elements list.

SPS structure and design

The SPS contains a single template—the main template—which is applied to the document node of a temporary internal XML document. This XML document has the structure of the user-defined schema which was created in the Schema Sources window. In Design View, **at**

this point, the HTML document components within the main template are included in the SPS as static components. The representation of these HTML components in Authentic Viewwill be as non-editable, non-XML content. The XSLT stylesheets will contain these HTML components as literal result elements. The schema, at this point, has only the document element `Root`; consequently, the temporary internal XML document contains only the document element `Root` with no child node.

When you create HTML selections as elements and attributes in the user-defined schema, you can do this in either of two ways:

1. By **converting** the selection to an element or attribute. In the design, the node tags are inserted with a `( content)` placeholder within the tag. In the schema, an element or attribute is created. In the XML document, the selection is converted to the text content of the schema node which is created in the XML document. The contents of the node created in the XML document will be inserted dynamically into the output obtained via the SPS.
2. By **surrounding** the selection with an element or attribute. In the design, the selection is surrounded by the node tags; no `( content)` placeholder is inserted. This means that the selection is present in the SPS design as static content. In the schema, an element or attribute is created. In the XML document, the node is created, but is empty. The static text which is within the schema node tags in the design will be output; no dynamic content will be output for this node unless a `( content)` placeholder for this node is explicitly inserted in the design.

The significance of the `( content)` placeholder is that it indicates locations in the design where data from the XML document will be displayed (in the output) and can be edited (in Authentic View).

Creating the Schema and SPS Design

The schema is created by dragging selections from Design View into the user-defined schema. You do this one selection at a time. The selection is dropped on a node in the schema tree (relative to which the new node will be created, either as a child or sibling). You select the type of the node to be created (element or attribute) and whether the selection is to be converted to the new node or surrounded by it.

The selection

The selection in Design View can be any of the following:

- A node in the HTML document.
- A text string within a node.
- Adjacent text strings across nodes.
- An image.
- A link.
- A table.
- A list.
- A combination of any of the above.

In this section we explain the process in general for any selection. The special cases of tables and lists are discussed in more detail in the section Creating Tables and Lists as Elements/ Attributes.

To make a selection, click an HTML document component or highlight the required text string. If multiple components are to be selected, click and drag over the desired components to highlight the selection. Note that StyleVision extends the selection at the beginning and end of the selection to select higher-level elements till the first and last selected elements belong to the same parent.

The location in the schema tree

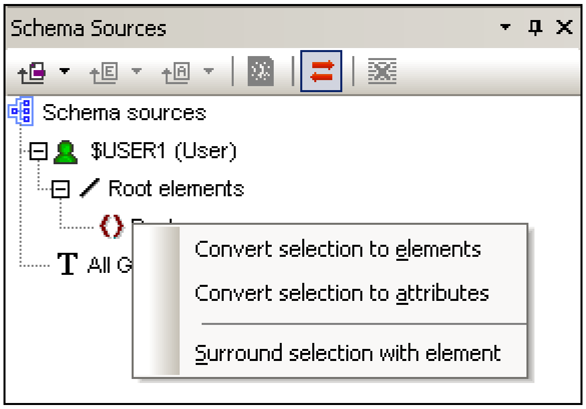
On dragging the selection over the desired schema tree node, one of the following symbols will appear together with the popup message: `Create new schema item`.

- Dropping the node when the Create as Sibling symbol ↓ appears, creates the selection as a sibling node of the node on which the selection is dropped.
- Dropping the node when the Create as Child symbol ↴ appears, creates the selection as a child node of the node on which the selection is dropped.

You should select the node on which the selection is to be dropped according to whether the selection is to be created as a sibling or child of that node.

Selecting how the node is created

When you drop the selection (*see previous section*), a context menu pops up (*screenshot below*) in which you make two choices: (i) whether the node is to be created as an element or attribute; (ii) whether the selection is to be converted to the node or whether the node is to simply surround the selection.


Schema Sources
Schema sources
$USER1 (User)
Root elements
All G
Convert selection to elements
Convert selection to attributes
Surround selection with element

The following points should be noted:

- When a selection is converted to a node (element or attribute), the node tags, together with a contained `(content)` placeholder, replace the selection in the design. In the design and the output the text content of the selection is removed from the static content. In the output, the text of the selection appears as dynamic content of the node in the XML document.
- If an HTML node is converted to an XML node, the XML node tags are inserted within the HTML node tags.
- When a selection (including HTML node selections) is surrounded by an XML node, the XML node tags are inserted before and after the selection. In the design and the output, the text content of the selection is retained as static text. In the schema tree (in the Schema Sources entry helper), such an XML node is indicated by parentheses containing an ellipsis.
- The inserted node tags are inserted with the necessary path (that is, with ancestor node tags that establish a path relative to the containing node). The path will be absolute or relative depending on the context of the node in the design.
- How to create nodes from table and list selections are described in Creating Tables and Lists as Elements/Attributes.

Adding and deleting nodes in the schema

You can add additional nodes (which are not based on an HTML selection) to the user-defined schema. Do this by right-clicking on a node and selecting the required command from the context menu. Alternatively, you can use the toolbar icons of the Schema Sources entry helper.

To delete a node, select the node and then use either the context menu or the toolbar icon. Note, however, that when a node is deleted, some paths in the design could be invalidated.

Modifying the design

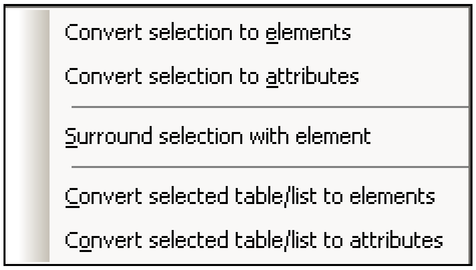
You can modify the structure of the design by dragging components around and by inserting static and dynamic components. Styles can also be modified using the various styling capabilities of StyleVision.

Creating Tables and Lists as Elements/Attributes

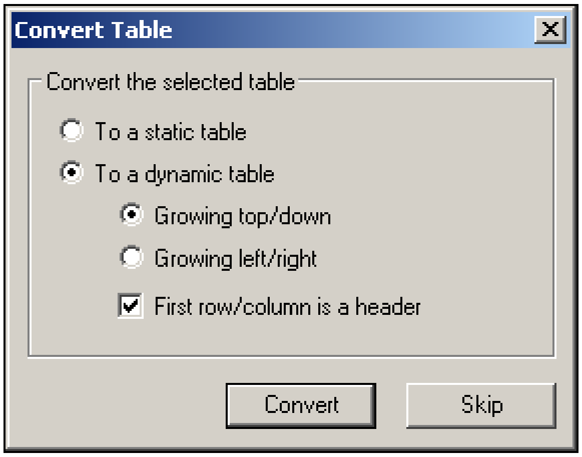
Tables and lists in the HTML document can be converted to element or attribute nodes in the XML Schema so that they retain the table or list structure in the schema.

Converting a table to elements/attributes
To convert a table to schema nodes, do the following:

1. Select the HTML table by highlighting some text in it.
2. Drag it to the node in the schema tree as a sibling or child of which you want to create it.
3. Drop the node when the Create as Sibling symbol ↓ or Create as Child symbol ⤵ appears.
4. In the context menu that now pops up (*screenshot below*), select the command **Convert selected table/list to elements** or **Convert selected table/list to attributes** according to whether you wish to create the contents of table cells as elements or attributes, respectively.


Convert selection to elements
Convert selection to attributes
Surround selection with element
Convert selected table/list to elements
Convert selected table/list to attributes

5. In the Convert Table dialog that pops up (*screenshot below*), select whether the table created in the SPS should be a static table or dynamic table.


Convert Table

Convert the selected table

○ To a static table
● To a dynamic table
● Growing top/down
○ Growing left/right
☑ First row/column is a header

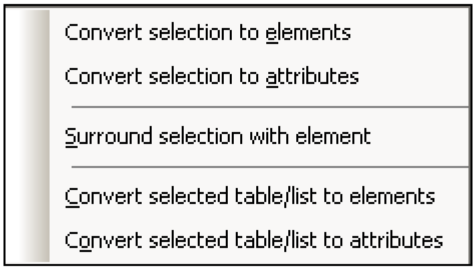
If the **static table** option is selected, then for each cell in the table, a schema node is created. In the design, each node is inserted with the `(content)` placeholder. The data in the table cells is copied to the temporary internal XML document (and to the generated XML document). The **dynamic table** option is available when the structure of all rows in the table are identical. When created in the SPS, the rows of the dynamic table are represented by a single row in the design (because each row has the same structure). The table data will be copied to the XML file. The dynamic table can grow top/down (rows are arranged vertically relative to each other) or left/right (rows become columns and extend from left to right). If you indicate that the first row/column is a header, then (i) a header row containing the column headers as static text is included in the design; and (ii) the schema element/attribute nodes take the header texts as their names. If the first row/column is not indicated as a header, then no header row is included in the design.

6. After you have selected the required option/s, click **Convert** to finish.

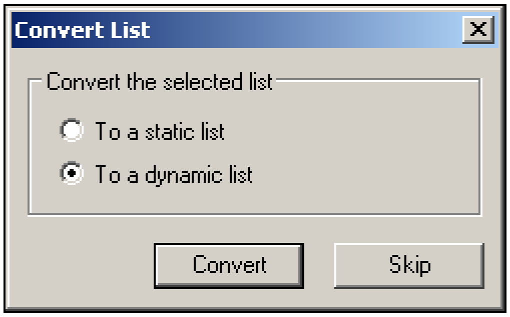
Converting a list to elements/attributes

To convert a list to schema nodes, do the following:

1. Select the HTML list by highlighting some text in it.
2. Drag it to the node in the schema tree as a sibling or child of which you want to create it.
3. Drop the node when the Create as Sibling symbol ↓ or Create as Child symbol ↴ appears.
4. In the context menu that now pops up (*screenshot below*), select the command **Convert selected table/list to elements** or **Convert selected table/list to attributes** according to whether you wish to create the contents of table cells as elements or attributes, respectively.


Convert selection to elements
Convert selection to attributes
Surround selection with element
Convert selected table/list to elements
Convert selected table/list to attributes

5. In the Convert List dialog that pops up (*screenshot below*), select whether the table created in the SPS should be a static table or dynamic table.


Convert List

Convert the selected list

○ To a static list

● To a dynamic list

If the **static list** option is selected, then for each list item, a schema node is created. In the design, each node is inserted with the text of the HTML list item included as static content of the list item. If the **dynamic list** option is selected, then each list item is represented by a single list item node in the design. In the design, the list item element is inserted with the `(content)` placeholder.

6. After you have selected the required option, click **Convert** to finish.

Generating Output

After completing the SPS, you can generate the following output using the **File | Save Generated Files** command:

- Generated user-defined schema, which is the schema you have created in the Schema Sources entry helper.
- Generated user-defined XML data, which is an XML document based on the schema you have created and containing data imported from the HTML file.
- XSLT stylesheets for HTML output.
- HTML output.

6.5 Command Line Interface: StyleVisionBatch

StyleVision's file-generation functionality can be called via the StyleVisionBatch utility, which is included in your StyleVision installation. The utility is named `StyleVisionBatch.exe` and is located in the StyleVision application folder. The syntax for invoking StyleVision commands via StyleVisionBatch is explained in the StyleVisionBatch Syntax sub-section. When a command is executed StyleVision runs silently (i.e. without the GUI being opened), generates the required output files, and closes.

Output files

Using StyleVisionBatch, you can generate one or more of the following files:

- XSLT-for-HTML (`.xslt`) file from the specified SPS
- HTML (`.html`) file using the XML and XSLT files in the specified SPS or using alternative XML and/or XSLT files
- XML Schema file of a database-based SPS
- XML data file of a database-based SPS

How to use the command line

There are two ways you can use the command line:

- Commands can be entered singly on the command line and be executed immediately. For example, in a DOS window you can go to the directory in which the StyleVisionBatch utility is, then enter a command such as: `StyleVisionBatch -v Test.sps -OutXSLT=Test.xslt`., and press **Enter** to execute the command.
- A series of commands can be entered in a **batch file** for batch processing. For example:

```
@ECHO OFF
CLS
StyleVisionBatch -v Test.sps -inpXSLT=EN.xslt -OutHTML=TestEN.html
StyleVisionBatch -v Test.sps -inpXSLT=DE.xslt -OutHTML=TestDE.html
StyleVisionBatch -v Test.sps -inpXSLT=FR.xslt -OutHTML=TestFR.html
```

  When the batch file is processed, the commands are executed and the files generated.

StyleVisionBatch Syntax

The syntax for the command line interface utility StyleVisionBatch is:

```
StyleVisionBatch [ <Stylevision exe>] [ <options>]
```

where

`StyleVisionBatch`	is the CLI utility, which is located in the StyleVision application folder
`<Stylevision exe>`	is the StyleVision executable file; it needs to be specified only if the StyleVision executable is **not** named `stylevision.exe` **or** is not located in the same folder as `StyleVisionBatch.exe`. If specified, the name must end in `.exe`.
`<options>`	One or more of the options listed below.

StyleVisionBatch options
StyleVisionBatch options may be entered in any order. In the listing below they are organized into groups so as to provide a better overview. Note that FO, RTF, and PDF output-related options are available in the Enterprise edition only; these options are indicated with the words *Enterprise edition* in the list below.

- ***Utility***

`-help` or `-?`	Displays syntax at the command line
`-verbose` or `-v`	Displays processing information at runtime
`-FOPBatFile=<file>`	Sets FOP processor batch file *(Enterprise edition)*

- ***SPS and Parameters***

`<stylesheet>`	Sets SPS (`.sps`) stylesheet
`$<paramname>=<value>`	Assigns a value to a parameter declared in the XSLT stylesheet. If the value contains a space, enclose the value in double quotes. For example: `$paramname="A value"`. Multiple parameters are separated by spaces.

- ***XSLT file output***

`-OutXSLT=<file>`	Writes XSLT-for-HTML to the specified file
`-OutXSLRTF=<file>`	Writes XSLT-for-RTF to the specified file *(Enterprise edition)*
`-OutXSLFO=<file>`	Writes XSLT-for-FO to the specified file *(Enterprise edition)*

- ***Input files***

`-InpXML=<file>`	Sets input XML file
`-InpXSLT=<file>`	Sets input XSLT-for-HTML file
`-InpXSLRTF=<file>`	Sets input XSLT-for-RTF file *(Enterprise edition)*

`-InpXSLFO=<file>`	Sets input XSLT-for-FO file *(Enterprise edition)*

- ***Output files***

`-OutHTML=<file>`	Writes HTML output to the specified file
`-OutRTF=<file>`	Writes RTF output to the specified file *(Enterprise edition)*
`-OutFO=<file>`	Writes FO output to the specified file *(Enterprise edition)*
`-OutPDF=<file>`	Writes PDF output to the specified file *(Enterprise edition)*

- ***DB data output***

`-OutDBXML=<file>`	Writes XML generated from DB to the specified file
`-OutDBSchema=<file>`	Writes XML Schema generated from DB to the specified file

- ***Additional flag for XML DBs***

`-DBCellWhere:<param>=<cond>`	Specifies the cell in the XML DB to output. The parameter `<param>` identifies the DB cell schema source and `<cond>` is a simple SQL `WHERE` clause that identifies the particular cell/s to be output. *Also see note below and the Examples section*.

Explanatory points
The following points provide supplementary information about StyleVisionBatch syntax and the command line process.

- When `StyleVisionBatch` is called, it looks in the current directory for `StyleVision.exe`. If your StyleVision executable is named otherwise or located in another folder, use the `<Stylevision_exe>` argument to specify the executable.
- Paths may be absolute or relative and should use backslashes.
- Options are prefixed either with a minus sign (for example: `-OutHTML`) or a forward slash (for example: `/OutHTML`).
- If the filename or the path to it contains a space, then the entire path should be enclosed in quotes. For example: `"c:\My Files\MyXML.xml"` or `"c:\MyFiles\My XML.xml"`.
- Commands, paths, and folder and file names are case-insensitive.
- If the SPS file is specified, the Working XML File associated with it and the XSLT stylesheet generated from it will be used to generate output; therefore no input XML or XSLT file is required. If, however, the SPS file is not specified, an input XML file and input XSLT file must be specified as options. An input XML File must also be specified if the SPS file does not have a Working XML File assigned to it. For output from a DB, the SPS must be specified.
- Parameter declarations refer to parameters in the XSLT stylesheet. Parameter names and values are case-sensitive. If the SPS is DB-based and has a DB Filter which uses parameters, the XML generated from the DB will be appropriately filtered, using parameter values you specify at the command line. Each parameter declaration on the

command line must be prefixed with a `$`, and, if multiple parameters are used, they must be separated from each other with a space. If the value of the parameter contains a space, then the value must be enclosed in double quotes.

- No default output is specified, so you must specify the required output. For example: `OutHTML=Test.html`.
- If you specify only the output file (no XML file or XSLT file), the Working XML File or DB specified in the SPS is used for the source XML, and the required XSLT is generated from the SPS.
- Any temporary files that are created are deleted at the end of the processing.
- The `-verbose` option provides a detailed report of all steps carried out during the processing of the command.
- When specifying HTML output, make sure that the generated file is placed in a location in which relative paths to images, etc, will point correctly to their targets. The same applies to hyperlinks.
- When the `-DBCellWhere` flag is used (`-DBCellWhere:[<param>]=<cond>`), the parameter `<param>` identifies the schema source of the DB cell. For example, if the parameter is `$DBXMLFIELD`, then the schema source is known to the SPS because the parameter in the SPS is keyed to a particular schema. The value of `<cond>` is an SQL `WHERE` clause which determines the cell/s to be used. It is typically of the form `"COLNAME = ROWNUM"`, where `COLNAME` is the name of the column containing the XML data cell and `ROWNUM` specifies the row/s in the column, and, by extension, the cell/s in the column. *See the examples in the next section*.
- When the `-DBCellWhere` flag is used and more than one row (or cell) is specified, then if the DB cell schema is the main schema in the SPS and the specified command requires input from the DBCell (these commands are: `-OutHTML`, `-OutRTF`, `-OutFO`, `-OutPDF`), then the command is executed once for each cell and each cell is output in a separate file; however, if the DB cell schema is a secondary schema in the SPS, the command is applied to only the first row (cell). *See the examples in the next section*.

StyleVisionBatch Examples

The examples below are organized according to output.

XSLT stylesheets
XSLT stylesheets can be generated from the SPS files. The only input required is the SPS file.

- The XSLT-for-HTML file is generated from the SPS.
```
StyleVisionBatch -v Test.sps -OutXSLT=Test.xslt
```

HTML output
HTML output is obtained by transforming an XML file with an XSLT stylesheet. The XML file may be the Working XML File assigned in the SPS or may be specified on the command line. The XSLT file may be that generated from the SPS or may be specified on the command line.

- Working XML file in SPS transformed with XSLT stylesheet generated from SPS.
```
StyleVisionBatch -v Test.sps -OutHTML=Test.html
```

- Specified XML file transformed with XSLT stylesheet generated from SPS.
```
StyleVisionBatch -v Test.sps -InpXML=External.xml -OutHTML=Test.html
```

- Working XML file in SPS transformed with specified XSLT stylesheet.
```
StyleVisionBatch -v Test.sps -InpXSLT=External.xslt -OutHTML=Test.html
```

- Specified XML file transformed with specified XSLT stylesheet.
```
StyleVisionBatch -v -InpXML=External.xml  -InpXSLT=External.xslt
-OutHTML=Test.html
StyleVisionBatch -v Test.sps -InpXML=External.xml
-InpXSLT=External.xslt -OutHTML=Test.html
```

DB data
XML, XML Schema, XSLT, and output files can be generated from a DB-based SPS. The appropriate switches must be set for the respective outputs.

- XML Schema file from DB-based SPS.
```
StyleVisionBatch -v DB.sps -OutDBSchema=DB.xsd
```

- XML data file from DB-based SPS.
```
StyleVisionBatch -v DB.sps -OutDBXML=DB.xml
```

- XSLT-for-HTML file from DB-based SPS.
```
StyleVisionBatch -v DB.sps -OutXSLT=DB.xslt
```

- HTML file from DB-based SPS.
```
StyleVisionBatch -v DB.sps -OutHTML=DB.html
```

- Combinations of DB data output on a single command line.
```
StyleVisionBatch -v DB.sps -OutDBSchema=DB.xsd -OutXSLT=DB.xslt
StyleVisionBatch -v DB.sps -OutDBXML=DB.xml -OutXSLT=DB.xslt
-OutHTML=DB.html
```

In the case of XML DBs, two situations should be distinguished: when the schema source of the DB cell is: (i) the main schema of the SPS, or (ii) a secondary schema of the SPS. Consider the following example command:

```
StyleVisionBatch DBCellTest.sps -OutHTML=Out.html
-DBCellWhere:$DBXMLFIELD="NHE_TEST_PK < 4"
```

- The parameter `$DBXMLFIELD` identifies the source schema and also the table and column containing the XML data cell since this information is implicit in the DB cell schema.
- The SQL `WHERE` clause `"NHE_TEST_PK < 4"` specifies the first three rows of the column `NHE_TEST_PK`.
- If the schema identified by `$DBXMLFIELD` is the main schema of `DBCellTest.sps`, then the `-outHTML` command is processed thrice, once for each of the first three rows (cells), and three output files are created, named, respectively, `Out.html`, `Out(2).html`, and `Out(3).html`. If on the other hand the schema identified by `$DBXMLFIELD` is a secondary schema of `DBCellTest.sps`, then only the first row (cell) is output to the file `Out.html`.

Parameter Usage
For the XSLT transformation, parameters can be passed to the XSLT stylesheet from the command line.

- Parameters passed to XSLT stylesheet generated from the SPS.

```
StyleVisionBatch -v Test.sps -OutHTML=Test.html $myparam=MyText
StyleVisionBatch -v Test.sps -inpXML=External.xml -OutHTML=Test.html
$myparam="My Text"
StyleVisionBatch -v Test.sps -OutHTML=Test.html -OutFO=Test.fo
$myparam="MyText"
StyleVisionBatch -v Test.sps -OutHTML=Test.html $myparam=2006
StyleVisionBatch -v Test.sps -OutHTML=Test.html $myparam="2006"
```

- Parameters passed to specified XSLT stylesheet.

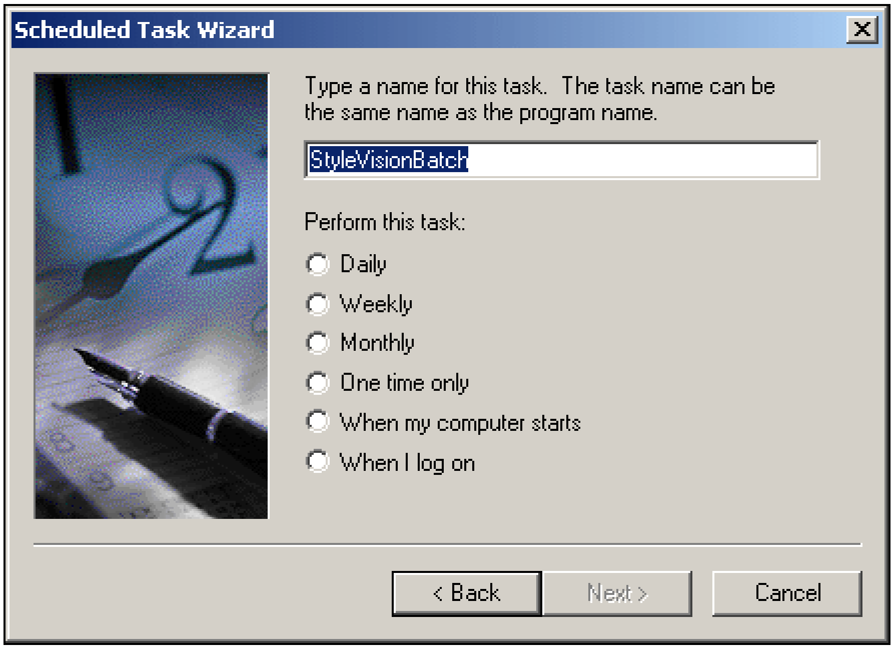
```
StyleVisionBatch -v Test.sps -inpXSLT=External.xslt -OutHTML=Test.html
$myparam=MyText
StyleVisionBatch -v Test.sps -inpXSLT=External.xslt -OutHTML=Test.html
$myparam="My Text"
```

Scheduling StyleVisionBatch Tasks

A StyleVisionBatch command or set of commands (described in the previous sections) can be set up to run to a pre-determined schedule. This scheduling is done with the Scheduled Tasks tool of Windows. The Scheduled Task tool opens the StyleVisionBatch utility and executes the command specified in the task.

To create a scheduled task, do the following:

1. If you plan to run a set of StyleVisionBatch commands as a scheduled task—as opposed to a single command—these commands should be created in a batch file (*see* Command Line Interface: StyleVisionBatch). If a single StyleVisionBatch command is to be scheduled, go to Step 2.
2. Open the Scheduled Task Wizard of Windows (**Start | Control Panel | Scheduled Tasks | Add Scheduled Task**).
3. Click **Next** to start setting up the task.
4. In the window to select the program to run, you select either `StyleVisionBatch.exe` (for a single StyleVisionBatch command) or a batch file (containing multiple StyleVisionBatch commands). Browse for the required file and select it. The next screen (*screenshot below*) appears.



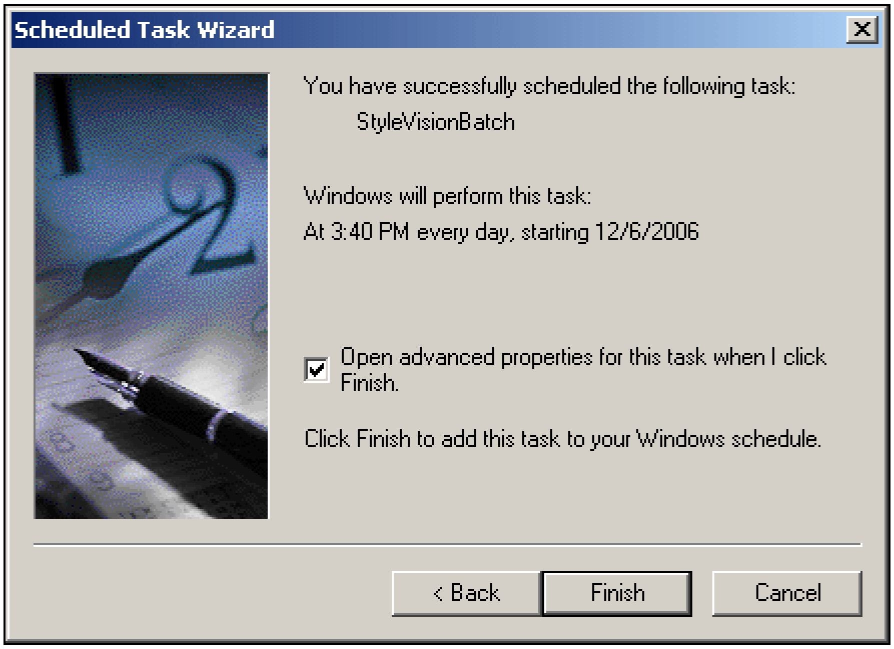
Type a name for this task. The task name can be the same name as the program name.

StyleVisionBatch

Perform this task:

○ Daily
○ Weekly
○ Monthly
○ One time only
○ When my computer starts
○ When I log on

5. Assign a name for the task, and set a frequency for it. Then click **Next**.
6. Select the starting day and time for the schedule. Then click **Next**.
7. Enter the appropriate user name and password. Then click **Next**.
8. In the finishing screen (*screenshot below*), if you are scheduling a single StyleVisionBatch command and have therefore selected `SVBATCH%>.exe` as the program to run, check the Open Advanced Properties... check box. (It is in the Advanced Properties dialog that the StyleVisionBatch command is specified.) Then click **Finish**.



You have successfully scheduled the following task:
StyleVisionBatch

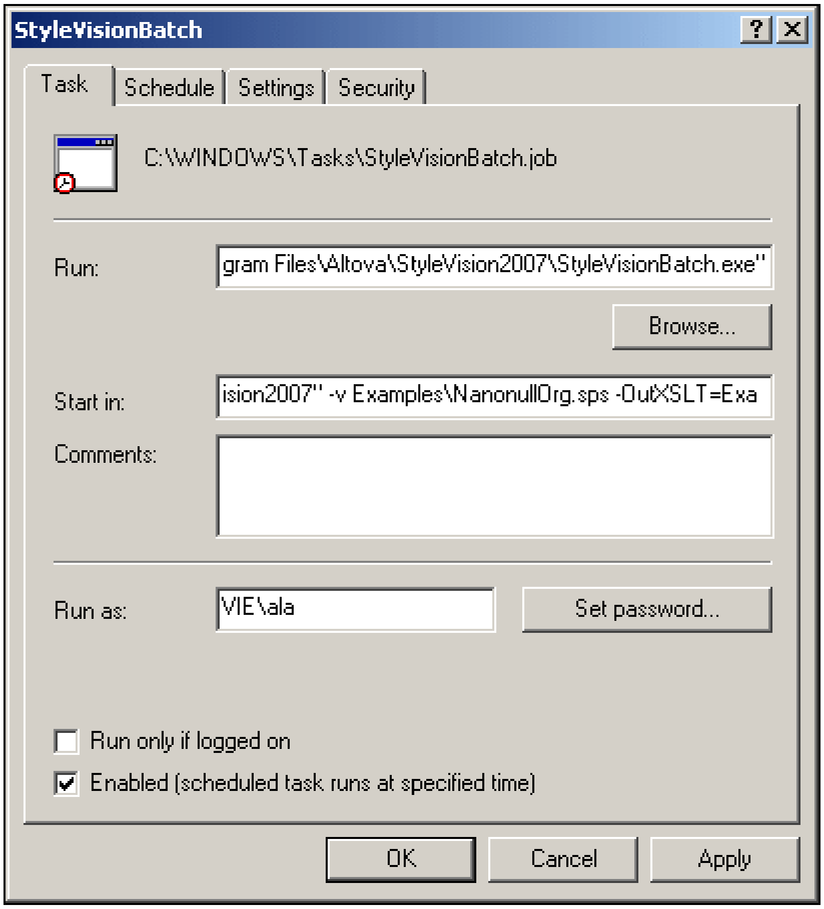
Windows will perform this task:
At 3:40 PM every day, starting 12/6/2006

☑ Open advanced properties for this task when I click Finish.

Click Finish to add this task to your Windows schedule.

If you have specified a batch file as the program to run for the task, there is no need to set any advanced properties and you can leave the Open Advanced Properties check box unchecked. In this case, the scheduling of the task is now complete.

9. This step is required only if you are scheduling a single StyleVisionBatch command as your task. On clicking **Finish** with the Open Advanced Properties... check box checked, a dialog showing the properties of the task pops up (*screenshot below*).



C:\WINDOWS\Tasks\StyleVisionBatch.job

Run: gram Files\Altova\StyleVision2007\StyleVisionBatch.exe"

Start in: ision2007" -v Examples\NanonullOrg.sps -OutXSLT=Exa

Comments:

Run as: VIE\ala

□ Run only if logged on

☑ Enabled (scheduled task runs at specified time)

In the Start In text field (*screenshot above*) enter the required StyleVisionBatch command, for example: `"C:\Program Files\Altova\StyleVision2007" -v Examples\NanonullOrg.sps -OutXSLT=Examples\Nano1.xslt`. Use quotes if there are spaces in your file or folder names, and, in your paths, use backslashes. If desired, enter a comment describing the task. Click **OK** to finish.

Chapter 7

Content Editing Procedures

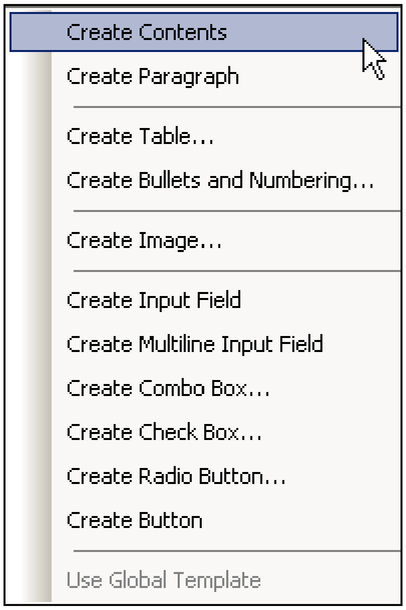
7 Content Editing Procedures

This section describes in detail the core procedures used to create and edit the components of an SPS template. The procedures are listed below and described in detail in the sub-sections of this section. These mechanisms are used to design any kind of template: main, global, or named.

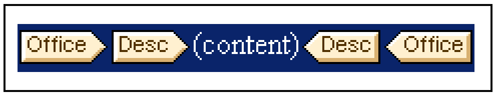
- Inserting XML Content as Text. XML data can be inserted in the design by dragging the relevant nodes into the design and creating them as `(contents)` and `(rest-of-contents)`.
- Sorting. A set of XML elements can be sorted on multiple sort-keys.
- Using Data-Entry Devices. XML data can be input by the Authentic View user via data-entry devices such as input fields and combo boxes. This provides a layer of user help as well as of input constraints. Individual nodes in the XML document can be created as data-entry devices.
- Creating Lists. Static lists, where the list structure is entered in the SPS design, and dynamic lists, where an XML document sub-structure is created as a list, provide powerful data-ordering capabilities.
- Working with Tables. Tables can be inserted by (i) the SPS designer, directly in the SPS design (static tables) or using XML document sub-structures, and (ii) the Authentic View user.
- Using Graphics: Graphics can be inserted in the SPS design using a variety of methods to determine the target URI (static, dynamic, a combination of both, and unparsed entity URIs).
- Bookmarks and Hyperlinks. Bookmarks mark key points in the output document, which can then be targeted by hyperlinks. Hyperlinks can also link to external resources using a variety of methods to determine the target URI (static, dynamic, a combination of both, and unparsed entity URIs).
- Auto-Calculations. Auto-Calculations are a powerful XPath-based mechanism to manipulate data and (i) present the manipulated data in the output as well as (ii) update nodes in the XML document with the result of the Auto-Calculation.
- Conditional Templates. A parts of the design document can be active when a certain condition pertaining to the XML document is true. In this way, the output can be made conditional on the data held in the XML document.

7.1 Inserting XML Content as Text

Data from a node in the XML document is included in the design by dragging the corresponding schema node from the Schema Sources window and dropping it into the design. When the schema node is dropped into the design, a menu pops up with options for how the node is to be created in the design (*screenshot below*).



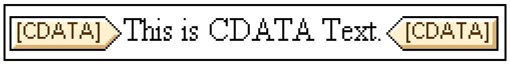
To output the text contents of the node, the node should be created as contents. When a node is created as contents, the node will look something like this in the design document:



In the screenshot above, the `Desc` element has been created as contents. The output will display the text content of `Desc`. If `Desc` has descendant elements, such as `Bold` and `Italic`, then the text content of the descendant elements will also be output as part of the contents of `Desc`. Note that attribute nodes of `Desc` are not considered its child nodes, and the contents of the attribute nodes will therefore not be output as part of the contents of `Desc`. Attribute nodes have to be explicitly inserted in order to be processed.

CDATA sections

If CDATA sections are present in the XML document they will be output, and in Authentic View, are indicated with tags when markup is switched on (using the menu command **Authentic | Markup**). CDATA sections can also be inserted in the XML document when editing the document in Authentic View (via the context menu).



In the sub-sections of this section, we describe other aspects of inserting XML content as text:

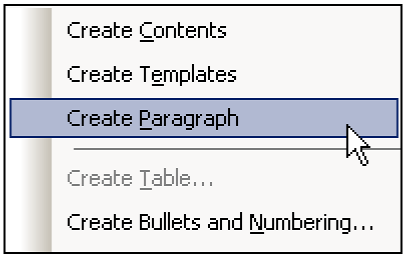
- How the text content of a node can be marked up with a predefined format directly when the node is inserted.

- How the structure of the source schema determines the effect of Authentic View usage.
- How descendant nodes not explicitly included within a node can be included for processing. See Rest-of-Contents.

Note: You can create an **empty template rule** by deleting the `(content)` placeholder of a node. An empty template rule is useful if you wish to define that some node not be processed, i.e. produce no output.

Inserting Content with a Predefined Format

The text content of a node can be directly inserted with the markup of one of StyleVision's predefined formats. To do this, drag the node from the Schema Sources window and drop it at the desired location. In the menu that pops up, select **Create Paragraph** (*screenshot below*).



The predefined format can be changed by selecting the predefined format tag and then choosing some other predefined format from the Format combo box in the toolbar or using the menu command **Insert | Format**. The predefined format can also be changed by changing the value of the `paragraph type` property of the *paragraph* group of properties in the Properties window, or by changing the paragraph type in the context menu.

Each paragraph type has particular formatting features that can be used to advantage. Note that the `pre` format type enables carriage returns and linefeeds to be output as such instead of them being normalized to whitespace.

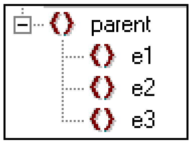
Adding Elements in Authentic View

When creating elements in the design, the way you create the elements determines how Authentic View will respond to user actions like pressing the Tab key and clicking the `Add. . .` prompt. The basic issue is what elements are created in Authentic View when an element is added by the user. For example, when the user adds an element (say, by clicking the **Insert Element** icon in the Elements Entry Helper), what child elements are created automatically?

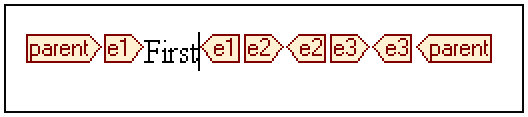
The most important point to bear in mind is that Authentic View follows the structure specified in the underlying schema. In order to ensure that Authentic View implements the schema structure correctly there are a few design rules you should keep in mind. These are explained below.

Unambiguous content model

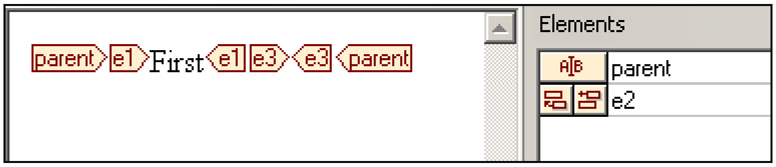
A content model is considered unambiguous when it consists of a single sequence (with `maxOccurs=1`) of child elements (with no choices, groups, substitutions, etc). In such cases, when the element is added, the sequence of child elements is unambiguously known, and they are automatically added. In the screenshot example below, the three child elements are all mandatory and can occur only once.



When the element `parent` is added in Authentic View, its child elements are automatically inserted (*screenshot below*). Pressing the tab key takes you to the next element in the sequence.



If the `e2` element were optional, then, when the element `parent` is added in Authentic View, the elements `e1` and `e3` are automatically inserted, and the element `e2` appears in the Elements Entry Helper so that it can be inserted if desired (*screenshot below*). Pressing the tab key in `e1` takes the user to `e3`.



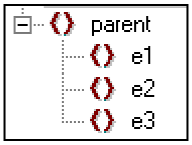
The above content model scenario is the only scenario Authentic View considers unambiguous. All other cases are considered ambiguous, and in order for Authentic View to disambiguate and efficiently display the desired elements the design must adhere to a few simple rules. These are explained below.

Ambiguous content model
For Authentic View to correctly and efficiently display elements correctly while an XML document is being edited, the SPS must adhere to the following rules.

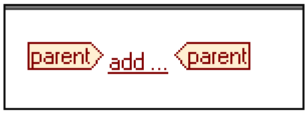
- Child elements will be displayed in the order in which they are laid out in the design.
- In order for Authentic View to disambiguate among sibling child elements, all child elements should be laid out in the design document in the required order **and within a single parent node**. If the sibling relationship is to be maintained in Authentic View, it is incorrect usage to lay out each child element of a single parent inside multiple instances of the parent node.

These two rules are illustrated with the following example.

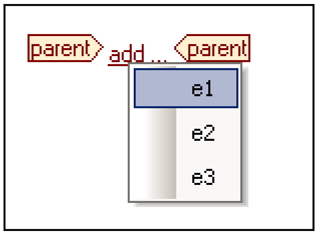
We consider a content model of an element `parent`, which consists of a single sequence of mandatory child elements. This content model is similar to the unambiguous content model discussed above, with one difference: the single sequence is optional, which makes the content model ambiguous—because the presence of the sequence is not a certainty. If you create a design document as shown in the screenshot below, there will be ambiguity in Authentic View.



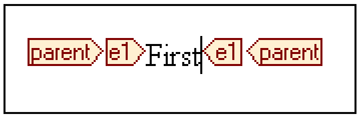
The Authentic View of the `parent` element will look like this (since the sequence is optional):



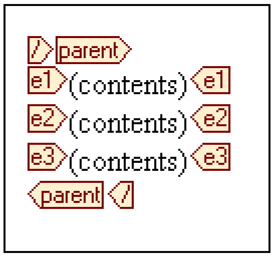
Clicking `add...` pops up a menu of the three child elements:



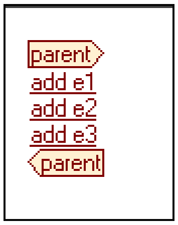
If you select one of these elements, it will be inserted (*screenshot below*), but since Authentic View cannot disambiguate the sequence it does not insert any of the remaining two elements, nor does it offer you the opportunity of inserting them:



The **correct** way to design this content model (following the rules given above) would be to explicitly create the required nodes in the desired order within the single parent node. The design document would look like this:



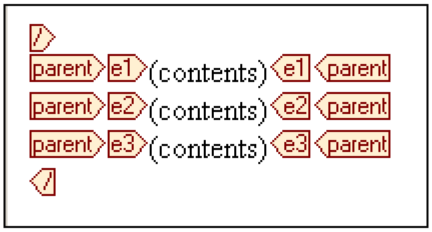
Note that all three child elements are placed **inside a single parent node**. The design shown above would produce the following Authentic View:



The Authentic View user clicks the respective `add element` prompt to insert the element and its content.

Note:

- If an element can occur multiple times, and if the rules above are followed, then the element appears in the Entry Helper till the number of occurrences in Authentic View equals the maximum number of occurrences allowed by the schema (`maxOccurs`).
- Creating each child element inside a separate parent node (*see screenshot below*) not only creates isolated child–parent relationships for each child element so instantiated; it also increases processing time because the parent node has to be re-traversed in order to locate each child element.


/
parent e1 (contents) e1 parent
parent e2 (contents) e2 parent
parent e3 (contents) e3 parent
/

Rest-of-Contents

The `rest-of-contents` placeholder applies templates to all the remaining child elements of the element for which the template has been created. As an example consider the following:

- An element `parent` has 4 child elements, `child1` to `child4`.
- In the template for element `parent`, some processing has been explicitly defined for the `child1` and `child4` child elements.

This results in only the `child1` and `child4` child elements being processed. The elements `child2` and `child3` will not be processed. Now, if the `rest-of-contents` placeholder is inserted within the template for `parent`, then, not only will `child1` and `child4` be processed using the explicitly defined processing rules in the template. Additionally, templates will be applied for the `child2` and `child3` child elements. If global templates for these are defined then the global templates will be used. Otherwise the built-in default templates (for element, attribute, and text nodes) will be applied.

Important: It is important to note what nodes are selected for `rest-of-contents`.

- As described with the example above, all child element nodes and child text nodes are selected by the `rest-of-contents` placeholder.
- Attribute nodes are not selected; they are not child nodes, that is, they are not on the child axis of XPath.
- If a global template of a child element is used in the parent template, then the child element does not count as having been used locally. As a result, the `rest-of-contents` placeholder will also select such child elements. However, if a global template of a child element is "copied locally", then this usage counts as local usage, and the child element will not be selected by the `rest-of-contents` placeholder.

Note: You can create an **empty template rule** by deleting the `(content)` placeholder of a node. An empty template rule is useful if you wish to define that some node not be processed, i.e. produce no output.

7.2 Sorting

A set of sibling element nodes of the same qualified name can be sorted on one or more sort-keys you select. For example, all the `Person` elements (within, say, a `Company` element) can be sorted on the `LastName` child element of the `Person` element. The sort-key must be a node in the document, and is typically a descendant node (element or attribute) of the element node being sorted. In the example mentioned, `LastName` is the sort-key.

If there are two elements in the set submitted for sorting that have sort-key nodes with the same value, then an additional sort-key could provide further sorting. In the `Person` example just cited, in addition to a first sort-key of `LastName`, a second sort-key of `FirstName` could be specified. So, for `Person` elements with the same `LastName` value, an additional sort could be done on `FirstName`. In this way, in an SPS, multiple sort instructions (each using one sort-key) can be defined for a single sort action.

The template is applied to the sorted set and the results are sent to the output in the sorted order. Sorting is supported in the HTML output.

Note: Sorting is not supported for Authentic View output.

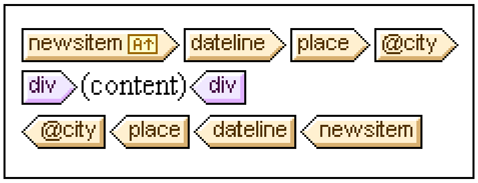
In this section

- The sorting mechanism is described.
- An example demonstrates how sorting is used.

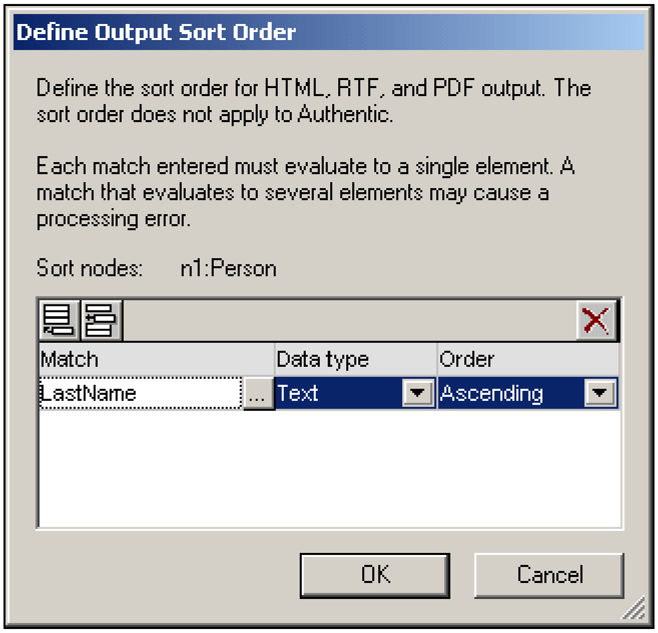
The Sorting Mechanism

Setting up a schema element node for sorting consists of two steps:

1. In Design View, select the schema element node that is to be sorted. Note that it is the instances of **this** element in the XML document that will be sorted. Often it might not immediately be apparent which element is to be sorted. For example, consider the structure shown in the screenshot below.

   Each `newsitem` has a `dateline` containing a `place` element with a `city` attribute. The `@city` nodes of all `newsitem` elements are to be output in alphabetical order. In the design, should the `@city` node be selected for sorting, or the `place`, `dateline`, or `newsitem` elements? With `@city` selected, there will be only the one `city` node that will be sorted. With `place` or `dateline` selected, again there will be just the one respective element to sort, since within their parents they occur singly. With `newsitem` selected, however, there will be multiple `newsitem` elements within the parent `newsitems` element. In this case, it is the `newsitem` element that should be sorted, using a sort-key of `dateline/place/@city`.
2. After selecting the element to sort, in the context menu (obtained by right-clicking the element selection), click the **Sort Output** command. This pops up the Define Output Sort Order dialog (*screenshot below*), in which you insert or append one or more sort instructions.

   Define Output Sort Order

   Define the sort order for HTML, RTF, and PDF output. The sort order does not apply to Authentic.

   Each match entered must evaluate to a single element. A match that evaluates to several elements may cause a processing error.

   Sort nodes: n1:Person

Match	Data type	Order
LastName	Text	Ascending

   Each sort instruction contains: (i) a sort-key (entered in the Match column); (ii) the datatype that the sort-key node should be considered to be (text or number); (iii) and the order of the sorting (ascending or descending). The order in which the sort instructions are listed is significant. Sorting is carried out using each sort instruction in turn, starting with the first, and working down the list when multiple items have the same value. Any number of sort instructions are allowed.

For an example of how sorting is used, see Example: Sorting on Multiple Sort-Keys.

A note about sort-keys

In both XSLT 1.0 and XSLT 2.0 SPSs, the XPath expression you enter for the sort-key must select a **single node** for each element instance—not a nodeset (XPath 1.0) or a sequence of items (XPath 2.0); the key for each element should be resolvable to a string or number value.

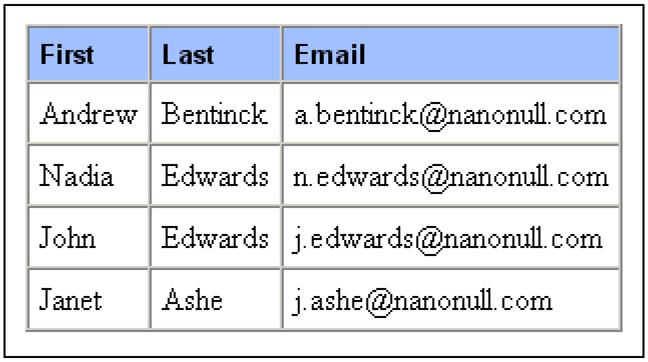
In an **XSLT 2.0** SPS, if multiple nodes are selected for the sort-key, an XSLT processing error will be returned. So, in the Person example cited above, with a context node of `Person`, an XPath expression such as: `../Person/LastName` would return an error because this expression returns all the `LastName` elements contained in the parent of `Person` (assuming there is more than one `Person` element). The correct XPath expression, with `Person` as the context node, would be: `LastName` (since there is only one `LastName` node for each `Person` element).

In **XSLT 1.0**, the specification requires that when a nodeset is returned by the sort-key selector, the text value of the first node is used. StyleVision therefore returns no error if the XPath expression selects multiple nodes for the sort-key; the text of the first node is used and the other nodes are ignored. However, the first node selected might not be the desired sort-key. For example, the XPath expression `../Person/LastName` of the example described above would not return an error. But neither would it sort, because it is the same value for each element in the entire sort loop (the text value of the first `LastName` node). An expression of the kind: `location/@*`, however, would sort, using the first attribute of the `location` child element as the sort-key. This kind of expression, however, is to be avoided, and a more precise selection of the sort-key (selecting a single node) is advised.

Example: Sorting on Multiple Sort-Keys

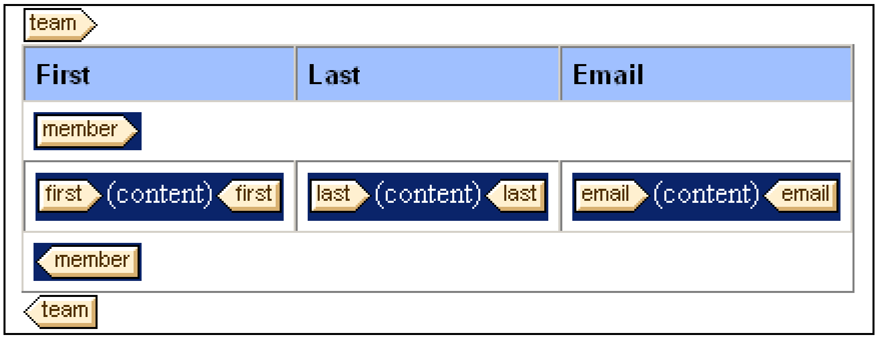
In the simple example below (available in the application folder `StyleVision2007/Examples/Tutorials/Sorting/SortingOnTwoTextKeys.sps`), team-members are listed in a table. Each member is listed with first name, last name, and email address in a row of the table. Let us say we wish to sort the list of members alphabetically, first on last name and then on first name. This is how one does it.

When the list is unsorted, the output order is the order in which the `member` elements are listed in the XML document (*screenshot below, which is the HTML output*).

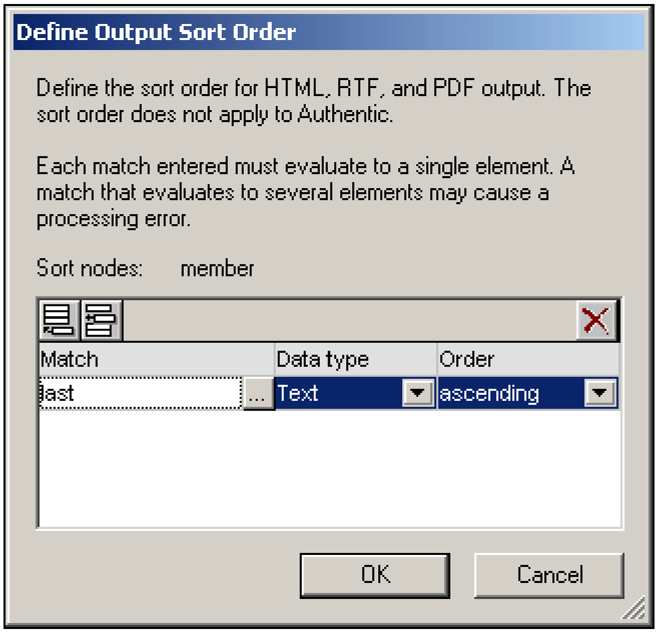


First	Last	Email
Andrew	Bentinck	a.bentinck@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com
John	Edwards	j.edwards@nanonull.com
Janet	Ashe	j.ashe@nanonull.com

In Design View, right-click the `member` element (*highlighted in screenshot below*), and from the context menu that appears, select the **Sort Output** command.



This pops up the Define Output Sort Order dialog (*screenshot below*). Notice that the element selected for sorting, `members`, is named at the Sort Nodes entry. This node is also the context node for XPath expressions to select the sort-key. Click the Add Row button (at left of pane toolbar) to add the first sort instruction. In the row that is added, enter an XPath expression in the Match column to select the node `last`. Alternatively, click the Build button ... to build the XPath expression. The Datatype column enables you to select how the sort-key content is to be evaluated: as text or as a number. The Order column lists the order of the sort: ascending or descending. Select `Text` and `Ascending`. Click **OK** to finish.



Define Output Sort Order

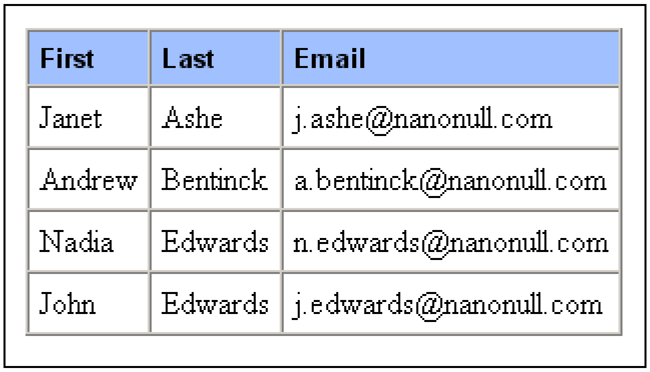
Define the sort order for HTML, RTF, and PDF output. The sort order does not apply to Authentic.

Each match entered must evaluate to a single element. A match that evaluates to several elements may cause a processing error.

Sort nodes: member

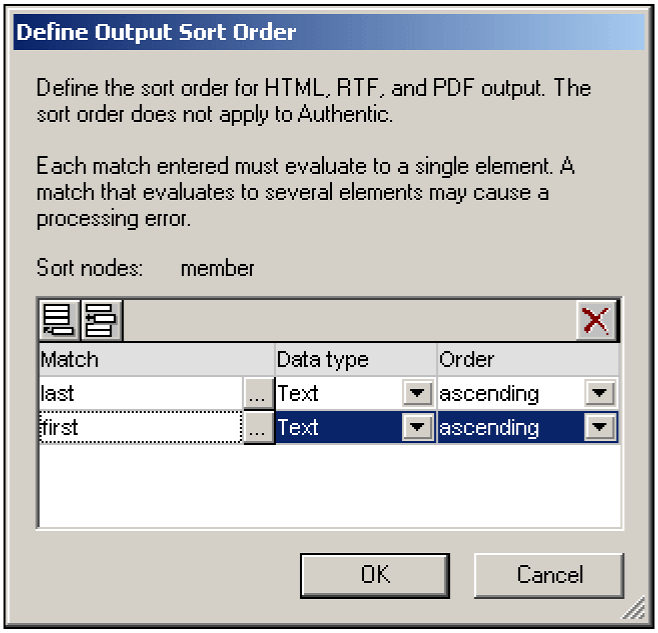
Match	Data type	Order
last	Text	ascending

In Design View, the `member` tag displays an icon indicating that it contains a sort filter member A↑. The HTML output of the team-member list, sorted on last name, is shown below. Notice that the two Edwards are not alphabetically sorted (Nadia is listed before John, which is the order in the XML document). A second sort-key is required to sort on first name.



First	Last	Email
Janet	Ashe	j.ashe@nanonull.com
Andrew	Bentinck	a.bentinck@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com
John	Edwards	j.edwards@nanonull.com

In Design View, right-click the `member` tag and select the **Sort Output** command from the context menu. The Define Output Sort Order dialog pops up with the `last` sort instruction listed. To add another sort instruction, append a new row and enter the `first` element as its sort-key (*screenshot below*). Click **OK** to finish.



Define Output Sort Order

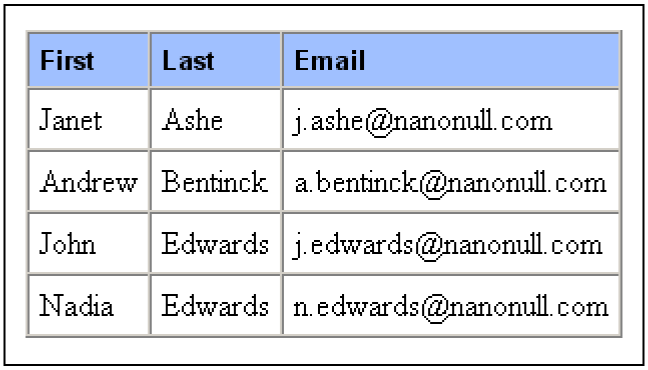
Define the sort order for HTML, RTF, and PDF output. The sort order does not apply to Authentic.

Each match entered must evaluate to a single element. A match that evaluates to several elements may cause a processing error.

Sort nodes: member

Match	Data type	Order
last	Text	ascending
first	Text	ascending

In the HTML output, the list is now sorted alphabetically on last name and then first name.



First	Last	Email
Janet	Ashe	j.ashe@nanonull.com
Andrew	Bentinck	a.bentinck@nanonull.com
John	Edwards	j.edwards@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com

7.3 Using Data-Entry Devices

Nodes in the XML document can be created as data-entry devices (such as input fields and combo boxes). Data-entry devices are intended for easier editing in Authentic View. For example, an input field makes it clear to the Authentic View user that input is expected in this location while a combo box lists, as well as restricts, the values a user can enter. When data is entered into a data-entry device, the data is inserted into the XML document as element content or as an attribute's value. In the HTML output, the data-entry device is rendered as an object that is the same as that displayed in Authentic View, or a near-equivalent. Note that data-entry devices accept input to the XML document and, therefore, will not work in the HTML output.

General mechanism
Given below is a list of the data-entry devices available in StyleVision, together with an explanation of how data is entered in the XML document for each device.

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

The text values entered in the input fields are entered directly into the XML document as XML content. For the other data-entry devices, the Authentic View user's selection is mapped to a value. StyleVision enables you to define the list of options the user will see and the XML value to which each option is mapped. Typically, you will define the options and their corresponding values in a dialog.

General usage
To create a data-entry device, do the following:

1. Drag a node from the Schema Sources window into Design View and drop it at the desired location.
2. From the context menu that appears, select the data-entry device you wish to create the node as.
3. For some data-entry devices, a dialog pops up. In these cases, enter the required information in the dialog, and click OK.

To **reopen and edit** the properties of a data-entry device, select the data-entry device (not the node containing it), and edit its properties in the Properties entry helper.

Note:

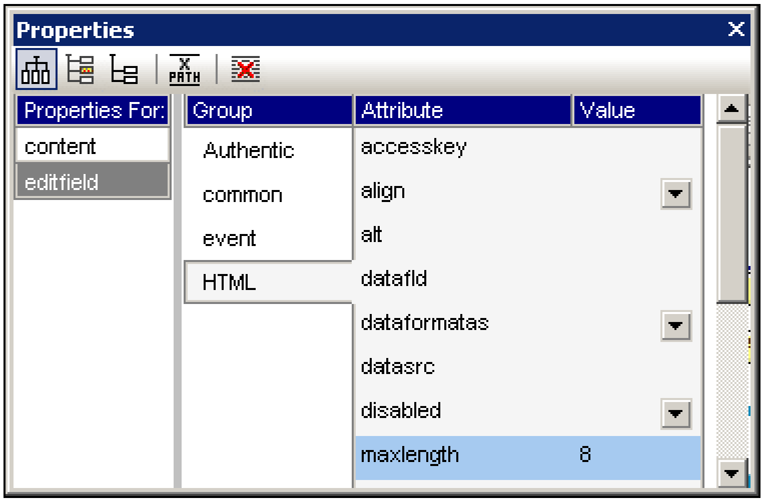
- Data can be entered in data-entry devices only in Authentic View.
- Data-entry devices can also be created by changing the current component type of a node to a data-entry device. To do this right-click the node and select **Change to**.
- In the HTML output, the entry selected by the user is displayed in the output. Changing the value of a data-entry device in the HTML document does not change the text value in either the XML document or HTML document.

Input Fields, Multiline Input Fields

You can insert an Input Field or a Multiline Input Field in your SPS when you drop a node from the Schema Sources window into Design View. The text that the Authentic View user enters into these fields is entered into the XML node for which the field was created. The content of that node is displayed in the input field or multiline input field.

Editing the properties of input fields

You can modify the HTML properties of input fields by selecting the input field and then modifying its HTML properties in the Properties entry helper (*see screenshot below*).



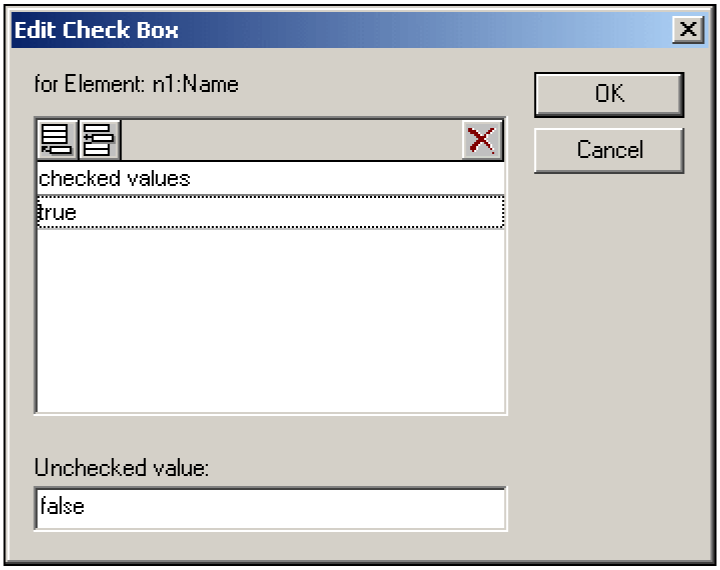
Properties

Properties For:	Group	Attribute	Value
content	Authentic	accesskey	
editfield	common	align	
	event	alt	
	HTML	datafld	
		dataformatas	
		datasrc	
		disabled	
		maxlength	8

For example, with the input field selected, in the Properties window select `editfield`, select the `HTML` group of properties and the `maxlength` property. Then double-click in the Value field of `maxlength` and enter a value.

Check Boxes

You can create a check box as a data-entry device. This enables you to constrain user input to one of two choices. In the Edit Check Box dialog (*shown below*), you specify the XML values to map to the checked and unchecked events.



checked values
true

Unchecked value:
false

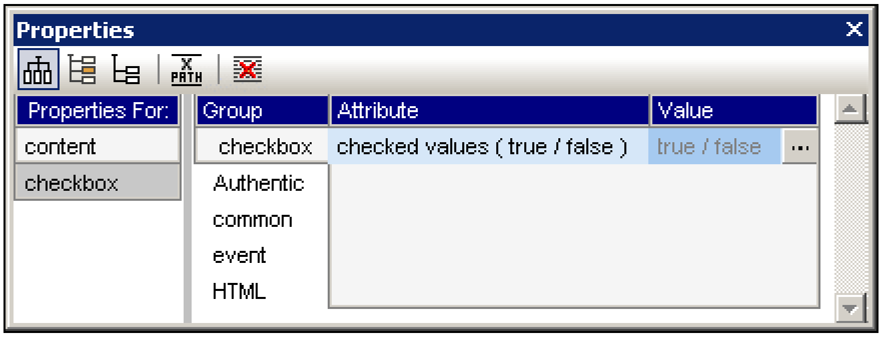
In the above screenshot, an element called `Name` has been created as a check box. If the Authentic View user checks the check box, a value of `true` will be entered as the value of the element `Name`. If the value is unchecked, then the value `false` is entered as the XML value of `Name` (as defined in the dialog).

Note: When a new `Name` (or check box) element is created in Authentic View, its XML value is empty (it is not the Unchecked Value). The Unchecked Value is entered only after the check box has first been checked, and then unchecked. To have a default value in a node, create a Template XML file that contains the default value.

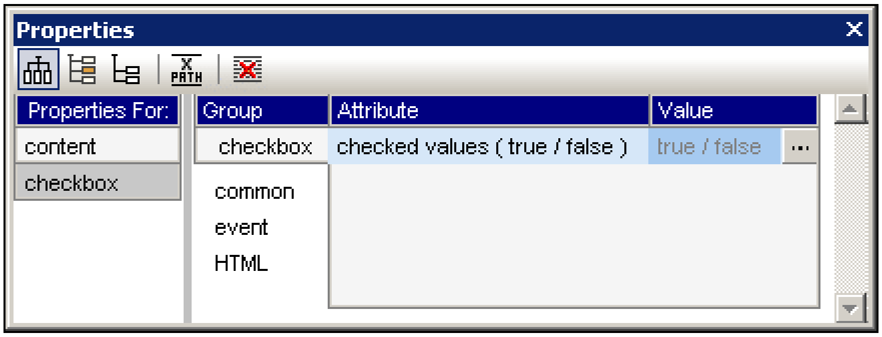
Accessing the Edit Check Box dialog

If you are creating a new check box, when you create the node as a check box, the Edit Check Box dialog pops up. To access the Edit Check Box dialog afterwards, do the following:

1. Select the check box in the design.
2. In the Properties entry helper, select the checkbox item and then the *checkbox* group of properties (*see screenshot below*).



Properties For:	Group	Attribute	Value
content	checkbox	checked values (true / false)	true / false
checkbox	Authentic		
	common		
	event		
	HTML		



Properties For:	Group	Attribute	Value
content	checkbox	checked values (true / false)	true / false ...
checkbox	common		
	event		
	HTML		

3. Click the Edit button ... of the `check values` property. This pops up the Edit Check Box dialog.

Note: You can modify the HTML properties of a check box by selecting it and then modifying its HTML properties in the Properties entry helper.

Combo Boxes

A combo box presents the Authentic View user with a list of options entries in a dropdown list. The selected option is mapped to a value that is entered in the XML document. The mapping of drop-down list entry to XML value is specified in the SPS.

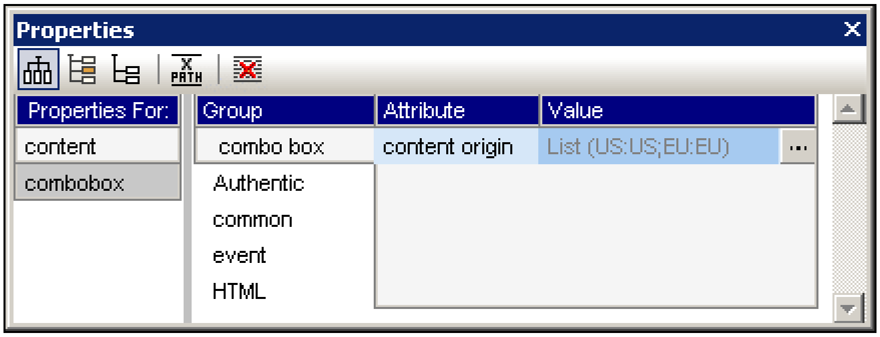
Mappings can be made in the Edit Combo Box dialog in one of three ways:

- From the schema enumerations for the selected node. In this case, the visible entry (in the dropdown list) will be the same as the XML value.
- From a list defined in the Edit Combo Box dialog. You enter the visible entry and the corresponding XML value, which may be different.
- From the result sequence of an XPath expression relative to the current node. The items in the result sequence are displayed as the entries of the drop-down list, and the list entry selected by the Authentic View user is entered as the value of the node. This is a powerful method of using dynamic entries in the combo box. The node that you create as the combo box is important. For example, say you have a `NameList` element that may contain an unlimited number of `Name` elements, which themselves have `First` and `Last` children elements. If you create the `Name` element as a combo box, and select the `Last` child element for the list values, then, in Authentic View, you will get as many combo boxes as there are `Name` elements and each combo box will have the `Last` child as its dropdown menu entry. In order to get a single combo box with all the `Last` elements in the dropdown menu list, you must create the single `NameList` element as the combo box, and select the `Last` element in the XPath expression.

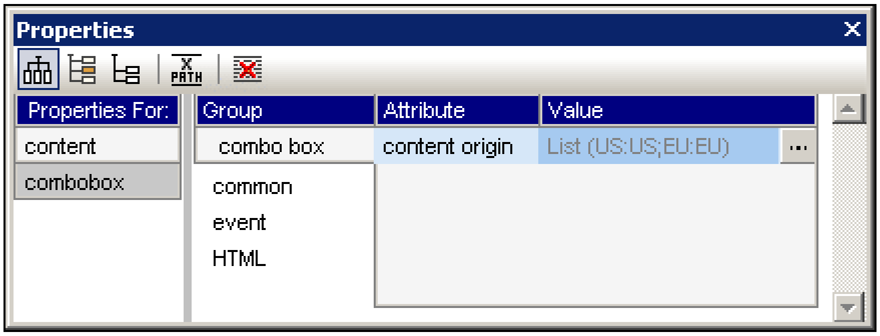
Accessing the Edit Combo Box dialog

If you are creating a new combo box, when you create the node as a combo box, the Edit Combo Box dialog pops up. To access the Edit Combo Box dialog afterwards, do the following:

1. Select the combo box in the design.
2. In the Properties entry helper, select the combobox item and then the *combo box* group of properties (*see screenshot below*).



content origin: List (US:US;EU:EU)

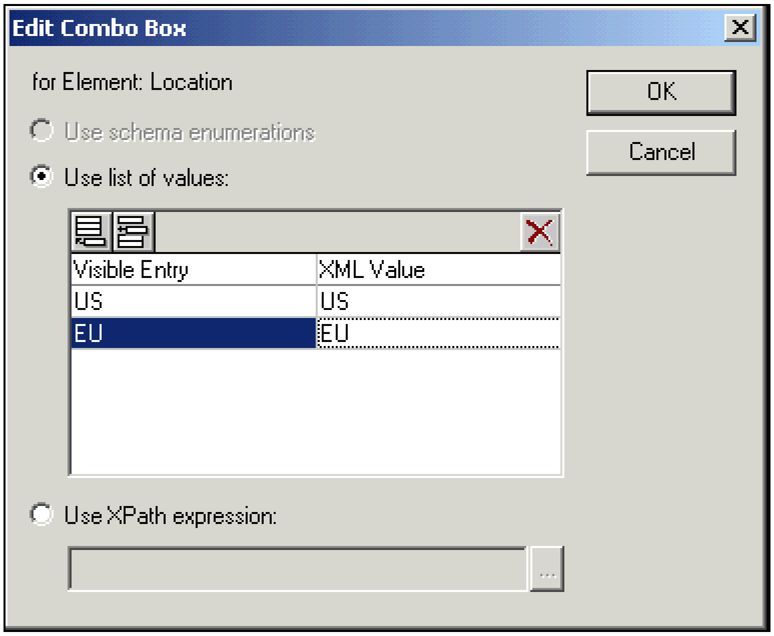


content origin: List (US:US;EU:EU)

3. Click the Edit button ... of the the `content origin` property. This pops up the Edit Combo Box dialog.

Using the Edit Combo Box dialog

The Edit Combo Box dialog is shown below.



Edit Combo Box

for Element: Location

○ Use schema enumerations

● Use list of values:

Visible Entry	XML Value
US	US
EU	EU

○ Use XPath expression:

To define the entries and values for the combo box, do the following:

1. Select the method with which you wish to define the entries and values by clicking the appropriate radio button.
2. If you select Schema Enumerations, the enumerations are entered in automatically. If you select Use List of Values, you can insert, append, edit, and delete any number of drop-down list entries with their corresponding XML values. If you wish to use values from the XML file, select Use XPath Expression, and enter or build an XPath expression to generate the desired sequence.
3. Click **OK** to finish.

Note

- Using XPath to select the items of the combo box drop-down list enables you to create combo boxes with dynamic entries from the XML file itself.
- You cannot sort items in the dropdown list of the combo box since there is no XPath function that achieves this. In order to obtain a sorted list, make sure that the source data (either the schema enumerations or the XML data) is sorted in the required order .
- You can modify the HTML properties of a combo box by selecting it and then modifying its HTML properties in the Properties entry helper.

Radio Buttons, Buttons

There are two types of button: radio buttons and buttons. Radio buttons allow the Authentic View user to enter data into the XML file. Buttons **do not allow** data-entry in Authentic View, but are useful for triggering events in the HTML output.

Radio buttons
Inserting radio buttons in the SPS allows you to give the user a choice among multiple alternatives. Each radio button you insert maps to one XML value. The user selects one radio button. The radio buttons for a node are mutually exclusive; only one may be selected at a time, and the associated XML value is entered as the value of the node. The way to use this feature is to create the node for which the data-entry is required multiple times as a radio button. For each radio button enter (i) some static text to indicate its value to the user, and (ii) an XML value for each radio button. To edit the XML value of a radio button, in the Properties entry helper, select the radio button item, then click the Edit button of the *radio button* property. This pops up the Edit Radio Button dialog.

Buttons
The button option allows you to insert a button and specify the text on the button. This is useful if you wish to associate scripts with button events in the generated HTML output. Note, however, that a button does not map to any XML value and does not allow data entry in Authentic View.

Note: You can modify the HTML properties of a radio button or button by selecting it and then modifying its HTML properties in the Properties entry helper.

7.4 Creating Lists

There are two types of lists that can be created in the SPS:

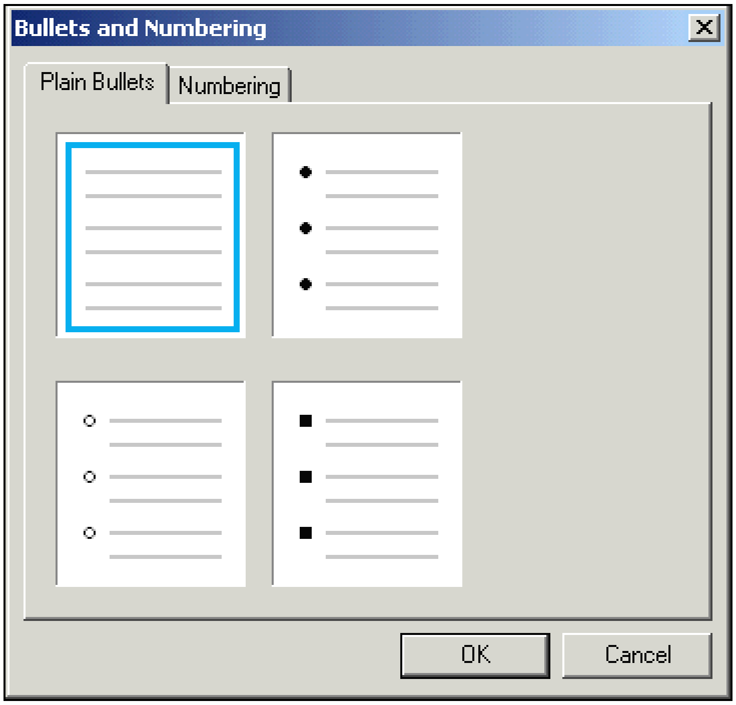
- Static lists, which are lists, the contents of which are entered directly i the SPS. The list structure is not dynamically derived from the structure of the XML document.
- Dynamic lists, which are lists that derive their structure and contents dynamically from the XML document.

These two list types are described in detail in the sub-sections of this section.

Static Lists

A static list is one in which list item contents are entered directly in the SPS. To create a static list, do the following:

1. Place the cursor at the location in the design where you wish to create the static list.
2. Click **Insert | Bullets and Numbering**. The Bullets and Numbering dialog pops up (*screenshot below*).



3. Select the desired list item marker and click **OK**. An empty list item is created.
4. Type in the text of the first list item.
5. Press **Enter** to create a new list item.

To create a nested list, place the cursor inside the list item that is to contain the nested list and click **Insert | Bullets and Numbering**. Then use the same procedure as given above, from Step 3 onwards, to create the nested list.

Changing static text to a list

There are two ways to change static text to a list:

- Highlight the text to change, click **Insert | Bullets and Numbering**, select the marker type and click **OK**. If the text contains a CR-LF, carriage-return and/or linefeed (inserted by pressing the **Enter** key), then separate list items are created for each text fragment separated by a CR-LF.
- With the cursor placed in a text fragment, click **Insert | Bullets and Numbering**, select the marker type and click **OK**. That text fragment, till the CR-LF separators on either side, is created as a list item.

Dynamic Lists

Dynamic lists display the content of a set of sibling nodes of the same name, with each node represented as a single list item in the list. The element, the instances of which are to appear as the list items of the list, is created as the list. The mechanism and usage are explained below.

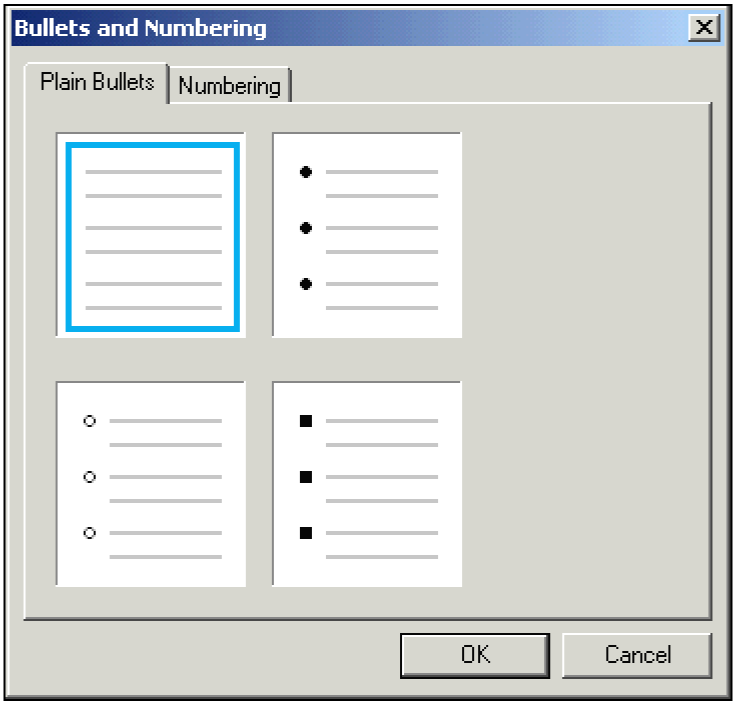
General usage mechanism

- Any element can be created as a list.
- When an element is created as a list, the instances of that element are created as the items of the list. For example, if in a `department` element, there are several `person` elements (i.e. instances), and you wanted to create a list of all the persons in the department, then you must create the `person` element as the list.
- Once the list has been created for the element, you can modify the appearance or content of the list or list item by inserting additional static or dynamic content such as text, Auto-Calculations, dynamic content, etc.

Creating a dynamic list

Create a dynamic list as follows:

1. Drag the element to be created as a list from the Schema Sources entry helper, and drop it at the desired location in the design.
2. From the context menu that pops up, select Create Bullets and Numbering. The Bullets and Numbering dialog (*screenshot below*) pops up.

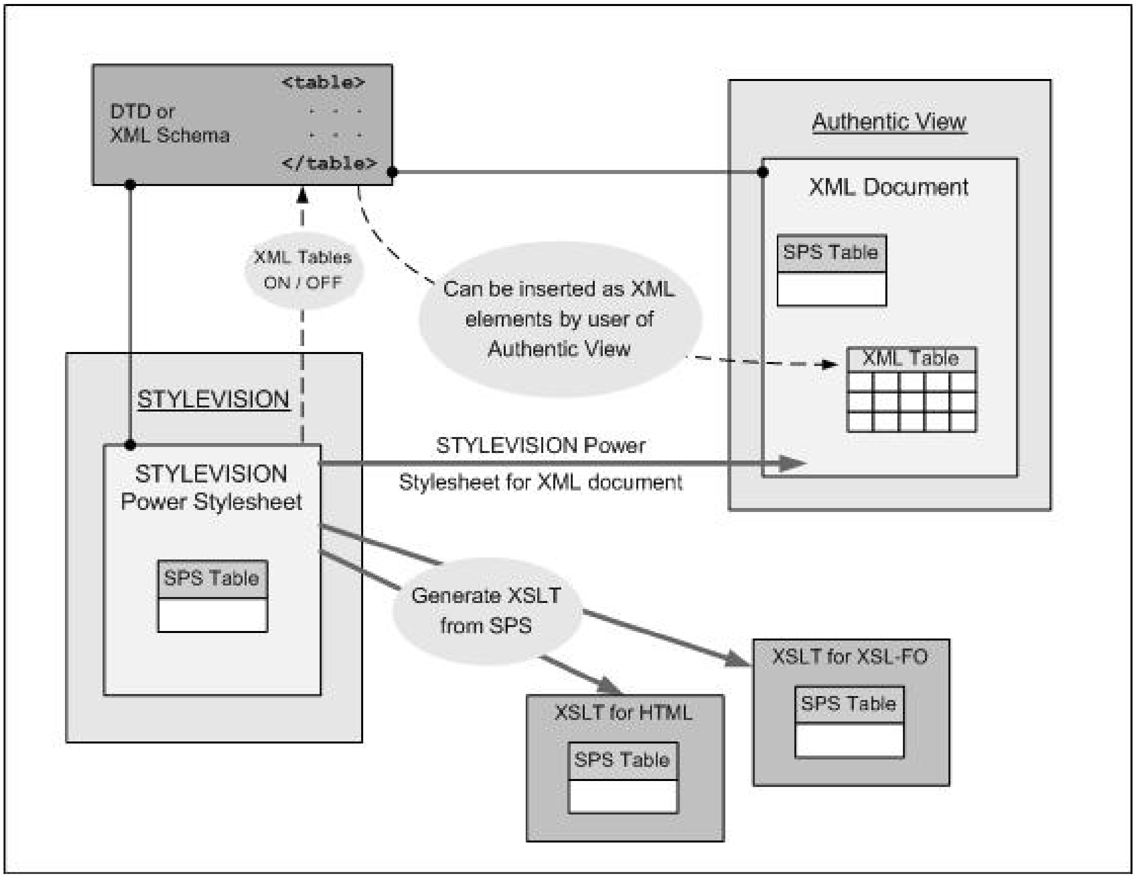


3. Select the required list type and click **OK**. The dynamic list is created.

7.5 Working with Tables

In an SPS, two types of tables are used: **SPS tables** and **XML tables**. There are crucial differences between the two types, and it is important to understand these. This section contains a detailed description of SPS tables and XML tables, and instructions about how to use them. For now, we look at the broad picture.

The illustration below shows the relationship of SPS tables and XML tables to the SPS and to the XML document.



SPS tables
An **SPS table** is a component of an SPS, and is created and formatted using StyleVision. If present in an SPS, an SPS table appears in Authentic View as well as in the XSLT stylesheets you generate with StyleVision.

The structure of an SPS table is specified by the person who designs the SPS. An SPS table can be created anywhere in an SPS, and any number of SPS tables can be created.

SPS tables are entirely presentational devices and are represented using the presentational vocabulary of Authentic View and the output format. The **structure** of an SPS table is **not represented by nodes in the XML document**—although the content of table cells may come from nodes in the XML document. SPS tables occur in three types of output:

- Rendered in Authentic View; a vocabulary specific to Authentic View is used to mark up SPS tables.
- In StyleVision-generated XSLT stylesheets for HTML output, SPS tables are marked up as HTML tables.

There are two types of SPS tables:

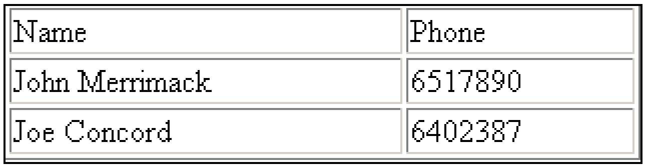
- **Static tables** are built up, step-by-step, by the person designing the SPS. After the table structure is created, the content of each cell is defined separately. The content of cells can come from random locations in the schema tree and even can be of different types. It is important to realize that the rows of a static table are not intended to represent a repeating data structure. This is why the table is said to be static: it has a fixed structure that does not change with the XML content.
- **Dynamic tables** are intended for data structures in the XML document that repeat. They can be created for schema elements that have a substructure—that is, at least one child attribute or element. Any element with a substructure repeats if there is more than one instance of it. Each instance of the element would be a row in the dynamic table, and all or some of its child elements or attributes would be the columns of the table. A dynamic table's structure, therefore, reflects the content of the XML file and changes dynamically with the content.

XML tables

An XML table is created by the **Authentic View user** as a data structure in the XML document. The purpose of XML tables is to give the Authentic View user the option of inserting a table-type data structure in the XML document. This XML data structure can then be transformed to the table markup of the output format.

The data structure for an XML table must correspond to either the HTML or CALS table model. One element in the XML document corresponds to the `table` element of the CALS or HTML table model, and must have a substructure that corresponds to either the CALS or HTML table model. An XML table can be inserted at any point in the XML document where it is allowed according to the schema. An XML table is formatted after it is inserted in the XML document.

Shown below is the Authentic View of an XML table that corresponds to the HTML table model.



Name	Phone
John Merrimack	6517890
Joe Concord	6402387

Data that is entered into the table's cells is entered as content of the corresponding XML elements. For example, the HTML text fragment for the XML table shown in the illustration above looks like this:

```
<table border="1" width="40%">
  <tbody>
    <tr>
      <td>Name</td>
      <td>Phone</td>
    </tr>
    <tr>
      <td>John Merrimack</td>
      <td>6517890</td>
    </tr>
    <tr>
      <td>Joe Concord</td>
      <td>6402387</td>
    </tr>
  </tbody>
</table>
```

The original XML document might look like this:

```
<phonelist border="1" width="40%">
  <items>
```

```
    <person>
      <name>Name</name>
      <phone>Phone</phone>
    </person>
    <person>
      <name>John Merrimack</name>
      <phone>6517890</phone>
    </person>
    <person>
      <name>Joe Concord</name>
      <phone>6402387</phone>
    </person>
  </items>
</phonelist>
```

Note that the element names in the XML document have no relation to table terminology; the table structure, however, corresponds to the HTML table model (it could also correspond to the CALS table model in order to be allowed as an XML table). Also note the following:

- An XML table can be inserted at any location in the XML document where, according to the schema, a table is allowed.
- In Authentic View, data is entered directly into table cells. This data is stored as the content of the corresponding XML table element.
- The formatting properties of an XML table are assigned in Authentic View.

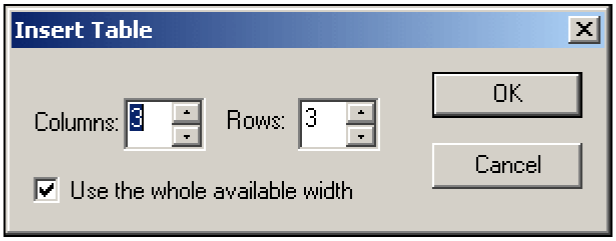
Note that XSLT stylesheets generated with StyleVision will not contain XML tables—because no template for the XML table is automatically included in the SPS.

Summary for designer
From the document designer's perspective, the following points should be noted:

- An **SPS table** occurs in the XML document at a location determined by the designer of the document—not the user of Authentic View. The structure and formatting of SPS tables are specified by the designer of the SPS in StyleVision.
- The location, structure, and formatting of **XML tables** are specified by the user of Authentic View. The user may insert an XML table wherever this is allowed by the schema (remember: the table element corresponds to an element in the schema). Note also that the table format of XML tables is available only in Authentic View. The HTML output will not automatically display a table format. You will have to create your own template/s to match the table element, and manually add these to the generated XSLT stylesheets.

Creating Static Tables

To insert a static table, click **Table | Insert table** or the ▦ icon. The following dialog appears:

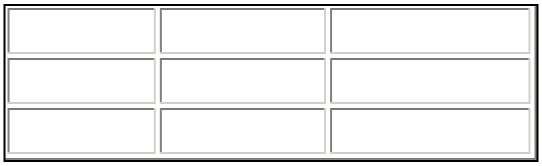


Insert Table

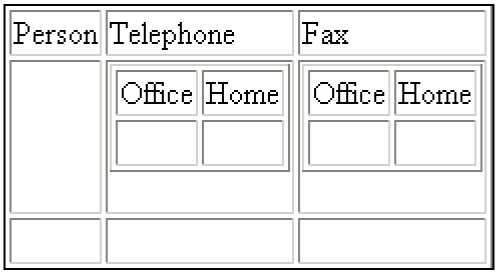
Columns: 3 Rows: 3

☑ Use the whole available width

You can select the dimensions of the table and specify whether the table should occupy the whole available width. When you click OK, an empty table with the specified dimensions, as shown below, is created.



You can now enter content into table cells using regular StyleVision features. Cell content could be text, or elements dragged from the XML tree, or objects such as images and nested tables. The figure below shows a table containing nested tables.



<table>
<tr><td>Person</td><td>Telephone</td><td>Fax</td></tr>
<tr><td></td><td><table><tr><td>Office</td><td>Home</td></tr><tr><td></td><td></td></tr></table></td><td><table><tr><td>Office</td><td>Home</td></tr><tr><td></td><td></td></tr></table></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

Static SPS tables are especially well-suited for organizing XML data that is randomly situated in the schema hierarchy.

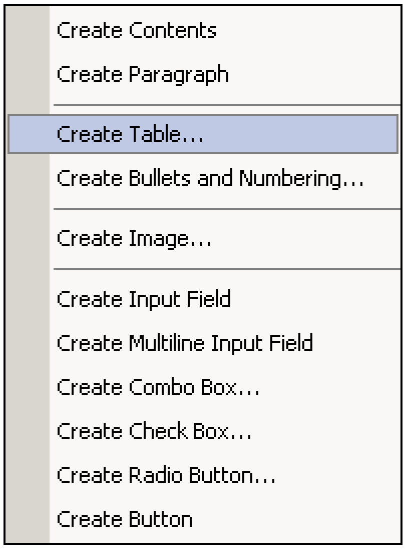
Deleting columns, rows, and tables
To delete a column, row, or table, place the cursor in the column, row, or table to be deleted, and click the menu item **Table | Delete Column**, **Table | Delete Row**, or **Table | Delete Table**, respectively. If you have nested tables, the table immediately containing the cursor will be deleted when the **Table | Delete Table** command is used.

Toolbar table editing icons
The table editing icons in the second row of the toolbar are shortcuts to the **Table** menu commands. These commands allow you to insert, delete, edit the structure of, and assign formatting properties to the static table. These icons can also be used for dynamic tables. They **cannot be used for XML tables**, since XML tables cannot be created in StyleVision but must be created in Authentic View by the user. XML tables can only be enabled in StyleVision.

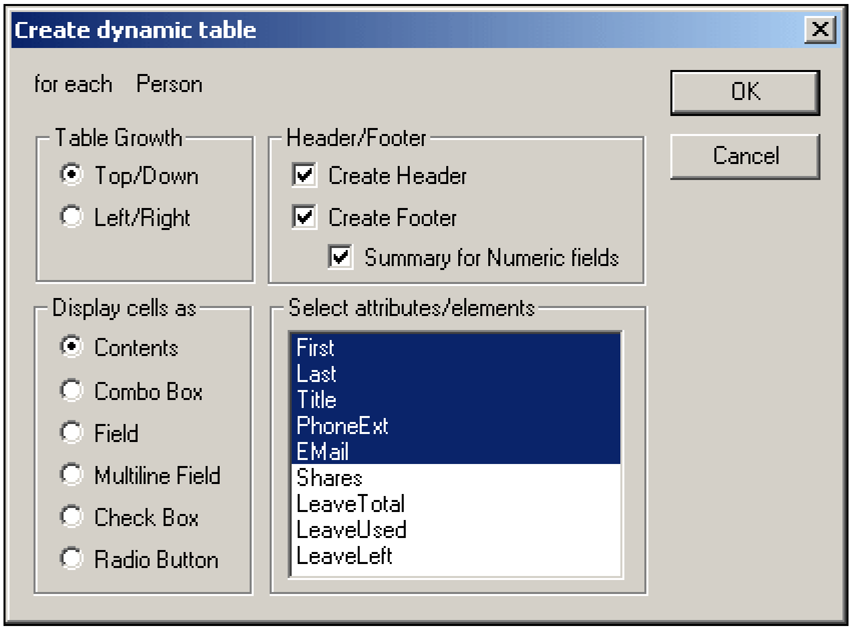
Creating Dynamic Tables

Creating a dynamic SPS table

A dynamic SPS table is created by dragging an element from the XML tree into the Design window. On dropping the element into the desired location, the following context menu appears.



Click Create Table. This brings up the following dialog.



for each Person

Table Growth
● Top/Down
○ Left/Right

Header/Footer
☑ Create Header
☑ Create Footer
☑ Summary for Numeric fields

Display cells as
● Contents
○ Combo Box
○ Field
○ Multiline Field
○ Check Box
○ Radio Button

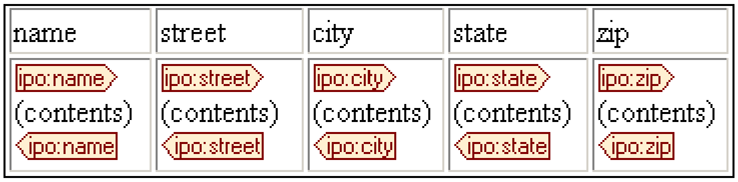
Select attributes/elements
First
Last
Title
PhoneExt
EMail
Shares
LeaveTotal
LeaveUsed
LeaveLeft

The child elements and attributes of the element that has been dragged into the Design window are displayed In the "Select attributes/element" list and can be created as columns of the table. Deselect the child nodes that you do not want and select any attribute/element you want to include as columns. (In the figure above, the elements `Shares`, `LeaveTotal`, `LeaveUsed` and `LeaveLeft` have been deselected.)

Note that columns are created only for child elements and attributes, but for no descendant on a lower level.

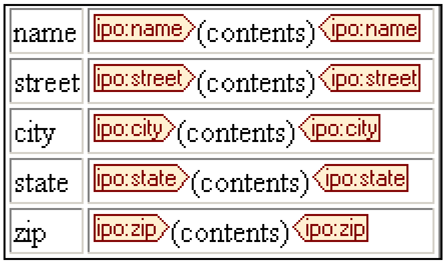
Table grows down or right

When a table grows top-down, this is what it would look like:



name	street	city	state	zip
ipo:name (contents) ipo:name	ipo:street (contents) ipo:street	ipo:city (contents) ipo:city	ipo:state (contents) ipo:state	ipo:zip (contents) ipo:zip

When a table grows left-right it looks like this:



name	ipo:name (contents) ipo:name
street	ipo:street (contents) ipo:street
city	ipo:city (contents) ipo:city
state	ipo:state (contents) ipo:state
zip	ipo:zip (contents) ipo:zip

Headers and footers

Columns can be given headers, which will be the names of the column elements. To include headers, check the Create Header check-box. To include footers, check the Create Footer check-box. Footers can only be created for tables that grow top-down. The footer of numeric columns will also sum each of these columns if the Summary for Numeric Fields check box is checked.

Header and footer cells can be joined and split, and rows can be inserted, appended, and deleted; this gives you considerable flexibility in structuring headers and footers. Additionally, headers and footers can contain any type of static or dynamic content, including conditional templates and auto-calculations.

Note: Headers and footers must be created when the dynamic table is defined. You do this by checking the Create Header and Create Footer options in the Create Dynamic Table dialog. Appending or inserting a row within a dynamic table does not create headers or footers but an extra row. The difference is significant. Real headers and footers are added to the top and bottom of a table, respectively. If a row is inserted or appended, then the row occurs for each occurrence of the element that has been created as a dynamic table.

Nested dynamic tables

You can nest one dynamic table within another dynamic table if the element for which the nested dynamic table is to be created is a child of the element that has been created as the containing dynamic table. Do the following:

1. Create the outer dynamic table so that the child element to be created as a dynamic table is created as a column.
2. In the dynamic table in Design View, right-click the child element.
3. Select **Change to | Table**. This pops up the Create Dynamic Table dialog.
4. Define the properties of the nested dynamic table.

To nest a dynamic table in a static table, drag the element to be created as a dynamic table into the required cell of the static table. When you drop it, select **Create Table** from the context menu that appears.

Tables for elements with text content

To create columns (or rows) for child elements, the element being created as a table must have a **child element or attribute node**. Having a **child text node** does not work. If you have this kind of situation, then create a child element called, say, `Text`, and put your text node in the `TableElement/Text` elements. Now you will be able to create `TableElement` as a dynamic table. This table will have one column for `Text` elements. Each row will therefore contain one cell containing the text node in `Text`, and the rows of the table will correspond to the occurrences of the `TableElement` element.

Contents of table body cells

When you create a dynamic table, you can create the node content as any one of a number of StyleVision components. In the examples above, the table body cells were created as contents. They could also have been created as data-entry devices. There are two points to note here:

- The setting you select is a global setting for all the table body cells. If you wish to have an individual cell appear differently, edit the cell after you have created the table: right-click in the cell and, in the context menu that appears, select "Change to" and then the required cell content type.
- If you create cells as element contents, and if the element has descendant elements, then the content of the cell will be a concatenation of the text strings of the element and all its descendant elements.

Deleting columns, rows, and tables

To delete a column, row, or table, place the cursor in the column, row, or table to be deleted, and click the menu item **Table | Delete Column**, **Table | Delete Row**, or **Table | Delete Table**, respectively. If you have nested tables, the table immediately containing the cursor will be deleted when the **Table | Delete Table** command is used.

Toolbar table editing icons

The table editing icons in the second row of the toolbar are shortcuts to the **Table** menu commands. These commands allow you to insert, delete, edit the structure of, and assign formatting properties to the dynamic table. These icons can also be used for static tables. They **cannot be used for XML tables**, since XML tables cannot be created in StyleVision but must be created in Authentic View by the user. XML tables can only be enabled in StyleVision.

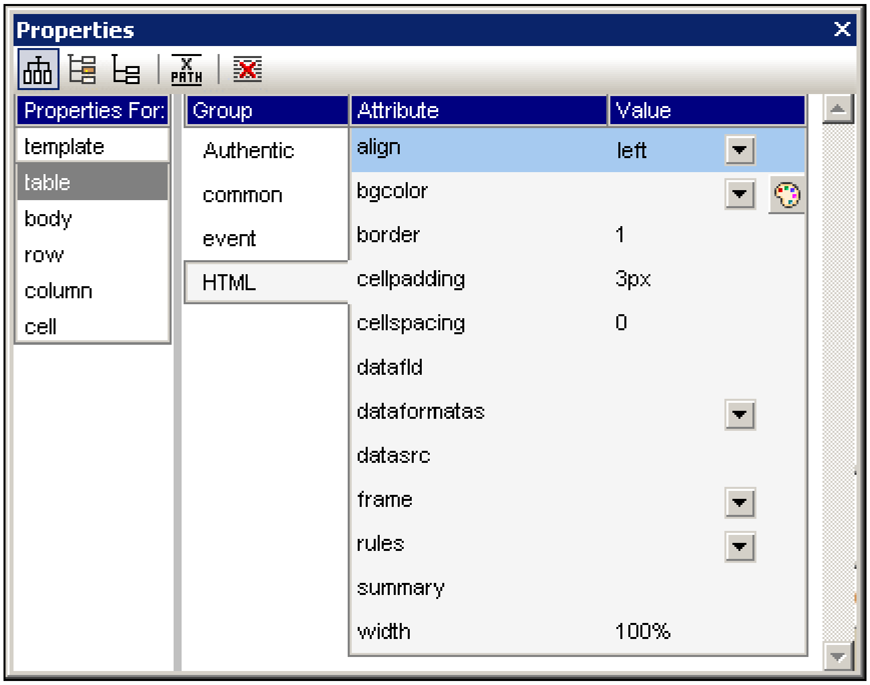
Creating dynamic tables in global templates

You can also create dynamic tables in global templates. The process works in the same way as for the Root Template (given above). The important point to note is that, in a global template, a dynamic table can only be created for **descendant elements** of the global template node; it cannot be created for the global template node itself. For example, if you wish to create a dynamic table for the element `authors` within a global template, then this dynamic table must be created within the global template of the parent element of `authors`. It cannot be created within the global template of the `authors` element.

Formatting Static and Dynamic Tables

Static and dynamic tables can be formatted using HTML table formatting properties and CSS properties.

The HTML table formatting properties are available in the Properties entry helper (*screenshot below*). These properties are available in the HTML group of properties for the table component and its sub-components (body, row, column, and cell).

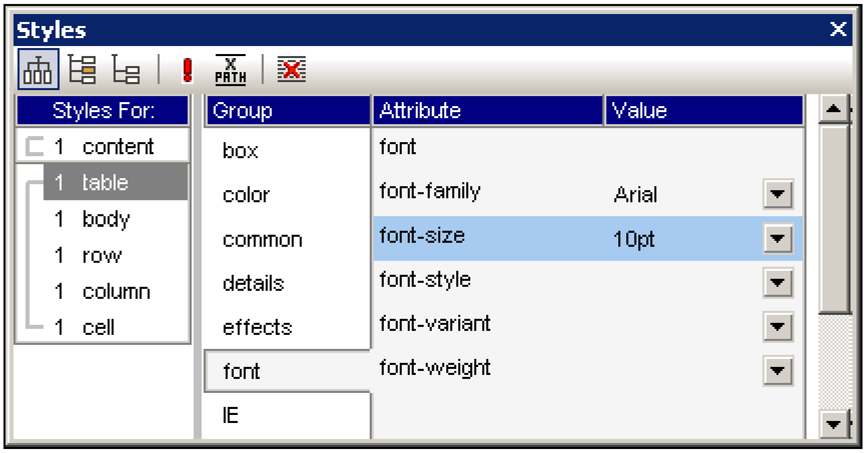


Properties For:
template
table
body
row
column
cell

Group
Authentic
common
event
HTML

Attribute	Value
align	left
bgcolor	
border	1
cellpadding	3px
cellspacing	0
datafld	
dataformatas	
datasrc	
frame	
rules	
summary	
width	100%

The CSS table formatting properties are available in the Styles entry helper (*screenshot below*). CSS properties are available for the table component and its sub-components (body, row, column, and cell).



Styles For:
1 content
1 table
1 body
1 row
1 column
1 cell

Group
box
color
common
details
effects
font
IE

Attribute	Value
font	
font-family	Arial
font-size	10pt
font-style	
font-variant	
font-weight	

Note: If all table cells in a row are empty, Internet Explorer collapses the row and the row might therefore not be visible. In this case, you should use the HTML workaround of putting a non-breaking space in the appropriate cell/s.

XML Tables

An XML table is defined as a hierarchical XML structure, the elements of which contain the cell content of the table. This XML structure must correspond exactly to the CALS or HTML table model. In order for **users of Authentic View** to be able to insert XML tables the following two conditions must be fulfilled:

- An element must exist in the schema (DTD or XML Schema) with a content model corresponding either to the HTML or CALS table model
- XML tables must be enabled in the StyleVision Power Stylesheet

Note: The purpose of XML tables is to give the **user of Authentic View** the option of entering data as a table in Authentic View. This data **will be displayed as a table in Authentic View but will not automatically be displayed as a table in HTMLoutput**. This is because no default processing for the XML table elements is defined. In order to obtain table formatting in the HTML output, you must manually define your own templates to provide processing for the XML table elements, and add these templates to the generated XSLT files. If you wish to have a table in your HTML output, we recommend that you use static and/or dynamic SPS tables.

Defining the table content model in the schema

The `table` element in the schema must have a content model corresponding to either the HTML or the Exchange model subset of the CALS table model. The content model of the `table` element in your schema **must correspond exactly** with either of these two table models, i.e. all elements and attributes defined in the table model must be correspondingly present in the element content model. For information about the CALS table model, see the CALS table model at OASIS. For an example of a `table` element having an HTML table structure, see the HTML-OrgChart XML Schema in the Examples folder (`HTML-OrgChart. xsd`).

A table model corresponding to the HTML table model would have a structure as shown in the XML fragment below.

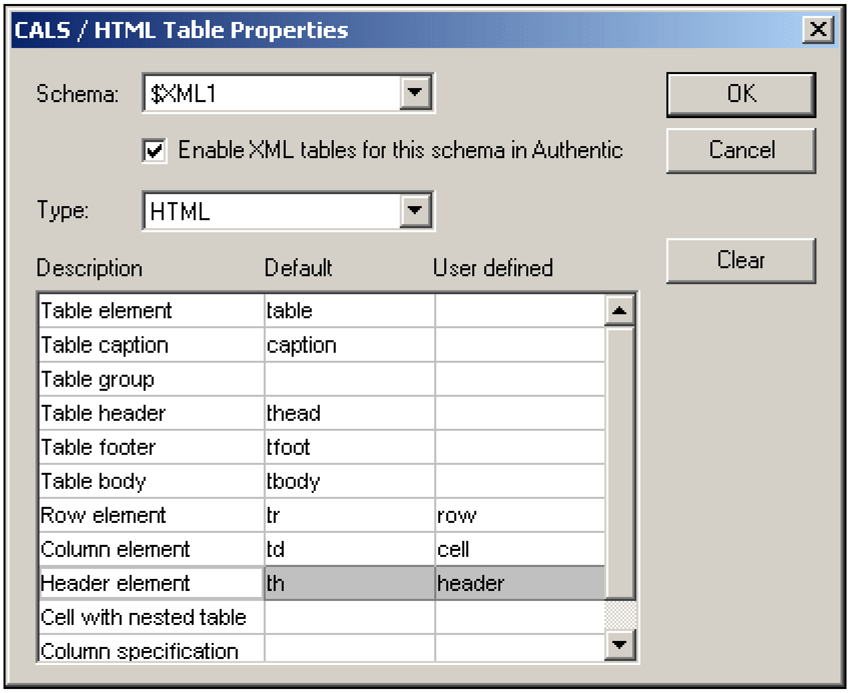
```
<table>
  <tbody>
    <tr>
      <td/>
    </tr>
  </tbody>
</table>
```

The element names in the example above are the default names you will find in the SPS. If the names of the table elements in your schema do not match these default names, you must map the names of the table elements in your schema to the default names in the CALS / HTML Table Properties dialog (**Authentic | CALS/HTML Tables**). You can do this when you check the Enable XML tables option in the dialog. If there is more than one element in the schema that has a valid table content model, then the mapping to the default names determines which element will be used as the table element.

Caution: If an element called `table` exists in the schema, it will be treated as the XML table element if XML tables have been enabled in the StyleVision Power Stylesheet (because `table` is the default name for the table element in the SPS). This could lead to errors if the element `table` is not intended to be used as a table element.

Enabling XML tables with StyleVision

In order for the user of Authentic View to be able to create XML tables in an XML document, XML tables must be enabled in the SPS. To enable XML tables, click **Authentic | CALS/HTML Tables**. This pops up the CALS / HTML Table Properties dialog (*screenshot below*).



CALS / HTML Table Properties

Schema: $XML1

☑ Enable XML tables for this schema in Authentic

Type: HTML

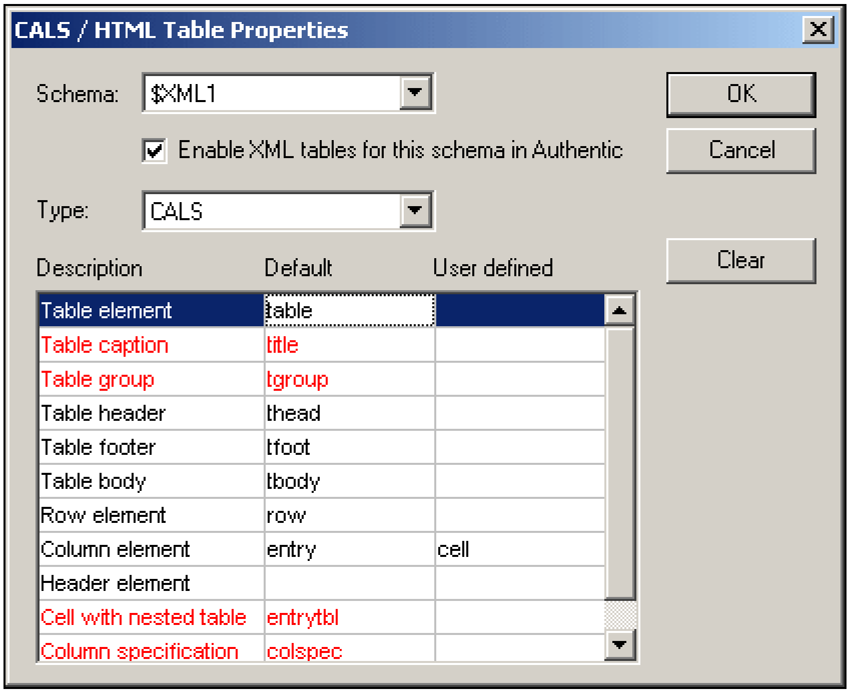
Description	Default	User defined
Table element	table	
Table caption	caption	
Table group		
Table header	thead	
Table footer	tfoot	
Table body	tbody	
Row element	tr	row
Column element	td	cell
Header element	th	header
Cell with nested table		
Column specification		

With the relevant schema selected, check the "Enable XML tables in Authentic" check-box.

Then select the table model type: either HTML or CALS. The screenshot above shows the default names for the HTML table model. If an element in the table model in your schema matches a default element name, the default element name is colored black, otherwise red. If a corresponding element exists in your schema for a default element in the CALS/HTML table model, then you can map your schema element to the default by entering its name in the User Defined column for that default element. The screenshot above indicates the following about the schema:

- There is an element called `table` with a content model the same as the HTML table model (all default names appear black).
- The elements corresponding to the default `tr`, `td`, and `th` are `row`, `cell`, and `header`, respectively.

The content model of a `table` element that follows the HTML model would not correspond with the CALS table model, which is different. Given below is a screenshot of the CALS / HTML Table Properties dialog (for the same schema as above) with the table model type set to CALS.



CALS / HTML Table Properties

Schema: $XML1

☑ Enable XML tables for this schema in Authentic

Type: CALS

Description	Default	User defined
Table element	table	
Table caption	title	
Table group	tgroup	
Table header	thead	
Table footer	tfoot	
Table body	tbody	
Row element	row	
Column element	entry	cell
Header element		
Cell with nested table	entrytbl	
Column specification	colspec	

Notice the following:

- The default CALS table elements that do not exist in the schema are colored red. Those CALS table model elements that do exist in the `table` element's content model appear black, including `row`.
- The `entry` element has been mapped to `cell`; this causes both `entry` and `cell` to be displayed in black.

If you wish to use the CALS table model, you should alter the schema to properly include the missing elements and attributes. Once you have selected the appropriate table model, click OK. XML tables are now ready to be used in Authentic View.

Note:
The following general points about XML tables should be noted:

- The `table` element can only be inserted at locations in the XML document where the schema allows the table element.
- You, as the person who designs the SPS, only enables XML tables. It is the Authentic View user who inserts an XML table at his or her discretion.
- An XML table is structured and formatted in Authentic View, that is, by the Authentic View user. The formatting of an XML table cannot be controlled through the SPS.
- In the CALS table model, the `cols` attribute of the Table group element (`tgroup`) specifies the number of columns in the table. This attribute–value pair is entered (both attribute and value) into the XML document when the Authentic View user inserts an XML table. This is because the user must specify the number of columns when inserting the table. The value of this attribute changes automatically whenever a column is inserted, appended, or deleted using the Authentic View GUI tools. The `cols` attribute is therefore not shown in the Attributes Entry Helper and its value, therefore, cannot be modified there.

- Data entered in the cells of an XML table is entered as content of the `entry` element (in the case of a CALS table) or `td` element (HTML table), or corresponding user-defined element, as the case may be.
- To obtain the content of XML tables in HTML output, you must generate the XSLT (**File | Save Generated Files**) and, in it, manually create a template for outputting the table element.

7.6 Using Graphics

There are two ways in which graphics are used in an SPS:

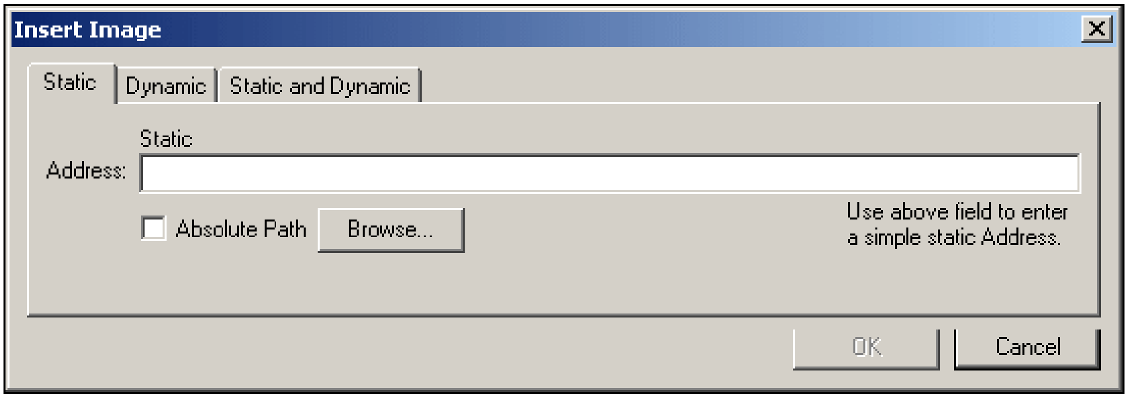
- As images in the design document, and
- As Authentic View toolbar icons for applying markup to the XML document (text state icons).

When inserting images in the design document, the location of the image can be specified directly in the SPS (by the SPS designer) or can be taken or derived from a node in the XML document. How to specify the location of the image is described in the section Image URIs. The section Text State Icons describes how toolbar icons for Authentic View can be defined.

Image URIs

Images can be inserted at any location in the design document. These images will be displayed in Authentic View and the output documents; in Design View, inserted images are indicated with placeholders.

To insert an image, click the **Insert | Image** menu command, which pops up the Insert Image dialog (*screenshot below*). The URI of the image to be inserted is entered in this dialog.



There are three ways in which the URI of the image can be entered:

- In the Static tab, directly as an absolute or relative URI. For example, `nanonull.gif` (*relative URI; see section below*), and `C:/images/nanonull.gif` (*absolute URI*).
- In the Dynamic tab, as an XPath expression that selects a node containing either (i) a URI (absolute or relative), or (ii) an unparsed entity name. For example, the entry `image/@location` would select the `location` attribute of the `image` element that is the child of the context node (that is, the node within which the image is inserted). The location node in the XML document would contain the image URI. How to use unparsed entities is described in the section Unparsed Entity URIs.
- In the Static and Dynamic tab, an XPath expression in the Dynamic part can be prefixed and/or suffixed with static entries (text). For example, the static prefix could be `C:/XYZCompany/Personnel/Photos/`; the dynamic part could be `concat(First, Last)`; and the static suffix could be `.png`. This would result in an absolute URI something like: `C:/XYZCompany/Personnel/Photos/JohnDoe.png`.

Accessing the image for output

The following points should be noted:

- For Design View and Authentic View in StyleVision, as well as for Authentic View in Altova products, you can set, in the Properties dialog, whether relative paths to images should be relative to the SPS or to the XML file.
- For HTML output, the URI of the image is passed to the HTML file and the image is accessed by the browser. So, if the path to the image is relative, it must be relative to the location of the HTML file. For the HTML Preview in StyleVision, a temporary HTML file is created in the same folder as the SPS file, so, for rendition in HTML Preview, relative paths must be relative to this location.
- Whether the URI is relative or absolute, the image must be physically accessible to the process that renders it.

Editing image properties

To edit an image, right-click the image placeholder in Design View, and select Image Properties from the context menu. This pops up the Edit Image dialog, which is the same as the Insert Image dialog (*screenshot above*) and in which you can make the required modifications. The Edit Image dialog can also be accessed via the `URL` property of the *image* group of properties in the Properties window. The *image* group of properties also includes the `alt` property, which specifies alternative text for the image.

Deleting images

To delete an image, select the image and press the Delete key.

Image Types and Output

The table below shows the image types supported by StyleVision in the various output formats supported by StyleVision.

Image Type	Authentic	HTML	RTF	PDF
JPEG	Yes	Yes	Yes	Yes
GIF	Yes	Yes	Yes	Yes
PNG	Yes	Yes	Yes	Yes
BMP	Yes	Yes	Yes	Yes
SVG	No	No	No	Yes

Note the following points:

- FOP reports an error if an image file cannot be located and does not generate a PDF.
- If FOP is being used to produce PDF, rendering PNG images requires that the JIMI image library be installed and accessible to FOP.
- For more details about FOP's graphics handling, visit the FOP website.
- RTF and PDF outputs are supported only in the Enterprise edition of StyleVision; is supported in the Enterprise and Professional editions.

Text State Icons

A Text State Icon is a toolbar icon in Authentic View which can be used to mark up text that is inside an element of mixed content. The markup that is added is that for descendant elements of the mixed-content element. When the markup is added, the newly marked-up element in Authentic View takes the formatting assigned to the global template of that element.

Prerequisites for creating a text state icon

The following prerequisites apply:

- A text state icon can be created only for an element that is a child of an element of mixed content.
- The element for which a text state icon is to be created must be declared in the XML Schema as a global element. (In a DTD, all elements are global elements.)

Text state icons are most commonly used to mark up bold and italic text in paragraphs. The XML Schema declarations for such a document structure typically would be something like this:

```
<xs:element name="bold" type="TextType"/>
<xs:element name="italic" type="TextType"/>
<xs:complexType name="TextType" mixed="true">
       <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="bold"/>
              <xs:element ref="italic"/>
       </xs:choice>
</xs:complexType>
<xs:element name="para" type="TextType"/>
```

The `para` element is of mixed content and may contain `bold` and `italic` child elements. The `bold` and `italic` elements may themselves contain `bold` and `italic` child elements. In such a structure, text state icons can be created for the `bold` and `italic` elements.

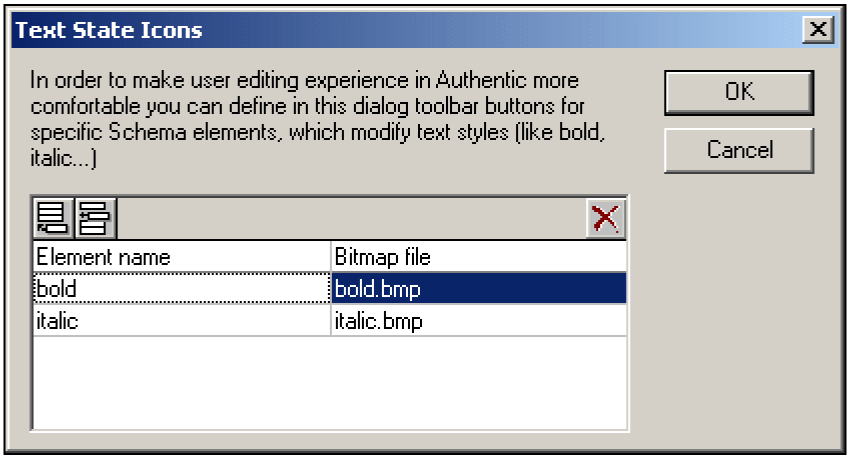
Procedure for creating text state icons

The procedure for creating text state icons can be divided into two broad steps:

- Assign a bitmap image file (the text state icon) to the element to be marked up with the help of the text state icon.
- Define formatting, in a global template, for the element to be marked up.

Assuming the schema structure defined above, text state icons for the `bold` and `italic` elements would be created as follows.

1. Select the menu command **Authentic | Text State Icons**. The Text State Icons dialog (*screenshot below*) appears:



Text State Icons

In order to make user editing experience in Authentic more comfortable you can define in this dialog toolbar buttons for specific Schema elements, which modify text styles (like bold, italic...)

Element name	Bitmap file
bold	bold.bmp
italic	italic.bmp

2. Enter `bold` as the element for which the text state icon is being created.
3. Enter `bold.bmp` as the name of the bitmap image file that is to be associated with the element `bold`. The icon file `bold.bmp` must be saved in a folder called `sps\Picts` in your application folder. This image will be used as the toolbar icon in Authentic View.
4. Now append a row in the dialog and add the `italic` element and `italic.bmp` as the bitmap file for the italic icon.
5. Click **OK** to finish and save the SPS.

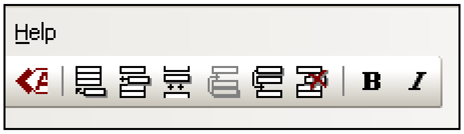
The steps above assign bitmap image files to the elements to be marked up (`bold` and `italic`). Next, formatting for the two elements must be defined in the global templates of these elements. This would be done as follows:

1. In the All Global Elements list in the Schema Sources window, right-click the `bold` element and select **Make Global / Remove from Global**. This creates a global template in the SPS.
2. In Design View, select the (contents) placeholder and assign it a formatting of bold. (In the Font group of the Styles window, set font-weight to bold.)
3. Repeat the previous two steps for the `italic` element, but set font-style to italic.
4. Save the SPS.

Now formatting has been defined in global templates for the bold and italic elements.

Using text state icons in Authentic View

The text state icons you create can be viewed when that SPS (or XML document associated with that SPS) is opened in any Authentic View except that of StyleVision (*screenshot below*). (In the `StyleVision2007/Examples/Tutorials/Images` folder of the StyleVision application folder, the file `Images.xml`, which is associated with the SPS file `Images.sps` demonstrates the use of the `bold` and `italic` text state icons.)



In our example, the text state icons will be grayed out when the cursor is not inside a `para` element. They will be enabled when you place the cursor inside a `para` element (because the `bold` and `italic` elements are child elements of `para`). To apply markup by using text state icons, first highlight some text inside the `para` element in the example file, and then click a text state icon. The relevant markup will be inserted and the corresponding formatting will be applied.

Removing text state icons
To remove a text state icon, in the Text State Icons dialog (**Authentic | Text State Icons**, *screenshot of dialog above*), place the cursor in the row containing tetext state icon to be deleted and click the **Delete** button at the top right of the dialog. Then click **OK** to finish.

Example: A Template for Images

The StyleVision package contains an SPS file that demonstrates the use of images in StyleVision. This file is: `StyleVision2007/Examples/Tutorials/Images/Images.sps`). The Images document (`Images.xml` and `Images.sps`) consists of three parts:

- The first part shows how text state icons can be used in Authentic View (can be created in Enterprise and Professional editions only). When you open the file `Images.xml` in the Authentic View of XMLSpy, Authentic Desktop, or Authentic Browser, you can try out the use of text state icons. Note that text state icons are not available in the Authentic Preview of StyleVision, so you cannot try out this feature in StyleVision. How to create text state icons is described in the Text State Icons section of this user manual.
- The second part contains a table showing which image formats are supported in the various StyleVision output formats. Note that the RTF and PDF output formats are available only in the Enterprise Edition of StyleVision. In Design View, only images with static URIs will be displayed. All the image formats listed in the table are displayed in Part 3 of the Images document.
- In Part 3, all the popular image formats supported by StyleVision are displayed one below the other. After opening the file `Images.sps` in StyleVision, you can switch among the various previews of StyleVision to see how each image is displayed in that preview. Since the location of the image is in an XML node, you can also enter the location of your own images in Authentic View and test their appearances in the preview windows.

7.7 Bookmarks and Hyperlinks

In the SPS document, bookmarks can be inserted anywhere within the design. These bookmarks are transformed into anchors in the output, which can be linked to from hyperlinks. Hyperlinks can not only link to bookmarks, but also to external resources like Web pages. StyleVision offers considerable flexibility in the way target URIs for hyperlinks can be built.

In this section, we describe:

- How bookmarks can be inserted in the SPS.
- How hyperlinks can be inserted in the SPS and how they link to the pages required.

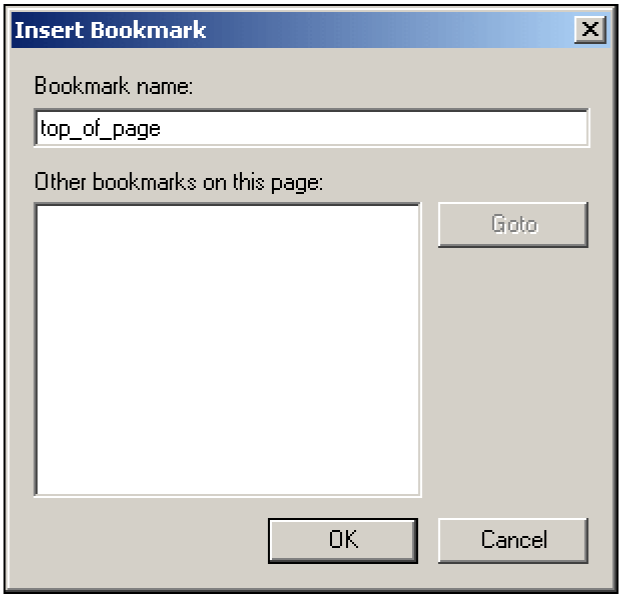
Inserting Bookmarks

A static bookmark (or anchor) can be inserted anywhere in the SPS, at a cursor insertion point or around an SPS component. A bookmark is created for a single, fixed location in the SPS. If created around an element that repeats, the bookmark is effectively created around the first occurrence of that element, and hyperlinks that reference this bookmark will link to the first occurrence of the bookmarked element.

Creating a bookmark

To insert a bookmark, do the following:

1. Place the cursor at the location where you wish to create the bookmark.
2. Select the menu command **Insert | Bookmark**, or right-click and select **Insert | Bookmark**.
3. In the Insert Bookmark dialog (*screenshot below*), enter a name for the bookmark in the Name field. The name of the bookmark is a static name that you give. It does not come dynamically from the XML document. You cannot assign a dynamic name to a bookmark. (If there are more bookmarks in the design, they are displayed in the Other Bookmarks on This Page window.)



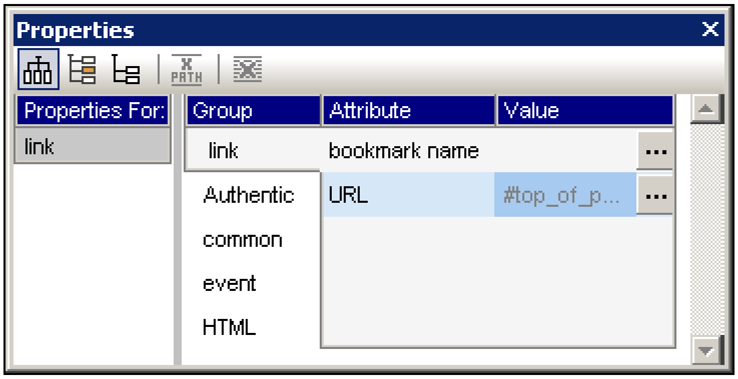
4. Click **OK**. The bookmark is defined.

After a bookmark has been created, it can be linked to by a hyperlink.

Modifying a bookmark

After a bookmark has been created, its name can be modified via the Edit Bookmarks dialog. This dialog is accessed as follows:

1. Select the bookmark in the design.
2. In the Properties entry helper, click the **Edit** button of the `URL` property (*screenshot below*) in the *link* group of properties. This pops up the Edit Bookmarks dialog.



Properties For:
link

Group	Attribute	Value
link	bookmark name	
Authentic	URL	#top_of_p...
common		
event		
HTML		

3. In the Other Bookmarks pane of the dialog, select the bookmark, and, in the text input field, enter a new name.

Deleting a bookmark

To delete a bookmark, select it in the design and press the **Delete** key.

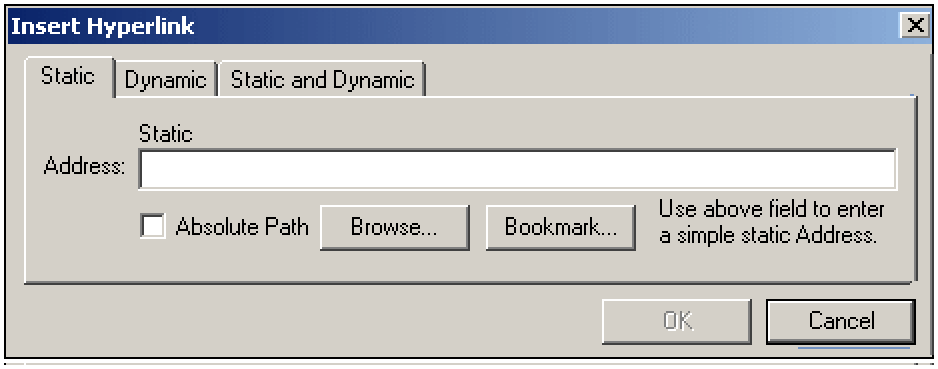
Defining Hyperlinks

Hyperlinks can be created around SPS components such as text or images. The targets of hyperlinks can be: (i) bookmarks in the SPS design, and (ii) external resources, such as web pages or email messages. In this section, we first discuss link content and then the targets of links.

Creating hyperlinks

A hyperlink can be created around text (static or dynamic), nodes, images, conditional templates, Auto-Calculations, and blocks of content or nodes; it cannot be created around a data-entry device such as an input field or combo box—though it can be created around a node or conditional template in which that data-entry device is.

To create a hyperlink select the required component in the design, then click the menu command **Insert | Hyperlink**, or right-click and select **Insert | Hyperlink**, or use the shortcut **Ctrl+L**. This pops up the Insert Hyperlink dialog (*screenshot below*), in which you specify the target of the link.



Use above field to enter a simple static Address.

The target of a link can be either:

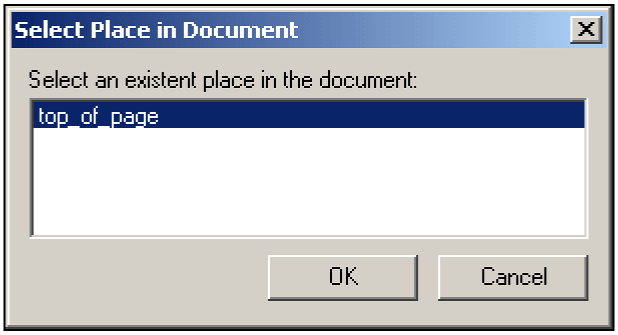
- A bookmark in the same SPS design (in which case the URI is a fragment identifier), or
- An external resource; the URI can be static (directly entered), dynamic (taken from a node in an XML document), a combination of static and dynamic parts, or the value of an unparsed entity.

How these two kinds of targets are defined is explained below. After the URI has been defined in the Insert/Edit Hyperlink dialog, click OK to finish.

Linking to bookmarks

To link to a bookmark, do the following:

1. In the Static tab of the Insert Hyperlink dialog, click the Bookmark button. This pops up the Select Place in Document dialog (*screenshot below*).



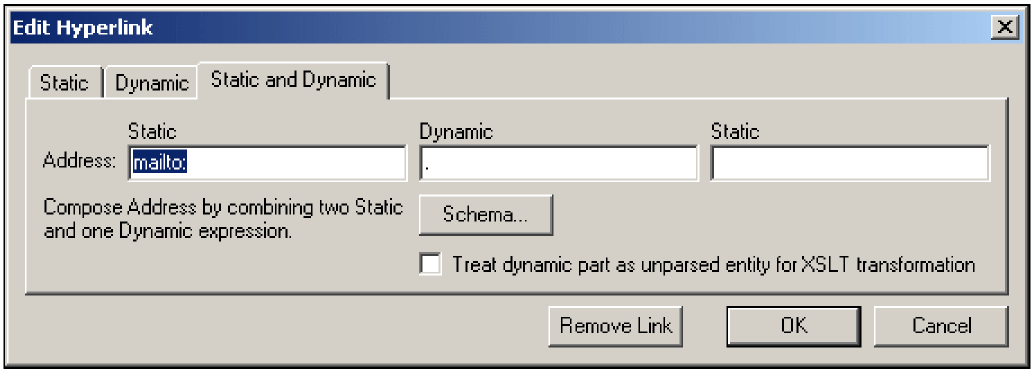
2. This dialog lists all the bookmarks defined in the SPS. Select the bookmark you wish to target, and click OK. The bookmark is entered as a fragment identifier in the URI input field.

When the XML document is transformed with an XSLT stylesheet the fragment identifier will be the target of the hyperlink in the output document.

Linking to external resources

URIs that locate external resources can be built in the following ways:

- By entering the URI directly in the Static tab of the Insert Hyperlink dialog. For example, a link to the Altova home page (`http://www.altova.com`) can be entered directly in the Address input field of the Static tab.
- By selecting a node in the XML document source in the Dynamic tab of the Insert Hyperlink dialog. The node in the XML source can provide a text string that is either: (i) the URI to be targeted, or (ii) the name of an unparsed entity which has the required URI as its value. For example, the Altova website address can be contained as a text string in a node.
- By building a URI that has both static and dynamic parts in the Static and Dynamic tab of the Insert Hyperlink dialog. This can be useful for adding static prefixes (e.g. a protocol) or suffixes (e.g. a domain name). For example, email addresses can be created using a static part of `mailto:` and a dynamic part that takes the string content of the `//Contact/@email` node (*screenshot below*).



Address: mailto: | . |

Compose Address by combining two Static and one Dynamic expression.

□ Treat dynamic part as unparsed entity for XSLT transformation

How to use unparsed entities is described in the section Unparsed Entity URIs.

Editing hyperlink properties

To edit a hyperlink, right-click either the start or end hyperlink (`A`) tag, and select Hyperlink Properties from the context menu. This pops up the Edit Hyperlink dialog (*screenshot above*). The Edit Hyperlink dialog can also be accessed via the `URL` property of the *link* group of properties in the Properties window. The *link* group of properties also includes the `bookmark name` property, which specifies the name of a bookmark in the document that can the target of the link. The Properties window thus provides access to the targets of the hyperlink, enabling these to be edited.

Removing and deleting hyperlinks
A hyperlink is an SPS component (for example, a text string or image), which links to a document fragment or other resource, via a URI. When the target URI is cleared, the hyperlink is said to be **removed**; the SPS component however remains in the design document. On the other hand, when a hyperlink is **deleted**, the SPS component is deleted from the design document. These two actions are carried out as follows:

- To remove a hyperlink, right-click the hyperlink and, in the Edit Hyperlink dialog (*screenshot above*), press the Remove Link button. The target URI is cleared.
- To delete a hyperlink, select the hyperlink (by clicking either the start or end hyperlink (`A`) tag), and press the Delete key. The hyperlink and its contents are deleted.

7.8 Auto-Calculations

The **Auto-Calculation** feature (i) displays the result of an XPath evaluation at any desired location in the output document, and (ii) optionally updates a node in the XML document with the result of the XPath evaluation.

The Auto-Calculation feature is a useful mechanism for including:

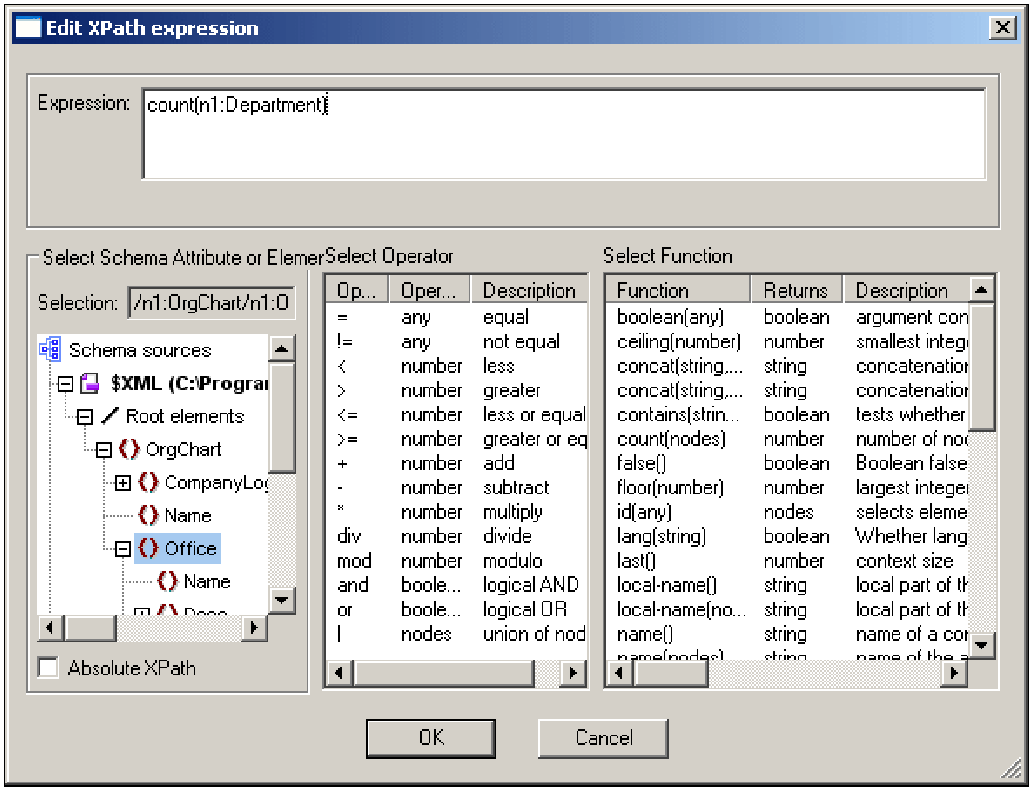
- Calculations involving operations on dynamic data values. For example, you can count the number of `Employee` elements in an `Office` element (with `count( Employee)`), or sum the values of all `Price` elements in each `Invoice` element (with `sum( Price)`), or join the `FirstName` and `LastName` elements of a `Person` element (with `concat( FirstName, ' ', LastName)`). In this way you can generate new data from dynamically changing data in the XML document, and send the generated data to the output.
- Calculations based on the dynamic structure of the document. For example, you can use the `position()` function of XPath to dynamically insert row numbers in a dynamic table, or to dynamically number the sections of a document. This has the advantage of automatically generating information based on dynamically changing document structures.
- Inserting data from external XML documents. The `doc()` function of XPath 2.0 provides access to the document root of external XML documents, and thus enables node content from the external XML document to be inserted in the output.
- The value of nodes in the XML document can be changed by updating them with the value of an Auto-Calculation. For example, the node `Addressee` could be updated with an XPath expression like `concat( Title, ' ', FirstName, ' ', LastName)`.
- The Auto-Calculation feature is also a useful way to display the contents of some node at another location.

Editing and Moving Auto-Calculations

Creating Auto-Calculations

To create an Auto-Calculation, do the following:

1. Place the cursor as an **insertion point** at the location where the Auto-Calculation result is to be displayed and click **Insert | Auto-Calculation**. In the submenu that appears, select Value if the result is to appear as plain text, select Input Field if it is to appear within an input field (i.e. a text box), or select Multiline Input Field if it is to appear in a multiline text box. The Edit XPath Expression dialog pops up (*screenshot below*).



Expression: count(n1:Department)

Select Schema Attribute or Elemer

Selection: /n1:OrgChart/n1:O

Select Operator

Op...	Oper...	Description
=	any	equal
!=	any	not equal
<	number	less
>	number	greater
<=	number	less or equal
>=	number	greater or eq
+	number	add
-	number	subtract
*	number	multiply
div	number	divide
mod	number	modulo
and	boole...	logical AND
or	boole...	logical OR
|	nodes	union of nod

Select Function

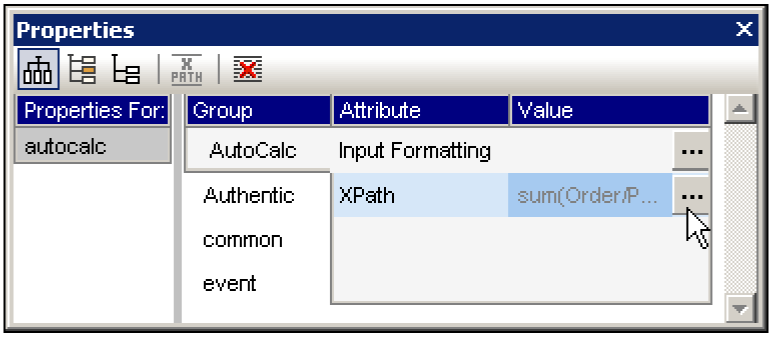
Function	Returns	Description
boolean(any)	boolean	argument con
ceiling(number)	number	smallest integ
concat(string,...	string	concatenatior
concat(string,...	string	concatenatior
contains(strin...	boolean	tests whether
count(nodes)	number	number of noc
false()	boolean	Boolean false
floor(number)	number	largest integer
id(any)	nodes	selects eleme
lang(string)	boolean	Whether lang
last()	number	context size
local-name()	string	local part of th
local-name(no...	string	local part of th
name()	string	name of a cor
name(nodes)	string	name of the a

2. In the Expression pane, enter the XPath expression for the Auto-Calculation via the keyboard. Alternatively, enter the expression by double-clicking nodes, operators, and/or functions in the entry helper panes of the dialog. It is important to be aware of the context node at the insertion point; the context node is highlighted in the schema sources tree when the dialog pops up. (If you have selected XSLT 1.0 as the version of the XSLT language for your SPS, then you must use XPath 1.0 expressions; if you have selected XSLT 2.0, then you must use XPath 2.0 expressions.)
3. Optionally, if you wish to copy the value of the Auto-Calculation to a node in the XML document, you can select that node via an XPath expression. How to update nodes with the result of the Auto-Calculation is described in the section, Updating Nodes with Auto-Calculations.

Click the **OK** button finish. In the Design tab, the Auto-Calculation symbol is displayed. To see the result of the Auto-Calculation, change to Authentic View or an Output View.

Editing Auto-Calculations

To edit the XPath expression of the Auto-Calculation, select the Auto-Calculation and, in the Properties entry helper, click the **Edit** button of the `XPath` property in the *AutoCalc* group of properties (*screenshot below*). This pops up the Edit XPath Expression dialog (*screenshot* *above*), in which you can edit the XPath expression.



Formatting Auto-Calculations

You can apply predefined formats and CSS styles to Auto-Calculations just as you would to normal text: select the Auto-Calculation and apply the formatting. Additionally, input formatting of an Auto-Calculation that is a numeric or date datatype can be specified via the Input Formatting property in the AutoCalc group of properties in the Properties window.

Note also that you can include carriage returns and/or linefeeds (CR/LFs) in the XPath expression. If the Auto-Calculation is enclosed in the `pre` special paragraph type, the output of a CR/LF will produce a new line in the output. An example of such an XPath expression is:

```
translate('a;b;c', ';', codepoints-to-string(13))
```

Moving Auto-Calculations

You can move an Auto-Calculation to another location by clicking the Auto-Calculation (to select it) and dragging it to the new location. You can also use cut/copy-and-paste to move/copy an Auto-Calculation. Note, however, that the XPath expression will need to be changed if the context node in the new location is not the same as that in the previous location.

Summary of important points

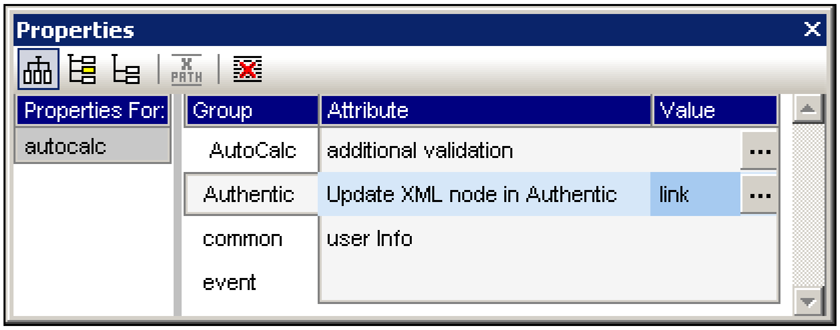
Note the following points:

- An Auto-Calculation can be inserted anywhere in the Design Document.
- The point at which you insert the Auto-Calculation determines the context node for the XPath evaluation.
- In Authentic View, an Auto-Calculation is re-evaluated each time any value relevant to the calculation (that is, any node included in the XPath expression) changes.
- An Auto-Calculation result is non-editable in Authentic View or any other output view.
- Any node in the XML document can be updated with the result of the Auto-Calculation.

Updating Nodes with Auto-Calculations

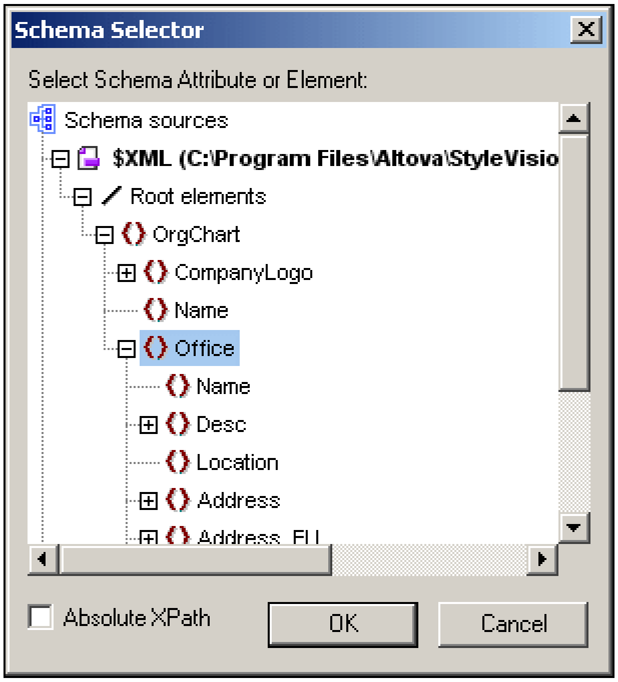
You can copy the value (or result) of an Auto-Calculation to a node in the XML document. You do this as follows:

1. Create the Auto-Calculation as described in Editing and Moving Auto-Calculations.
2. Select the Auto-Calculation and, in the Properties entry helper, click the **Edit** button of the `Update XML node in Authentic` property in the *Authentic* group of properties (*screenshot below*).



Properties For:	Group	Attribute	Value
autocalc	AutoCalc	additional validation	...
	Authentic	Update XML node in Authentic	link ...
	common	user Info	
	event		

3. In the Schema Selector dialog that pops up (*screenshot below*), select the node to update and click **OK** to finish.



The node to update is now specified.

The XPath expression must select **a single node**. If the XPath expression selects multiple nodes, no node will be updated.

IMPORTANT!
For a node in the XML document to be updated two conditions must be fulfilled:

1. The XPath expression for the Auto-Calculation must include at least one value that is

related to an XML node, i.e. a dynamic value. For example, `Price*1.2`. This expression involves the element `Price`, and is therefore dynamic. If the XPath expression contains a static value (for example, `string("Nanonull, Inc.")`), then the Update XML Node feature will not work.

2. Any one of the nodes used in the XPath expression must be modified in Authentic View —not in any other view. So, if the XPath expression is `Price*1.2`, and the Auto-Calculation is set to update the `VATPrice` node, then the `Price` element must be modified in Authentic View in order for the `VATPrice` node to be updated.

Changing the node to update and cancelling the update

To change the node to update, click the **Edit** button of the `Update XML Node` property in the *Authentic* group of properties of the Auto-Calculation (in the Properties window) and then select the required node from the Schema Selector dialog box that pops up. To delete the `Update XML Node` property, click the Remove button in the toolbar of the Properties window.

Should you use the Auto-Calculation or the updated node contents for display?

If there are no conditional templates involved, you can use either the Auto-Calculation or the contents of the updated node for display. It is immaterial which one you choose because the node update happens immediately after the Auto-Calculation is evaluated and there is no factor to complicate the update. You should, however, be aware that there is a different source for the content displayed in each of the two cases.

Hiding the Auto-Calculation

You may find yourself in the situation where you wish to use an Auto-Calculation to make a calculation in order to update a node with the value of the Auto-Calculation. In this case, one of the following scenarios arise:

- You wish to display the result just once. You cannot hide the Auto-Calculation since it would then not be evaluated. If there is no conditional template involved, it is best to display the Auto-Calculation and not display the contents of the updated node.
- You wish not to display the result, merely to update the node. The best way to handle this scenario is to apply text formatting to the Auto-Calculation, so that it is invisible on the output medium (for example, by applying a white color to an Auto-Calculation on a white background).

Auto-Calculations Based on Updated Nodes

If you wish to create an Auto-Calculation (second Auto-Calculation) that uses a node updated by another Auto-Calculation (first Auto-Calculation), there are two possible situations:

- The two Auto-Calculations are in the same template templates.
- The two Auto-Calculations are in different templates.

Auto-Calculations in the same template
When two Auto-Calculations are in the same template, the SPS applies the following procedure:

1. A node used in the XPath expression of the first Auto-Calculation is modified.
2. All node values in the XML document are read and all Auto-Calculations are executed.
3. Assuming that the first Auto-Calculation is executed correctly, it updates the specified XML node (call it `Node-A`). The second Auto-Calculation, which is based on `Node-A`, will be executed but will use the value of `Node-A` before `Node-A` was updated. This is because the value of `Node-A` was read before it was updated, and has not been read since then.
4. If the document is now edited in any way or if document views are changed (from and to Authentic View), then the values of nodes are read afresh and Auto-Calculations are executed.
5. The second Auto-Calculation is now carried out. (If this Auto-Calculation is intended to update a node, then, as is usual for node updates, a node used in the XPath expression will have to be changed before the update takes place.)

The time lag between the updating of `Node-A` and the evaluation of the second Auto-Calculation with the updated value of `Node-A` could be confusing for the Authentic View user. To ensure that this situation does not occur, it is best that the XPath expression of the second Auto-Calculation contain the XPath expression of the first Auto-Calculation—not the updated node itself. As a result, the second Auto-Calculation will execute with the input to the first Auto-Calculation and perform that Auto-calculation as part of its own Auto-Calculation. This enables it to be evaluated independently of the contents of `Node-A`.

Example
The first Auto-Calculation calculates the VAT amount of a product using the nodes for (i) the net price, and (ii) the VAT rate; it updates the VAT-amount node. The second Auto-Calculation calculates the gross price, which is the sum of net price and VAT amount; it updates the gross price node.

- The Auto-Calculation to calculate the VAT amount is: `NetPrice * VATRate div 100.` When the VAT rate of the product is entered, the Auto-Calculation is executed and updates the `VATAmount` node.
- If the Auto-Calculation to calculate the gross price is: `NetPrice + VATAmount`, then the Auto-Calculation will execute with the value of `VATAmount` that was read in before `VATAmount` was updated.
- If, however, the Auto-Calculation to calculate the gross price is: `NetPrice + (NetPrice * VATRate div 100)`, then the Auto-Calculation will execute with the value of `VATRate` and will update the `GrossPrice` node. The updated `VatAmount` node has been left out of the second Auto-Calculation.

For a detailed example, see Example: An Invoice.

Auto-Calculations in different templates
When two Auto-Calculations are in different templates, a node updated by the first Auto-Calculation can be used by the second Auto-Calculation. This is because Auto-Calculations are

calculated and nodes updated for each template separately. For an example of how this would work, see Example: An Invoice.

Example: An Invoice

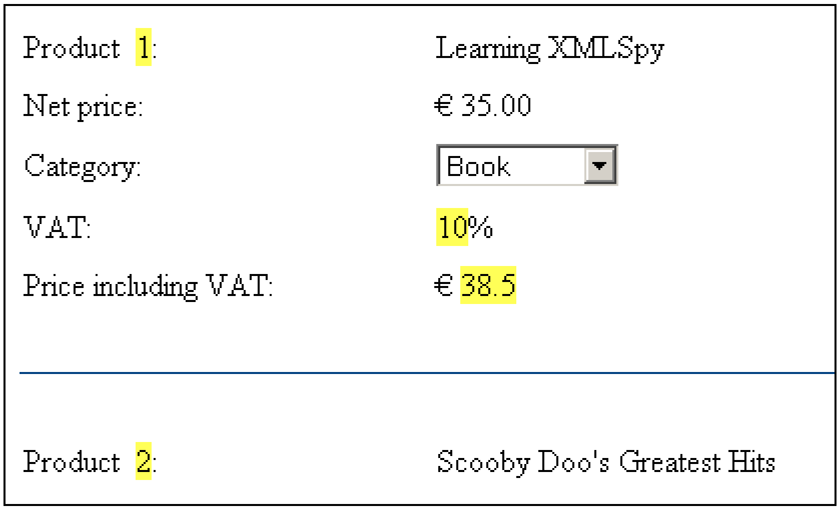
The `Invoice.sps` example in the folder `StyleVision2007/Examples/Tutorials/Auto-Calculations/` demonstrates how Auto-Calculations can be used for the following purposes:

- Counting nodes
- Selecting a node based on input from the Authentic View user
- Updating the content of a node with the result of an Auto-Calculation
- Using the result of one Auto-Calculation in another Auto-Calculation

In the example file, the Auto-Calculations have been highlighted with a yellow background color (*see screenshot below*).

Counting nodes

In the Invoice example, each product in the list is numbered according to its position in the list of products that a customer has ordered (`Product 1`, `Product 2`, etc). This numbering is achieved with an Auto-Calculation (*screenshot below*).



Product 1:	Learning XMLSpy
Net price:	€ 35.00
Category:	Book
VAT:	10%
Price including VAT:	€ 38.5
Product 2:	Scooby Doo's Greatest Hits

In this particular case, the XPath expression `position()` would suffice to obtain the correct numbering. Another useful way to obtain the position of a node is to count the number of preceding siblings and add one. The XPath expression would be: `count(preceding-sibling::Product)+1`. The latter approach could prove useful in contexts where using the `position()` function is difficult to use or cannot be used. You can test this Auto-Calculation in the example file by deleting products, and/or adding and deleting new products.

Selecting a node based on user input

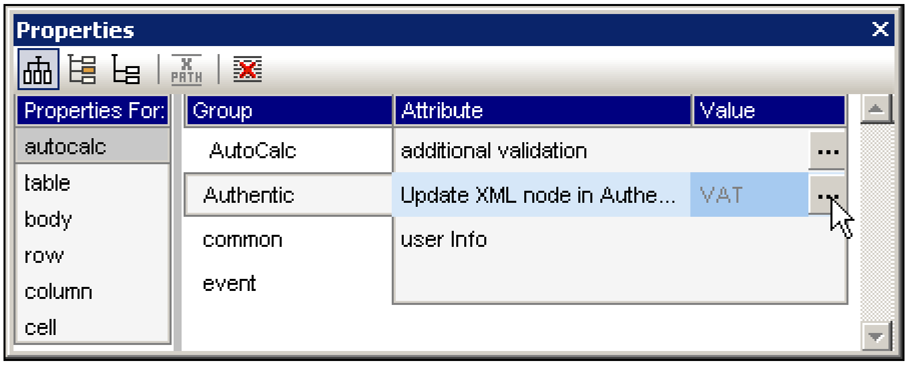
In the Invoice example, the user selects the category of product (`Book`, `CD`, `DVD`, or `Electronics`) via a combo box. This selection is entered in the `//Product/Category` node in the XML document. An Auto-Calculation then uses this value to reference a "lookup table" in the XML document and identify the node holding the VAT percentage for this product category. The XPath expression of this Auto-Calculation is:

```
for $i in Category return /Invoice/Categories/Category[. = $i]/@rate.
```

The VAT percentage is displayed at the Auto-Calculation location in the output. In the Invoices example, the lookup table is stored in the same XML document as the invoice data. However, such a table can also be stored in a separate document, in which case it would be accessed using the `doc()` function of XPath 2.0. To test this Auto-Calculation, change the product type selection in any product's Category combo box; the VAT value will change accordingly (`Book=10%; CD=15%; DVD=15%; Electronics=20%`).

Updating content of a node with the result of an Auto-Calculation

The VAT percentage obtained by the Auto-Calculation from the lookup is dynamic and stored temporarily in memory (for use in Authentic View). The result of the Auto-Calculation can, however, be stored in the `VAT` node in the XML document. This has two advantages: (i) the content of the `VAT` node does not have to be entered by the user; it is entered automatically by the Auto-Calculation; (ii) each time the lookup table is modified, the changes will be reflected in the VAT node when `Invoice.xml` is opened in Authentic. To cause an Auto-Calculation to update a node, select the Auto-Calculation in Design View, and in the Properties window (*screenshot below*), click the *Authentic* | *Update XML Node* button. In the dialog that pops up, select the `VAT` node, and then click **OK**.



Group	Attribute	Value
AutoCalc	additional validation	...
Authentic	Update XML node in Authe...	VAT ...
common	user Info	
event		

In Authentic View, when the user selects a different product category in the combo box, the Auto-Calculation obtains the VAT percentage by referencing the lookup table, displays the VAT percentage, and updates the `VAT` node.

Using an Auto-Calculation-updated node in another Auto-Calculation

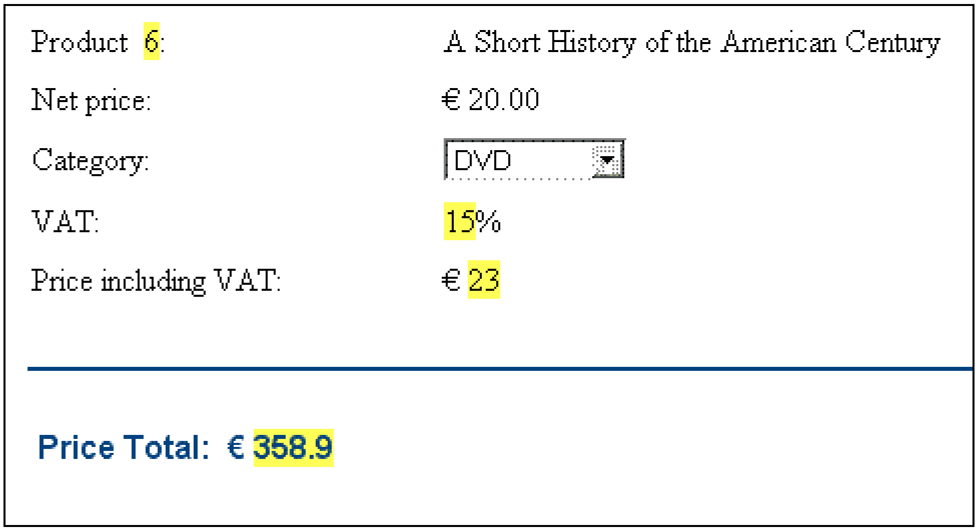
The VAT percentage, obtained by the Auto-Calculation described above, is required to calculate the gross price (net price + VAT amount) of each product. The formula to use would be derived as follows:

```
Gross Price = Net Price + VAT-amount
Since VAT-amount = Net Price * VAT-percentage div 100
Gross Price = Net Price + (Net Price * VAT-percentage div 100)
```

The net price of a product is obtained from the `PriceNet` node. The VAT percentage is calculated by an Auto-Calculation as described above, and this Auto-Calculation updates the `VAT` node. The content of the `VAT` node can now be used in an Auto-Calculation to generate the gross price. The XPath expression to do this would be:

```
PriceNet  + (PriceNet * (VAT div 100))
```

The XPath expression can be viewed and edited in the Properties window. You can test the Auto-Calculation for the gross price by changing either the price or product category of any product. Notice that the gross price (price including VAT) of the product also changes.



Product 6:	A Short History of the American Century
Net price:	€ 20.00
Category:	DVD
VAT:	15%
Price including VAT:	€ 23

Price Total: € 358.9

In the Invoice SPS, the gross price Auto-Calculation updates the `PriceGross` node in the XML document.

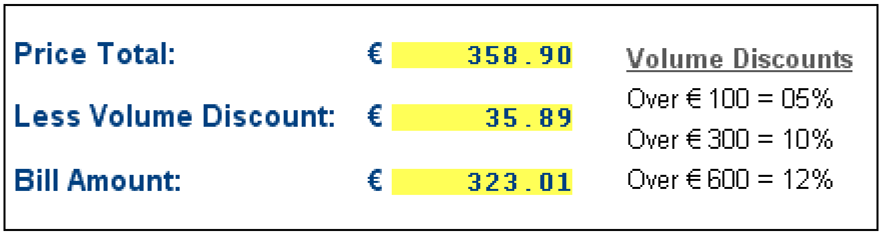
The updated `PriceGross` nodes can now be used in an Auto-Calculation that sums up the prices of all purchased products. The XPath expression would be: `sum( Order/Product/PriceGross)` . In the Invoice SPS, this Auto-Calculation updates the `PriceTotal` node. You can test this Auto-Calculation by modifying the prices of individual products and seeing the effect on the price total.

An Auto-Calculation Exercise

Now add two Auto-Calculation components to the SPS yourself.

1. Create an Auto-Calculation that calculates a volume discount for the entire invoice. If the order amount (price total) exceeds Euro 100, Euro 300, or Euro 600, discounts of 5%, 10%, and 12% apply, respectively. Display the discount amount (*see screenshot below*) and update the `DiscountAmount` node in the XML document.
2. Create an Auto-Calculation that calculates the discounted bill amount. This amount would be the price total less the discount amount (as calculated in the previous Auto-Calculation). Display the bill amount (*see screenshot below*) and update the `BillAmount` node in the XML document.

Set up these components so that the Authentic View output is as shown in the screenshot below.



Price Total:	€	358.90	**Volume Discounts**
Less Volume Discount:	€	35.89	Over € 100 = 05%
Bill Amount:	€	323.01	Over € 300 = 10%
			Over € 600 = 12%

You can see these two additional Auto-Calculations in the file `InvoiceWithDiscounts.sps`, which is in the `StyleVision2007/Examples/Tutorials/Auto-Calculations` folder.

7.9 Conditions

You can insert conditions anywhere in the design, in both the main template and global templates. A condition is an SPS component that is made up of one or more branches, with each branch being defined by an XPath expression. For example, consider a condition composed of two branches. The XPath expression of the first branch tests whether the value of the `Location` attribute of the context node is "`US`". The XPath expression of the second branch tests whether the value of the `Location` attribute is "`EU`". Each branch contains a template—a condition template. When a node is processed with a condition, the first branch with a test that evaluates to true is executed, that is, its condition template is processed, and the condition is exited; no further branches of that condition are evaluated. In this way, you can use different templates depending on the value of a node. In the example just cited, different templates could be used for US and EU locations.

This section consists of the following topics:

- Setting Up the Conditions, which describes how to create a condition and its branches.
- Editing Conditions, about how to edit the XPath expressions of condition branches after they have been created.
- Conditions for Specific Outputs, which shows how conditions are used to produce different output for different output formats.
- Conditions and Auto-Calculations, explains usage issues when conditions and Auto-Calculations are used in combination.

Setting Up the Conditions

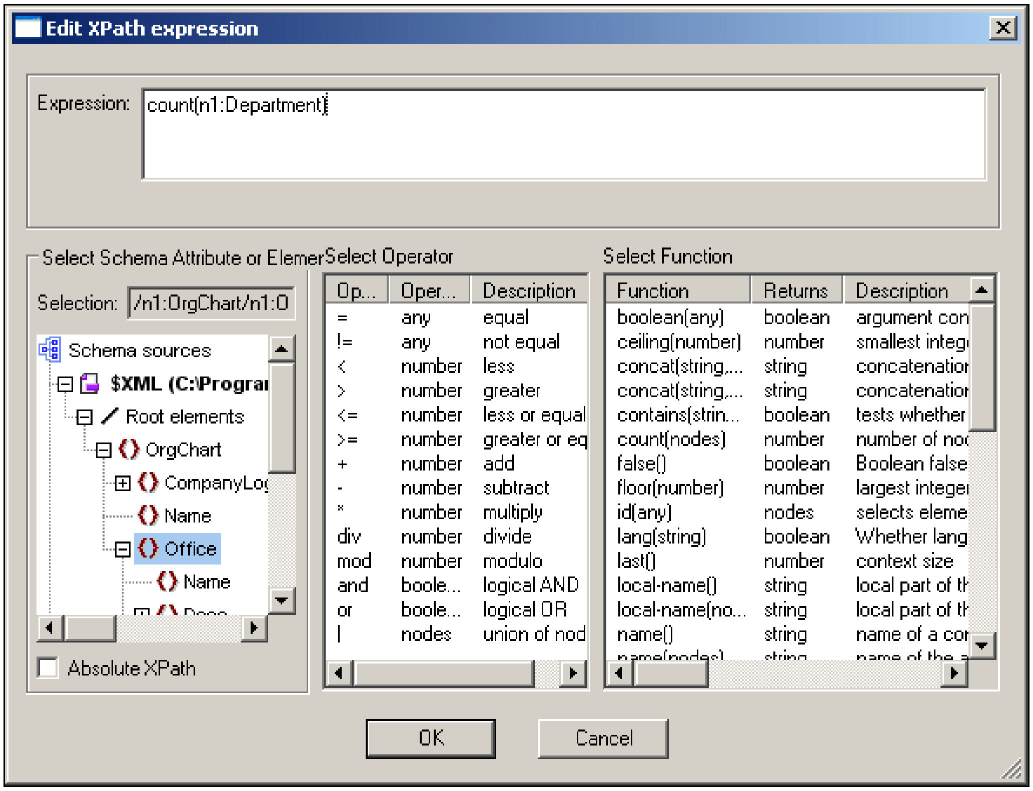
Setting up the condition consists of the following steps:

1. Create the condition with its first branch.
2. Create additional branches for alternative processing.
3. Create and edit the templates within the various branches of the condition.

Creating the condition with its first branch

Set up a condition as follows:

1. Place the cursor anywhere in the design or select a component and then select the menu command **Insert | Condition**.The Edit XPath Expression dialog pops up (*screenshot below*).



Expression: count(n1:Department)

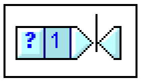
Selection: /n1:OrgChart/n1:O

☐ Absolute XPath

Op...	Oper...	Description
=	any	equal
!=	any	not equal
<	number	less
>	number	greater
<=	number	less or equal
>=	number	greater or eq
+	number	add
-	number	subtract
*	number	multiply
div	number	divide
mod	number	modulo
and	boole...	logical AND
or	boole...	logical OR
|	nodes	union of nod

Function	Returns	Description
boolean(any)	boolean	argument con
ceiling(number)	number	smallest integ
concat(string,...	string	concatenatior
concat(string,...	string	concatenatior
contains(strin...	boolean	tests whether
count(nodes)	number	number of noc
false()	boolean	Boolean false
floor(number)	number	largest integer
id(any)	nodes	selects eleme
lang(string)	boolean	Whether lang
last()	number	context size
local-name()	string	local part of th
local-name(no...	string	local part of th
name()	string	name of a cor
name(nodes)	string	name of the

2. In the Expression pane, enter the XPath expression for the condition branch via the keyboard. Alternatively, enter the expression by double-clicking nodes, operators, and/or functions in the entry helper panes of the dialog. It is important to be aware of the context node at the insertion point; the context node is highlighted in the schema sources tree when the dialog pops up.
3. Click **OK** to finish. The condition is created with its first branch; the XPath expression you entered is the XPath expression of the first branch. If the condition was inserted at a text insertion point, the first branch is empty (there is no template within it; *see screenshot below*). If the condition was inserted with a component selected, the condition is created around the component, and that component becomes the template of the first branch.

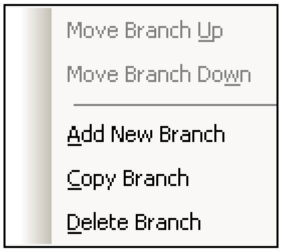


To select the entire condition, click the cell with the question mark. To select the first branch, click the cell with the number one.

After creating a condition with one branch (which may or may not have a template within it), you can create as many additional branches as required.

Creating additional branches

Additional branches are created one at a time. An additional branch is created via the context menu (*screenshot below*) and can be created in two ways: (i) without any template within it (**Add New Branch**); and (ii) with a copy of an existing template within the new branch (**Copy Branch**).



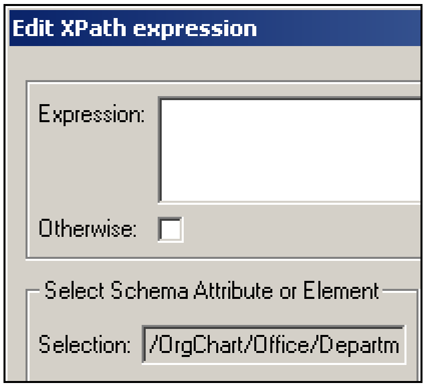
To create a new branch, right-click any branch of the condition and select **Condition | Add New Branch** from the context menu. The Edit XPath Expression dialog will pop up. After entering an XPath expression and clicking **OK**, a new empty branch is added to the condition. This is indicated in the design by a new cell being added to the condition; the new cell has a number incremented by one over the last branch prior to the addition.

To create a copy of an existing branch, right-click the branch of the condition you wish to copy and select **Condition | Copy Branch**. The Edit XPath Expression dialog will pop up, containing the XPath expression of the branch being copied. After modifying the XPath expression and clicking **OK**, a new branch is added to the condition. The new branch contains a copy of the template of the branch that was copied. The new branch is indicated in the design by a new cell with a number incremented by one over the last branch prior to the addition.

The Otherwise branch

The `Otherwise` branch is an alternative catch-all to specify a certain type of processing (template) in the event that none of the defined branches evaluate to true. Without the `Otherwise` branch, you would either have to create branches for all possible eventualities or be prepared for the possibility that the condition will be exited without any branch being executed.

To insert an `otherwise` branch, use either the **Add New Branch** or **Copy Branch** commands as described above, and in the Edit XPath dialog click the Otherwise check box (*screenshot below*).



Selection: /OrgChart/Office/Departm

Moving branches up and down

The order of the branches in the condition is important, because the first branch to evaluate to true is executed and the condition is then exited. To move branches up and down relative to each other, select the branch to be moved, then right-click and select **Condition | Move Branch Up** or **Condition | Move Branch Down**.

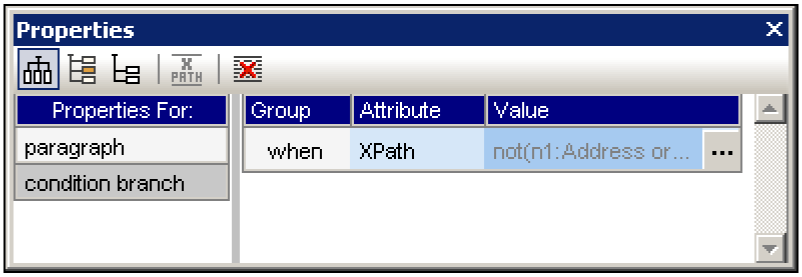
Deleting a branch

To delete a branch, select the branch to be deleted, then right-click and select **Condition | Delete Branch**.

Editing Conditions

To edit the XPath expression of a condition branch, do the following:

1. Select the condition branch (not the condition).
2. In the Properties entry helper, select `condition branch` in the Properties For column (*screenshot below*).



Properties For:
paragraph
condition branch

Group	Attribute	Value
when	XPath	not(n1:Address or...

3. Click the **Edit** button ... of the `XPath` property in the *When* group of properties. This pops up the Edit XPath Expression dialog, in which you can edit the XPath expression for that branch of the condition.

Output-Based Conditions

Individual components in the document design can be processed differently for StyleVision's different output formats (Authentic View and HTML). For example, consider the case where you wish to create a link, which, in Authentic View should point to a file on a local system, but in the HTML output should point to a Web page. In this case, you can create one condition to process content for Authentic View output and a second condition to process content for HTML output. Or consider the case where you want some text to be included in the Authentic View output but not in the HTML output. A condition could be created with a branch for processing Authentic View output, and no branch for HTML output.

Note: Conditions for specific output can be placed around individual parts or components of the document, thus providing considerable flexibility in the way the different output documents are structured.

Creating conditions for specific output

To create conditions for specific output, do the following:

1. In Design View, select the component (or highlight the document part) which you wish to create differently for different output formats.
2. Right-click, and, from the context menu that pops up, select **Condition | Insert Output-Based Condition**. This inserts the output condition with two branches, each having the same content (the selected component). Each branch represents a single output (Authentic View or HTML). To determine which branch represents which output, mouseover the branch tag or check the XPath expression of the selected branch (in the Properties entry helper, in the Condition Branch entry, click the Edit button).
3. Within each branch, define the required processing. If you wish not to have any processing for a particular output format, then delete the branch for that format (select the branch and press **Delete**, or select the branch and in the (right-click) context menu select **Condition | Delete Branch**).

Note: The output-based condition can also be created first and (static and/or dynamic) content for each branch inserted later. First insert the output-based condition at a cursor insertion point in the design. Then within the respective branches, insert the required static and/or dynamic content.

Editing the branches of an Output-Based Condition

The XPath expression of a branch of an output-based condition is `$SV_OutputFormat = 'format'`, where `format` is one of the values: `Authentic` or `HTML`. You can edit the XPath expression of a condition branch (in the Properties entry helper, in the Condition Branch entry, click the Edit button). For example, you could combine the Authentic View and HTML output formats in one condition branch (using the XPath expression: `$SV_OutputFormat = 'Authentic' or $SV_OutputFormat = 'HTML'`).

You can also (i) delete one or more branches; (b) create an `otherwise` branch in a condition; and (c) move the branches up or down relative to each other, thus changing the relative priority of the branches. For information on how to carry out these actions, see Setting Up the Conditions and Editing Conditions.

Using the `$SV_OutputFormat` parameter

In the XSLT file generated for each output, `$SV_OutputFormat` is created as a global parameter and assigned the value appropriate to that output format (that is, `Authentic` or `HTML`). This parameter can be overridden by passing another value for it to the processor at runtime. This could be useful, if, for example, you wish to create two alternative HTML output options, one of which will be selected at runtime. You could then create condition branches `$SV_OutputFormat = 'HTML-1'` and `$SV_OutputFormat = 'HTML-2'`. At runtime you could pass the required parameter value (`HTML-1` or `HTML-2`) to the processor. For information about how to use StyleVision from the command line, see Command Line Interface: StyleVisionBatch.

Conditions and Auto-Calculations

When using Conditions and Auto-Calculations together, there are a few issues to bear in mind. The two most fundamental points to bear in mind are:

- Only Auto-Calculations **in visible conditions**—that is the branch selected as true—are evaluated.
- Auto-Calculations are evaluated before Conditions.

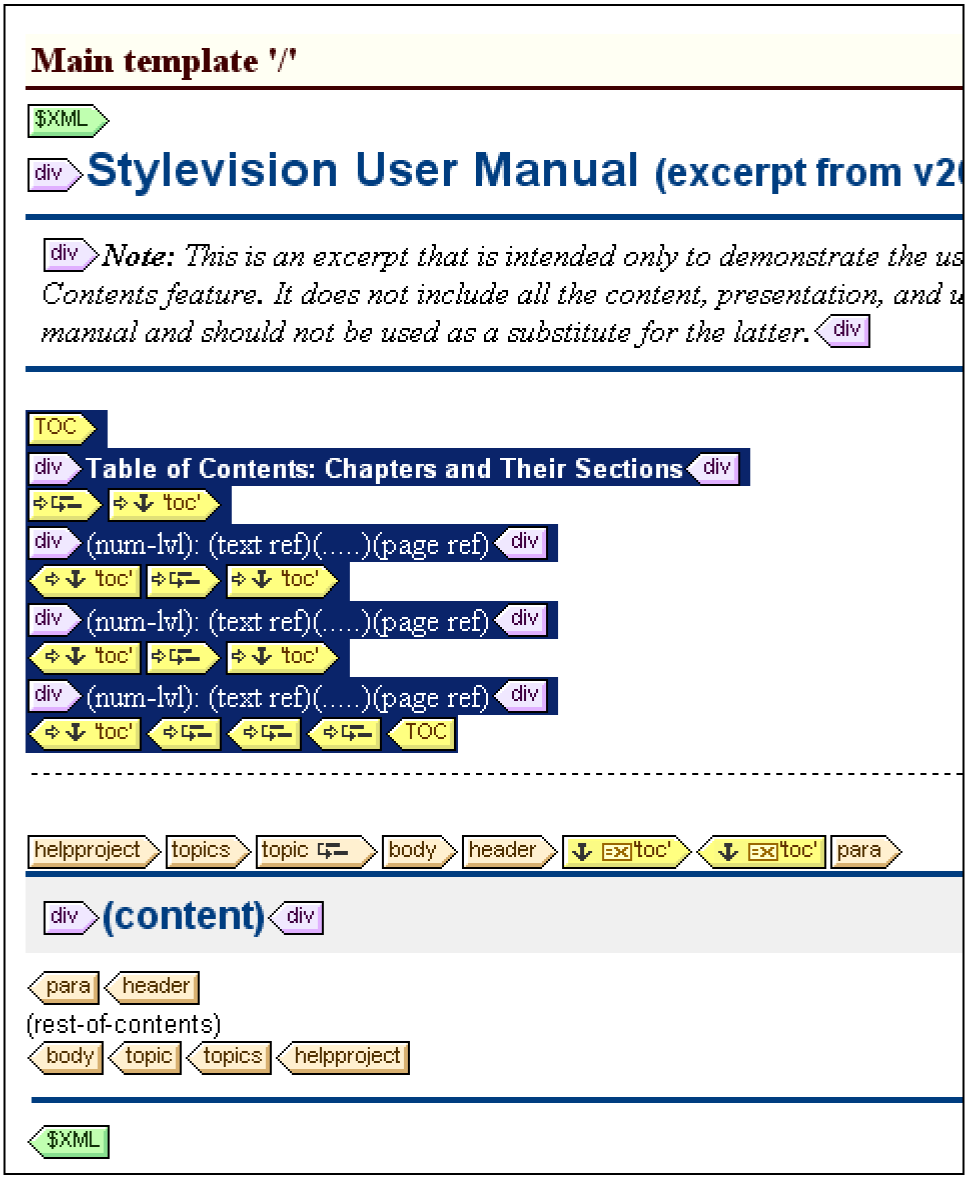
Here are a few guidelines that summarize these issues.

1. If an Auto-Calculation updates a node, and if that node is involved in a Condition (either by being in the XPath expression of a branch or in the content of a conditional template), then keep the Auto-Calculation outside the condition if possible. This ensures that the Auto-Calculation is always visible—no matter what branch of the condition is visible—and that the node will always be updated when the Auto-Calculation is triggered. If the Auto-Calculation were inside a branch that is not visible, then it would not be triggered and the node not updated.
2. If an Auto-Calculation must be placed inside a condition, ensure (i) that it is placed in every branch of the condition, and (ii) that the various branches of the condition cover all possible conditions. There should be no eventuality that is not covered by a condition in the Conditional Template; otherwise there is a risk (if the Auto-Calculation is not in any visible template) that the Auto-Calculation might not be triggered.
3. If you require different Auto-Calculations for different conditions, ensure that all possible eventualities for every Auto-Calculation are covered.
4. Remember that the order in which conditions are defined in a conditional template is significant. The first condition to evaluate to true is executed. The `otherwise` condition is a convenient catch-all for non-specific eventualities.

7.10 Table of Contents (TOC)

If you have selected **XSLT 2.0** (not XSLT 1.0) as the XSLT version of your SPS, you can create a table of contents (TOC) at any location in the design. The mechanism for creating the TOC consists of two parts, which are described in the sub-sections of this section:

- The items from the design that are to be included in the TOC are marked in the design. These items can be static content or dynamic content. In the bottom half of the screenshot below, yellow TOC bookmark tags 'toc' 'toc' within the `para` tag mark the `para` item for inclusion in the TOC. (In the design displayed in the screenshot, additional TOC bookmarks are in a global template that is not shown.)
- A template is created for the TOC (*highlighted in screenshot below*). The TOC template contains the design of the TOC; it can be located anywhere in the design. In the example shown in the screenshot below, the TOC template is located near the top of the document.



Main template '/'

Stylevision User Manual (excerpt from v2

Note: This is an excerpt that is intended only to demonstrate the us Contents feature. It does not include all the content, presentation, and u manual and should not be used as a substitute for the latter.

Table of Contents: Chapters and Their Sections

(num-lvl): (text ref)(.....)(page ref)

(num-lvl): (text ref)(.....)(page ref)

(num-lvl): (text ref)(.....)(page ref)

(content)

(rest-of-contents)

Either of these two parts can be created first, or bot parts can be created concomitantly.

The TOC is displayed in Authentic View and in the HTML, RTF, and PDF output. You should also note these features: TOCs can be created with a flat or a hierarchical structure (with corresponding numbering), and multiple TOCs can be created within a design. As a result, a stylesheet designer can create a document with, say, one (hierarchical) TOC at the book level and others (also hierarchical) at the chapter level, plus (flat) lists of figures and tables.

Procedure for creating TOCs

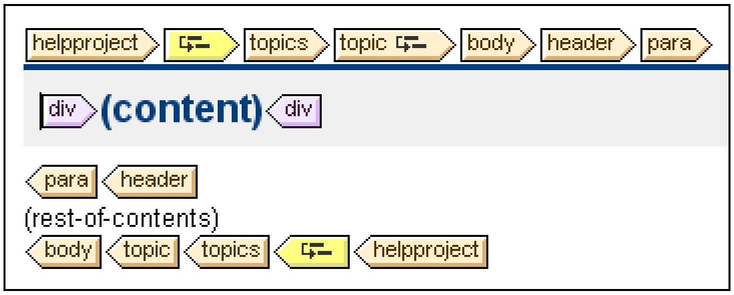
Given below is one step-by-step way of creating a TOC, in which items are first marked for inclusion, and the TOC template is constructed subsequently. (Alternatively, you can create the TOC template first, and then mark items for inclusion; or you can create the TOC template and select items for inclusion in parallel.)

1. Make sure that XSLT 2.0 is the selected XSLT version.
2. *Structure the document in levels*. If the TOC is to have multiple levels, structure the design in a hierarchy of nested levels. If the TOC is to have a flat structure (that is, one level only), then create at least one level that will enclose the TOC bookmarks.
3. *Create one or more TOC bookmarks within each level in the document design.* The TOC bookmarks identify the components within each level that are to appear in the TOC.
4. *Create a TOC template*. The TOC template should have the required number of TOC reference levels (reflevels). In the case of a multi-level TOC, the reflevels in the TOC template should be nested (*see screenshot above*).
5. *Create TOCrefs*. In the TOC template, set up a TOCref for each level. Each TOCref will reference, by name, the required TOC bookmarks within that level in the document; alternatively, the TOCref may reference TOC bookmarks in other levels.
6. *Format the TOC items*. Each TOC item can contain item numbering (including hierarchical), the TOC item text, a leader, and, for paged media, a page number. Each TOC item and its parts can be formatted as required. Note that you can include numbering not only in the TOC template, but also within a TOC bookmark in the main body of the document.

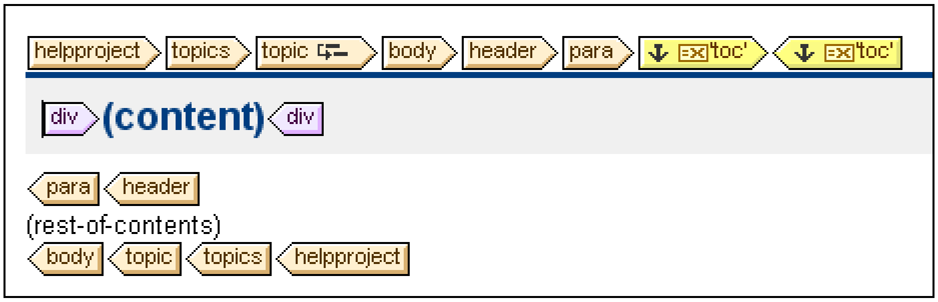
Marking Items for TOC Inclusion

Marking an item in the design for inclusion in a TOC consists of two steps, which can be done in any order:

1. Structuring the design document in a hierarchy of nested levels. A level is created in the design either on a template or around a design component. In the screenshot below, a level has been created on the `topic` template topic.

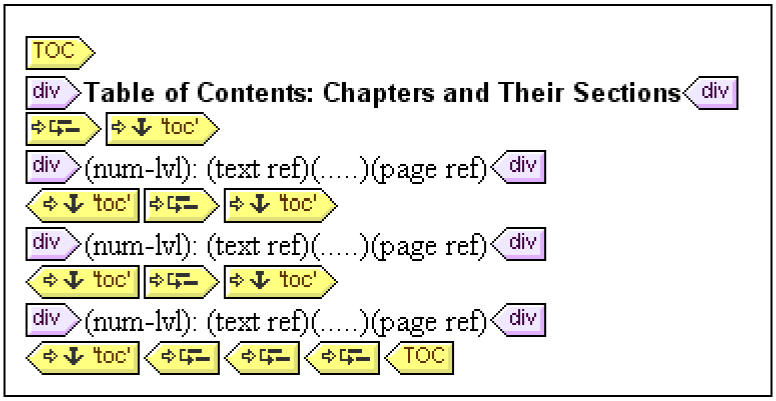
   When a level is created on a template, this is indicated by the level icon inside the start tag of the template. For example, topic. When a level is created around a component it is indicated by level tags. In the screenshot above, the `topics` template component is enclosed by a level. The difference between the two ways of marking levels is explained in the section Structuring the Design in Levels. When the TOC template is created, it must be structured in a hierarchy of levels, with the levels in the TOC template corresponding to the levels you have created in the design. Even for TOCs with a flat structure (one level), the design must have a corresponding level.
2. Creating a TOC bookmark in the design with a name and TOC-item text. The TOC bookmark can either enclose or not enclose a design component; in the latter case it is empty. In the screenshot below, the TOC bookmark does not enclose a design component.

   The TOC bookmark serves as an anchor in the document. In the screenshot above, the TOC bookmark (and anchor) is located at the start of `para` element instances. The TOC bookmark has two attributes: (i) a name that will be used to reference the TOC bookmark when creating the TOC item in the TOC template, and (ii) a text string that will be used as the text of the corresponding TOC item. How these two attributes are assigned is described in the section, Creating TOC Bookmarks.

How marked items are referenced in the TOC template
The TOC template is structured in nested levels (called reference levels (reflevels) to differentiate them from the levels created in the main body of the design template). Within each reflevel, a TOC reference (TOCref) 'toc' is inserted (*see screenshot below*). The TOCref within a level references TOC bookmarks using the TOC bookmark's name. Each TOC bookmark with that name and which is within the corresponding level in the XML document will

be created as a TOC item at this level in the TOC (when the scope of the TOCref is specified to be the current level). For example, the TOCref indicated with the tag ➪ ↓ 'chapters' references all TOC bookmarks named `chapters` in the corresponding level in the XML document (when the scope of the TOCref has been set to `current`). The text attribute of the respective instantiated TOC bookmark will be output as the text of the TOC item.



```
TOC
div Table of Contents: Chapters and Their Sections div
➪ ↓ 'toc'
div (num-lvl): (text ref)(.....)(page ref) div
➪ ↓ 'toc'  ➪ ↓ 'toc'
div (num-lvl): (text ref)(.....)(page ref) div
➪ ↓ 'toc'  ➪ ↓ 'toc'
div (num-lvl): (text ref)(.....)(page ref) div
➪ ↓ 'toc'  TOC
```

In the screenshot above of a TOC template, there are three nested reflevels, within each of which is a TOCref that contains the template for the TOC item of that level. For example, in the first level, there is a TOCref that references TOC bookmarks that have a name of `toc` ➪ ↓ 'toc'. As a result, all TOC bookmarks in the first level (as structured in the design) and named `toc` will be accessed for output at this level in the TOC.

In the sub-sections of this section, we describe: (i) how the design is structured into levels, and (ii) how bookmarks are created. How the TOC template is created is described in the section, Creating the TOC Template.

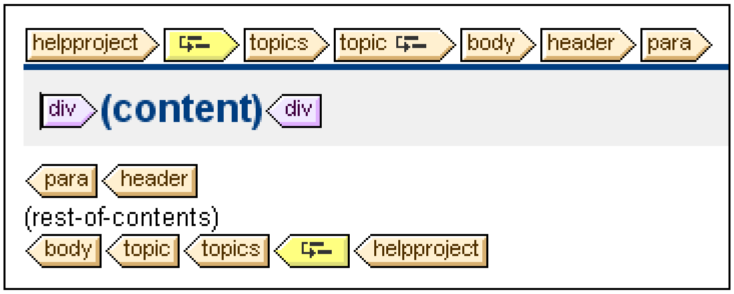
Structuring the Design in Levels

The hierarchical structure you wish to design for the TOC is specified as a set of **nested levels**. As such it is a hierarchical structure which, although related to the XML document structure, is separate from it. This structure is specified in the SPS document design. The TOC template that you construct will use a structure corresponding to this hierarchical structure. In the case of a TOC with a flat structure (one level only), the design document must have at least one level; the flat TOC can then be created for any level in the document design.

Levels can be created in the main template, in global templates, or in a combination of main template and global templates. The important thing to note is that wherever created, these levels must together define a hierarchical structure for the output of the SPS.

Creating levels

Each level is created separately. In the design document, levels can be created on a template or around a component. In the screenshot below, one level has been created on the `topic` template (indicated by topic ⊏) and another around the `topics` element (indicated by ⊏▷ ◁⊏). The essential difference between these two ways of creating levels is that the enclose-within-a-level option enables levels to be created around components other than templates.

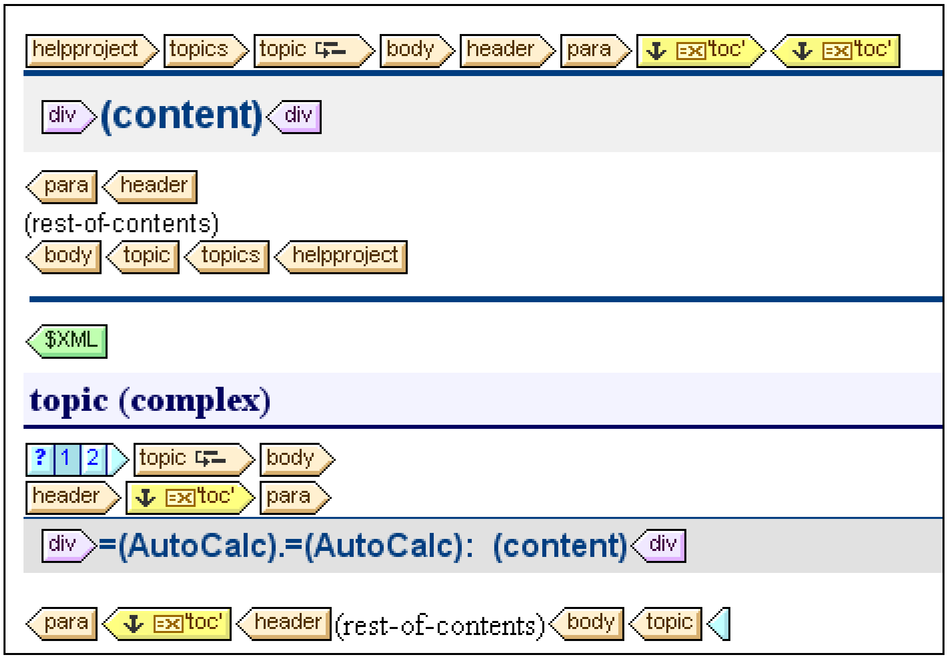


To create a level, do the following:

1. Select the component (template or other).
2. Right-click, and from the context menu select **Template Serves As Level** (enabled when a template is selected) or **Enclose With | Level**. Both these options are also available in the **Insert | Insert Table of Contents** menu: **Level** or **Template Serves as Level**.

Levels in global templates

Levels can also be set in global templates. In these cases, care must be taken to ensure that the levels created in various global templates, as well as those in the main template, **together** define a hierarchical structure when the SPS is executed. The screenshot below shows two levels, one in the main template (on the `topic` template) and one in the global template for `topic` (on the `topic` template).



In the content model represented by the screenshot above, `topic` is a recursive element, that is, a `topic` element can itself contain a `topic` element. In the main template (the end of which is indicated by the $XML tag), a level has been set for the first level of `topic` topic. The `rest-of-contents` instruction in the main template specifies that templates will be applied for all child elements of `topic/body` except `header`. This means that the global template for `topic` children of `topic/body` will be processed. In the global template for `topic`, a level has been set on the `topic` template (indicated by topic). This second level of the TOC hierarchy, which occurs on the second level of `topic` elements, is nested within the first level of the TOC hierarchy. Since this global template also has a `rest-of-contents` instruction, the global template for `topic` will be applied to all recursive `topic` elements, thus creating additional nested levels in the TOC hierarchy.

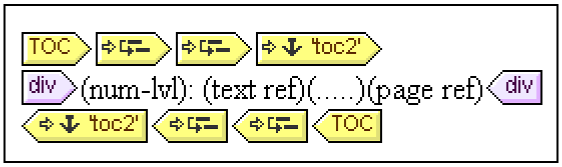
As a designer, you should be aware of the number of levels created in the design, because when the TOC template is constructed, you will need to explicitly specify how TOC items for each level will be selected and formatted.

Levels in a flat TOC hierarchy

In a flat TOC hierarchy, the TOC items will be output at a single level; the outline of the document in the TOC will be a simple list of items. In the TOC template, the items to be listed are specified in the usual way in the design document: by their name and the level in which they occur. Therefore, the document design must contain at least one level, and this level must contain all the required TOC bookmarks.

If the design contains more than one level, and the flat TOC is required, say, for items in the second level, then the TOC template could have two reflevels with a TOCref within the second level (*screenshot below*).

For example, consider the design document shown in the screenshot above: It has one level on the `topic` template in the main template and sub-levels on the `topic` template in the global template. The TOC template shown in the screenshot below will produce a flat TOC of the second-level topic headers.



This is because the TOCref in the TOC template references TOC bookmarks named `toc` that are within the second level. Notice that in the TOC template the TOCref item is created within the second reflevel of the TOC template. Since only one level is output (there is no output for the first reflevel), the resulting TOC will be flat.

Note: Alternatively, the `scope` attribute of TOCrefs can be used to specify what level/s in the design document should be looked up for bookmarks of a given name.

Creating TOC Bookmarks

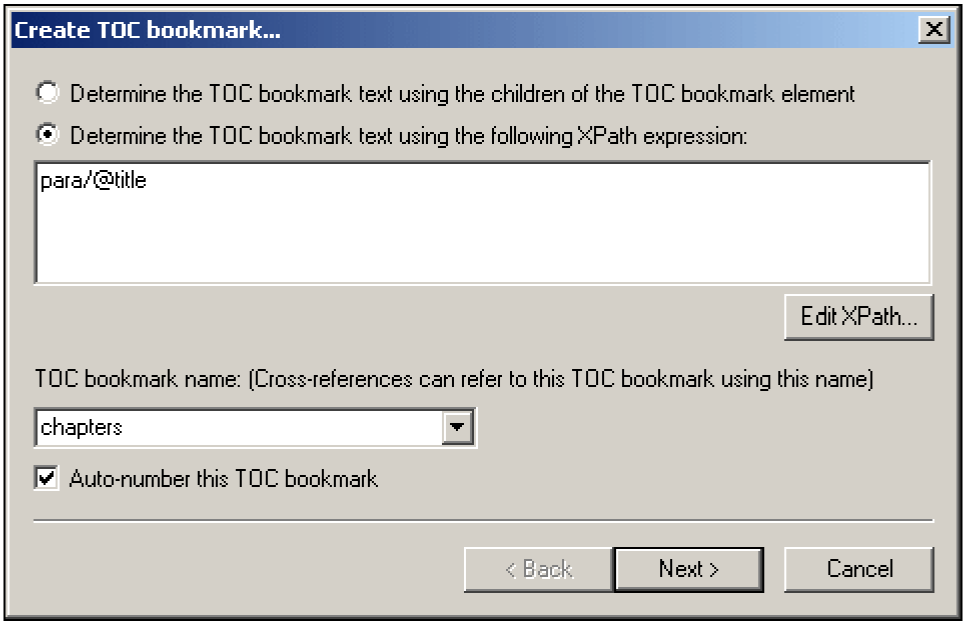
TOC bookmarks are created within a TOC level in the document design. They can be created in the main template and/or in global templates. A TOC bookmark serves two purposes:

- It marks a (static or dynamic) component in the design with a static name you assign. It can either enclose or not enclose a design component; in the latter case it is empty. In the output, the TOC bookmark is instantiated as an anchor identified by a name.
- It defines the text string that will be used as the text for the TOC item/s. This text string can be the content of child elements of the node where the marker is located, or it can be the output of an XPath expression.

You can create the TOC bookmark in two ways: (i) by using the Create Marker Wizard, which enables you to specify the TOC bookmark's name; its text entry; whether auto-numbering should be used; and the level within which it appears; and (ii) by inserting an empty TOC bookmark, the properties of which will be defined subsequently.

Creating the TOC bookmark with the Create Marker Wizard
To create a TOC bookmark using the TOC Bookmark Wizard, do the following:

1. Place the cursor at the point in the design document where you wish to insert the TOC bookmark, or select the design component around which you wish to insert the TOC bookmark.
2. From the **Insert** menu, select **Insert Table of Contents | TOC Bookmark (Wizard)**. This pops up the Create Marker Wizard (*screenshot below*).



○ Determine the TOC bookmark text using the children of the TOC bookmark element

● Determine the TOC bookmark text using the following XPath expression:

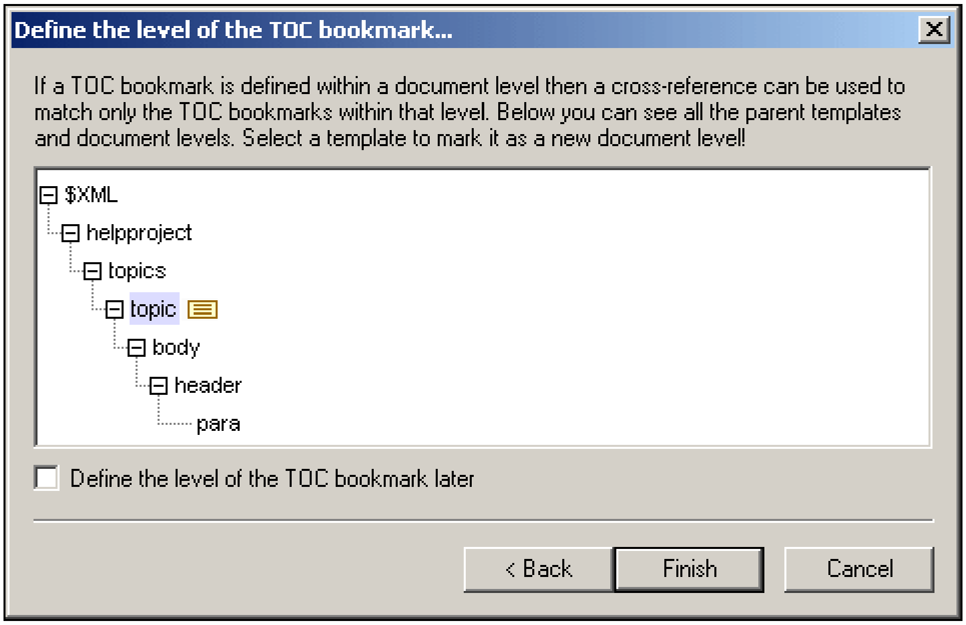
para/@title

TOC bookmark name: (Cross-references can refer to this TOC bookmark using this name)

chapters

☑ Auto-number this TOC bookmark

1. In the wizard's first screen (*screenshot above*) you: (i) define the text entry for the TOC item; (ii) set the TOC bookmark (or marker) name; and (iii) specify whether this marker should be numbered in the output. For the text entry you can select whether the text of child elements should be used, or the result of an XPath expression. For the name of the marker, you can enter text directly or select from a dropdown list containing the names of already specified marker names. When you are done, click **Next**.
2. In the wizard's second screen (*screenshot below*), you can select the level within which the TOC bookmark is to be inserted.



Define the level of the TOC bookmark...

If a TOC bookmark is defined within a document level then a cross-reference can be used to match only the TOC bookmarks within that level. Below you can see all the parent templates and document levels. Select a template to mark it as a new document level!

```
⊟ $XML
  ⊟ helpproject
    ⊟ topics
      ⊟ topic
        ⊟ body
          ⊟ header
              para
```

☐ Define the level of the TOC bookmark later

Ancestor templates on which levels are assigned are indicated with a level icon (in the screenshot above, the `topic` template has a level). Select a template-level within which the TOC bookmark is to be created. If a level already exists for this template, the TOC bookmark will be created within this level, otherwise a new level will be created on the selected template. Alternatively, you can choose to define the level later by checking the Define Level Later check box. When you are done, click **Finish**.

Creating a TOC bookmark

To create a TOC bookmark without attributes, do the following:

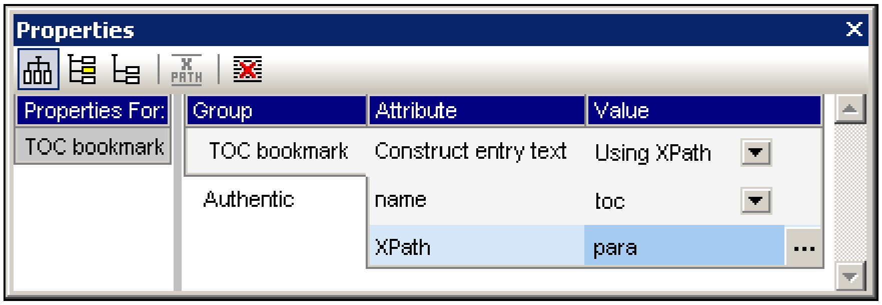
1. Place the cursor at the point in the design document where you wish to insert the TOC bookmark, or select the design component around which you wish to insert the TOC bookmark.
2. From the **Insert** menu, select **Insert Table of Contents | TOC Bookmark**. A TOC bookmark is inserted. This TOC bookmark has neither a name nor a text entry. These can be defined subsequently using the Edit commands.

Inserting hierarchical or sequential numbering for a component

Hierarchical or sequential numbering can be inserted within a TOC bookmark's tags. Right-click at the location where you wish to insert the numbering, then select **Insert Table Of Contents | Hierarchical / Sequential Numbering**. Since numbering can only be inserted at locations within a TOC bookmark, it is better, for numbering purposes, that a TOC bookmark be created around a component rather than be empty. This would allow greater layout flexibility in the placement of the numbering.

Editing the name and text entry of a TOC bookmark

The name and text entry of the TOC bookmark can be edited in the Properties window (*screenshot below*). To edit these properties, select the TOC bookmark, and either directly edit the property in the Property window or right-click the TOC bookmark and select the property you wish to edit.



Properties

Properties For:
TOC bookmark

Group	Attribute	Value
TOC bookmark	Construct entry text	Using XPath
Authentic	name	toc
	XPath	para

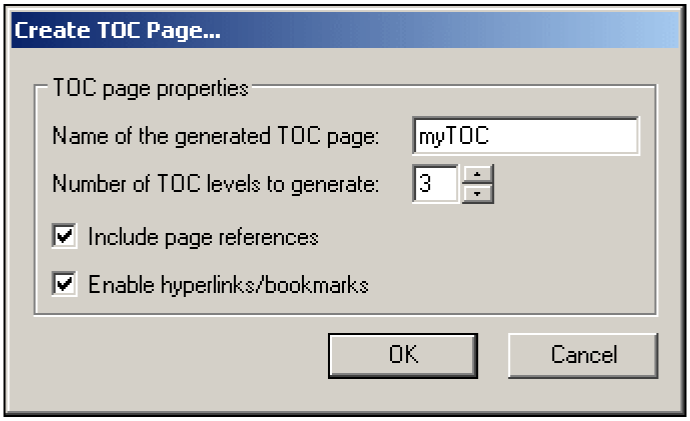
The TOC bookmark has three properties: (i) a toggle to specify whether the text entry come from child elements or from an XPath expression; (ii) the name of the TOC bookmark; and (iii) the XPath expression for the text entry. Note that the XPath expression will not be used unless the Construct Entry Text property specifies that the XPath expression is to be used.

Creating the TOC Template

The TOC template is the template that produces the table of contents in the output. It can be created anywhere within the SPS design, and multiple TOC templates can be created in a single SPS design.

The steps to create a TOC template are as follows:

1. Place the cursor at the location where the TOC template is to be inserted.
2. Click the menu command **Insert | Insert Table of Contents | Table of Contents**. This pops up the Create TOC Page dialog (*screenshot below*). (Alternatively, this command can be accessed via the context menu, which appears when you right-click.)



Create TOC Page...

TOC page properties

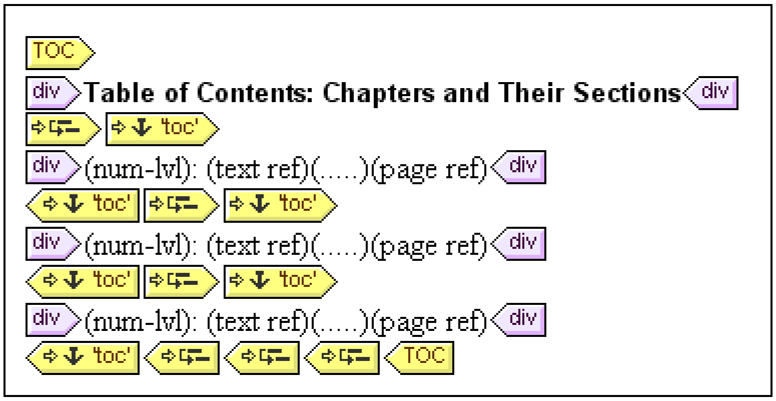
Name of the generated TOC page: myTOC

Number of TOC levels to generate: 3

☑ Include page references

☑ Enable hyperlinks/bookmarks

3. Enter the information requested in the dialog: (i) The name of the generated TOC page is the (TOCref) name that will be used to reference the TOC bookmarks in the design document. If you select multiple levels for the TOC (next option), the same TOCref name will be used in all levels (though individual TOCref names can be edited subsequently). (ii) The number of TOC reflevels specifies how many levels the TOC is to have. (iii) For printed media, the option to output page references (i.e. page numbers) is available. (iv) The text entries in the TOC can be used as links to the TOC bookmarks.
4. Click **OK** to finish. The TOC template is created with the specified number of reflevels (*screenshot below; the formatting of the TOC template has been modified from that which is created initially*).



TOC
div
Table of Contents: Chapters and Their Sections
div
'toc'
div (num-lvl): (text ref)(.....)(page ref) div
'toc'
'toc'
div (num-lvl): (text ref)(.....)(page ref) div
'toc'
'toc'
div (num-lvl): (text ref)(.....)(page ref) div
'toc'
TOC

Within each reflevel is a TOCref having a name that identifies TOC bookmarks that are to be the TOC items for that TOC template reflevel. Within each TOCref is a default template for the TOC item, which you can edit at any time.

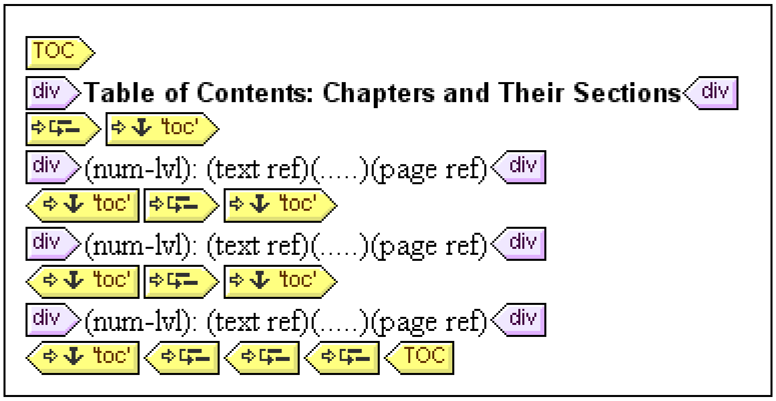
Editing the TOC template

The following editing options are available:

- The TOC template can be dragged to another location in the SPS. Note, however, that a change of context node could affect XPath expressions within the TOC template.
- Reflevels can be added to or deleted from the structure of the TOC template.
- The properties of individual TOC references (TOCrefs) can be edited. The name and scope of a TOCref can be changed, and you can choose whether the TOC item corresponding to the TOCref is created as a hyperlink or not.
- TOCrefs can be added to or deleted from any reflevel in the TOC template.
- The TOC item within a TOCref can be formatted with CSS properties using the standard StyleVision mechanisms.
- Standard SPS features (such as images, Auto-Calculations, and block-formatting components can be inserted anywhere in the TOC template.

Reflevels in the TOC Template

The TOC template is structured in **level references (or reflevels)**; *see screenshot below*. These levels are initially created when the TOC template is created, and the number of reflevels are the number you specify in the Create TOC Page dialog.


TOC
div
Table of Contents: Chapters and Their Sections
div
'toc'
div
(num-lvl): (text ref)(.....)(page ref)
div
'toc'
'toc'
div
(num-lvl): (text ref)(.....)(page ref)
div
'toc'
'toc'
div
(num-lvl): (text ref)(.....)(page ref)
div
'toc'
TOC

Notice that the reflevels are nested. For the purposes of the TOC design there is a one-to-one correspondence between the reflevels in the TOC template and the levels in the SPS design. Thus, the first reflevel of the TOC template corresponds to the first level in the SPS design, the second reflevel in the TOC template to the second level in the SPS design, and so on. The TOCrefs within a given reflevel of the TOC template identify TOC bookmarks within a specified scope in the SPS design.

Inserting and removing reflevels

Reflevels can be inserted in or deleted from the TOC template after the TOC template has been created.

To insert a reflevel, select the content in the TOC template around which a reflevel is to be created, then select **Insert | Insert Table of Contents | Level Reference**. Alternatively, from the context menu, select **Enclose With | Level Reference**. A reflevel can also be inserted at a cursor insertion point in the TOC template.

To remove a reflevel from the TOC template, select the reflevel to be removed and either press the **Delete** key or select **Remove** from the context menu. Note that only the reflevel will be removed—not its contents.

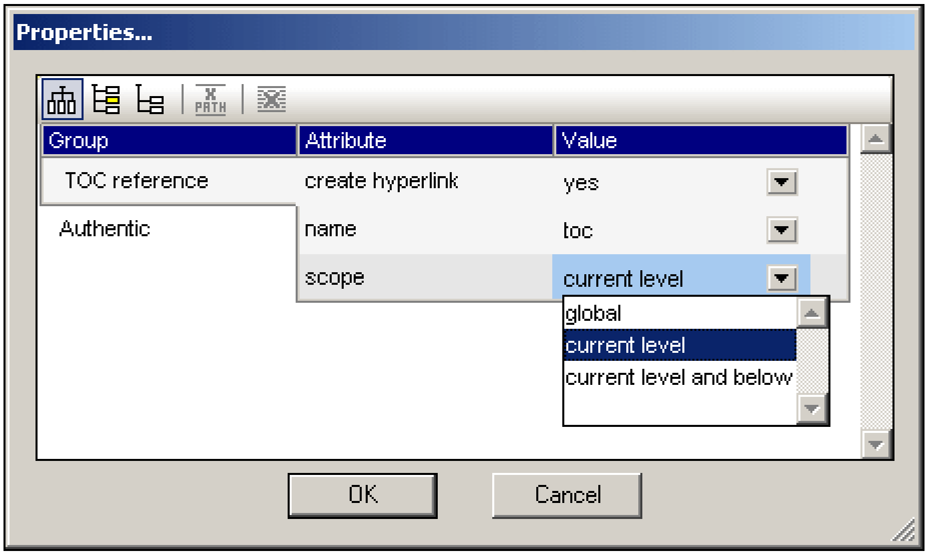
TOC References: Name, Scope, Hyperlink

TOC references (TOCrefs) occur within level references (reflevels) and have three properties:

- A ***name***, which identifies TOC bookmarks of the same name that occur within the specified scope as the items to be included at that level of the TOC.
- A ***scope***, which specifies to which corresponding levels in the SPS design the TOCref applies. Three options are available: global, current level, current level and descendant levels.
- A ***hyperlink*** property which can be toggled between `yes` and `no` to specify whether the corresponding TOC items are created as hyperlinks or not.

To insert a TOCref, place the cursor within a reflevel and, from the **Insert** menu or context menu, select Insert **Table of Contents | TOC Reference**.

To edit a TOCref property, right-click the TOCref tag in the TOC template and select the property you wish to edit (Create Hyperlink, Edit Name, or Edit Scope). This pops up the Properties window with the specified property selected for editing (*screenshot below*).



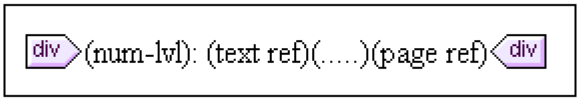
Group	Attribute	Value
TOC reference	create hyperlink	yes
Authentic	name	toc
	scope	current level

Alternatively, with the TOCref tag selected, go directly to the required property in the Properties window (*TOC reference* group of properties).

Formatting TOC Items

The TOC item can contain up to four standard components, plus optional user-specified content. The four standard components are (*see also screenshot below*):

- the text entry of the TOC item, indicated in the TOC template by `(text ref)`
- the leader between the text entry and the page number (for paged media output), indicated by `(.....)`
- the page reference of the TOC item, indicated by `(page ref)`
- hierarchical or sequential numbering, indicated by `(num-lvl)` and `(num-seq)`, respectively



When the TOC template is initially created, the text entry is automatically inserted within TOCrefs. If the Include Page Reference option was selected, then the leader and page reference components are also included. Subsequently, components can be inserted and deleted from the TOC item. To insert a component, place the cursor at the desired insertion point within the TOC item, and in the context menu, select **Insert Table Of Contents | TOC Reference | Text Entry / Leader / Page Reference** or **Insert Table Of Contents | Hierarchical Numbering / Sequential Numbering** as required. To delete a component, select it and press the **Delete** key.

Additionally, you can insert static content (e.g. text) and dynamic content (e.g. Auto-Calculations) within the TOC item.

Formatting the TOC item
The TOC item can be formatted with CSS styles via the Styles entry helper. Individual TOC item components can be separately formatted by selecting the component and assigning it style properties in the Styles entry helper.

Chapter 8

Presentation Procedures

8 Presentation Procedures

In the SPS design, a single set of styling features is defined for components. These styles are converted to the corresponding style markup in the respective outputs (*Authentic View*, *HTML, RTF, and PDF in the Enterprise Edition; Authentic View and HTML in the Professional Edition; HTML in the Standard Edition*).

Styling of SPS components
All styling of SPS components is done using CSS2 principles and syntax. Styles can be defined in external stylesheets, globally for the SPS, and locally on a component. The cascading order of CSS2 applies to the SPS, and provides considerable flexibility in designing styles. How to work with CSS styles is described in detail in the Working with CSS Styles sub-section of this section.

The values of style properties can be entered directly in the Styles or Properties entry helpers, or they can be set via XPath expressions. The benefits of using XPath expressions are: (i) that the property value can taken from an XML file, and (ii) that a property value can be assigned conditionally according to a test contained in the XPath expression.

Additionally, in the SPS design, certain HTML elements are available as markup for SPS components. These predefined formats are passed to the HTML output. The formatting inherent in such markup is therefore also used to provide styling to SPS components. When CSS styles are applied to predefined formats, the CSS styles get priority over the inherent style of the predefined format. Predefined formats are described in the Predefined Formats sub-section of this section.

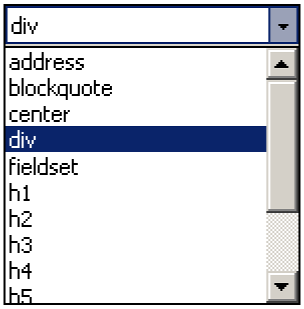
8.1 Predefined Formats

StyleVision provides a number of pre-defined formats, each of which corresponds to an HTML element (*screenshot below*). When you apply a Predefined Format to a component in the Design, that component is marked up as a component having the corresponding HTML semantics. This has two effects:

- Formatting inherent to the selected predefined format is applied.
- The component is contained in the component type, *paragraph*, which makes it available for local styling by component type.

Assigning Predefined Formats

Predefined formats can be assigned by clicking **Insert | Format**, and then the required format, or by selecting the required format from the Format drop-down list in the Toolbar (shown below).



Inherent styles

The predefined formats used in StyleVision have either one or both of the following two styling components:

- a text-styling component
- a spacing component.

For example, the predefined `para ( p)` format has a spacing component only; it puts vertical space before and after the selected component, and does not apply any text styling. On the other hand, the predefined `Heading 1 ( h1)` format has both a text-styling component and a spacing component.

The following styling points about predefined formats should be noted:

- The spacing component of a predefined format applies for any type of SPS component, but the text styling only if it can be applied. For example, if you select an image and apply a predefined format of `Heading 1 ( h1)` to it, then the spacing component will take effect, but the text-styling component will not.
- The text-styling component of predefined formats does not apply to data-entry devices.
- Only one predefined format applies to a component at any given time.

Defining additional styling for a predefined format

Styles additional to the inherent styling can be defined for a predefined format by selecting it and applying a local style via the Styles entry helper.

The Return key and predefined formats

In Authentic View, when the **Return** key is pressed within the contents of an element having a predefined format, the current element instance and its block are terminated, and a new element instance and block are inserted at that point. This property is useful, for example, if you want the Authentic View user to be able to create a new element, say a paragraph-type element, by pressing the `Return` key.

8.2 Working with CSS Styles

The SPS design document is styled with CSS rules. Style rules can be specified:

- In external CSS stylesheets.
- In global stylesheets for the SPS, which can be considered to be defined within the SPS and at its start. (In the HTML output these global styles are defined within the `style` child element of the `head` element.)
- Locally, on individual components of the document. In the HTML output, such rules are defined in the `style` attribute of individual HTML elements.

Each of the above methods of creating styles is described in detail in the sub-sections of this section (*links above*).

Terminology

A CSS stylesheet consists of one or more style rules. For example:

```
H1 { color: blue }
```

or

```
H1 { color: blue;
      margin-top: 16px; }
```

Each rule has a selector (in the examples above, `H1`) and a declaration (`color: blue)`. The declaration is a list of properties (for example, `color`) with values (`blue`). In StyleVision, CSS styles can be defined in the Styles Entry Helper (local styles) and Style Repository Entry Helper (global styles).

Cascading order

The cascading order of CSS applies. This means that precedence of rules are evaluated on the basis of:

1. ***Origin.*** External stylesheets have lower precedence than global styles, and global styles have lower precedence than local styles. External stylesheets are considered to be imported, and the import order is significant, with the latter of two imported stylesheets having precedence.
2. ***Specificity.*** If two rules apply to the same element, the rule with the more specific selector has precedence.
3. ***Order.*** If two rules have the same origin and specificity, the rule that occurs later in the stylesheet has precedence. Imported stylesheets are considered to come before the rule set of the importing stylesheet.

CSS Support in Internet Explorer

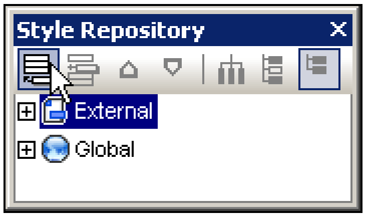
Versions of Internet Explorer (IE) prior to IE 6.0 interpret certain CSS rules differently than IE 6.0 and later. As a designer, it is important to know for which version of IE you will be designing. IE 6.0 and later offers support for both the older and newer interpretations, thus enabling you to use even the older interpretation in the newer versions (IE 6.0 and later). Which interpretation is used by IE 6.0 and later is determined by a switch in the HTML document code. In an SPS, you can specify whether the HTML and Authentic View output documents should be styled according to Internet Explorer's older or newer interpretation. You should then set CSS styles according to the selected interpretation. For more details, see Properties: CSS Support.

Note: For more information about the CSS specification, go to http://www.w3.org/TR/REC-CSS2/.

External CSS Stylesheets

To assign an external CSS stylesheet to the SPS, do the following:

1. In Design View, select the External item in the Style Repository window (*screenshot below*).



2. Click the **Add** button at the top left of the Style Repository toolbar (*see screenshot above*).
3. In the Open dialog that pops up, browse for and select the required CSS file, then click **Open**. The CSS file is added to the External item as part of its tree structure (*see tree listing and screenshot below*).
4. To add an additional external CSS stylesheet, repeat steps 1 to 3. The new CSS stylesheet will be added to the External tree, after all previously added external CSS stylesheets.

Viewing and modifying the tree of external CSS stylesheets

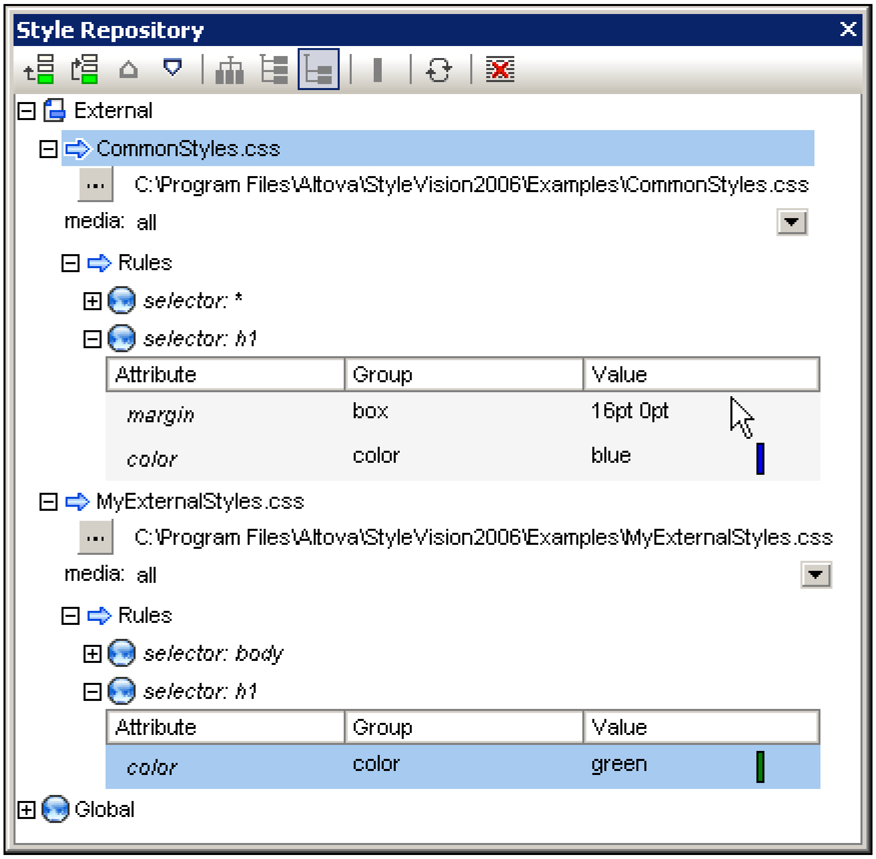
The tree of external CSS stylesheets is structured as follows (*also see screenshot below*):

```
- CSS-1.css
  - Location of file (editable in Style Repository window)
  - Media (can be defined in Style Repository window)
  - Rules (non-editable; must be edited in CSS file)
    - Selector-1
      - Property-1
      - ...
      - Property-N
    - ...
    - Selector-N
+ ...
+ CSS-N.css
```

Each CSS-file-location item can be edited in the Style Repository window; do this by clicking the Edit button ... and selecting the required CSS file. The media to which that particular stylesheet is applicable can also be edited in the Style Repository window; do this by clicking the down arrow to the right of the item and selecting the required media from the dropdown list). The rules defined in the external CSS stylesheet are displayed in the Style Repository window, but cannot be edited. The Stylesheet, Rules, and individual Selector items in the tree can be expanded and collapsed by clicking the + and - symbols to the left of each item (*see screenshot below*).

To delete an external stylesheet, select the stylesheet and click the **Reset** button in the Style Repository toolbar.

Note: Style rules with certain selectors will not be applied to RTF and PDF output. Such rules are commented: `Will be discarded in PDF, RTF.`



Style Repository

⊟ External

⊟ CommonStyles.css

... C:\Program Files\Altova\StyleVision2006\Examples\CommonStyles.css

media: all

⊟ Rules

⊞ *selector: **

⊟ *selector: h1*

Attribute	Group	Value
margin	box	16pt 0pt
color	color	blue

⊟ MyExternalStyles.css

... C:\Program Files\Altova\StyleVision2006\Examples\MyExternalStyles.css

media: all

⊟ Rules

⊞ *selector: body*

⊟ *selector: h1*

Attribute	Group	Value
color	color	green

⊞ Global

Changing the precedence of the external CSS stylesheets
The external CSS stylesheets that are assigned in the Style Repository window will be imported into the HTML output file using the `@import` instruction. In the HTML file, this would look something like this:

```
<html>
    <head>
        <style>
            <!--
            @import url("ExternalCSS-1.css");
            @import url("ExternalCSS-2.css")screen;
            @import url("ExternalCSS-3.css")print;
            -->
        </style>
    </head>
    <body/>
</html>
```

The order in which the files are listed in the HTML file corresponds to the order in which they are listed in the External tree of the Style Repository. To change the order of the CSS stylesheets in the External tree, select the stylesheet for which the precedence has to be changed. Then use the **Move Up** ⌂ and **Move Down** ▽ buttons in the Style Repository toolbar to reposition that stylesheet relative to the other stylesheets in the tree.

Important: What is important to note is that the lowermost stylesheet has the highest import precedence, and that the import precedence decreases with each previous stylesheet in the listing order. The order of import precedence in the listing shown above is: (i) `ExternalCSS-3`.

`css`; (ii) `ExternalCSS-2.css`; (iii) `ExternalCSS-1.css`. When two CSS rules, each in a different stylesheet, address the same node, the rule in the stylesheet with the higher import precedence applies.

Editing the properties of external CSS stylesheets

An external CSS stylesheet can be quickly replaced by another by clicking the **Edit** button ... and browsing for the required stylesheet. The media to which an external CSS stylesheet is to be applied can be selected by pressing the dropdown box of the Media item of an external stylesheet, and there selecting the required media from the list of options.

Defining CSS Styles Globally

Global styles are defined for the entire SPS design in the Style Repository and are listed in the Style Repository under the Global heading. They are passed to Authentic View and the HTML output document as CSS rules. In the HTML document, these CSS rules are written within the /html/head/style element.

In the Style Repository, a global style is a single CSS rule consisting of a selector and CSS properties for that selector. Creating a global style, therefore, consists of two parts:

- Adding a new style and declaring the CSS selector for it.
- Defining CSS properties for the style (or selector).

Supported selectors

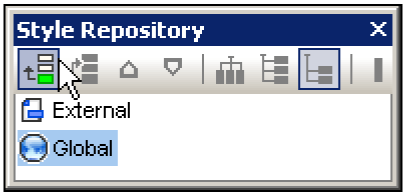
The following selectors are supported:

- *Universal selector:* written as *
- *Type selectors:* element names, such as h1
- *Attribute selectors:* for example, [class=maindoc]
- *Class selectors:* for example, .maindoc
- *ID selectors:* for example, #header

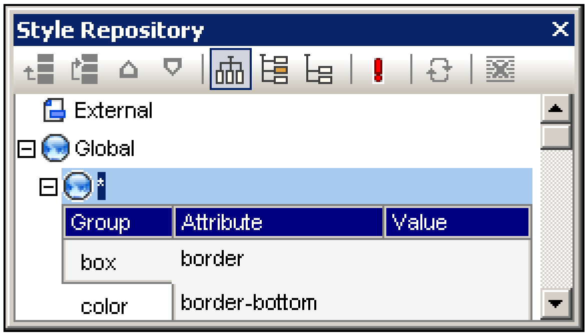
Adding a global style

To add a global style to the SPS design, do the following:

1. In Design View, select the Global item in the Style Repository window (*screenshot below*).



2. Click the **Add** button at the top left of the Style Repository toolbar (*see screenshot above*). A global style is inserted into the Global tree with a * selector (which selects all HTML elements); the universal selector is the default selector for each newly inserted global style.
3. To change the selector from the default universal selector, either: (i) right-click and select an option from the Add Selector submenu, or (ii) click the selector and edit it.



Group	Attribute	Value
box	border	
color	border-bottom	

4. Now set the CSS property values for the selector. How to do this is explained in the section Setting CSS Property Values.
5. To add another global style, repeat steps 1 to 4. The new global style will be added to

the Global tree, after all previously added global styles.

Note:

- Global styles can also be inserted before a selected global style in the Global tree by clicking the **Insert** button in the Style Repository window. The **Add** and **Insert** buttons are also available via the context menu that appears when you right-click a global style or the Global item in the Style Repository window.
- A global style with a selector that is an HTML element can be inserted by right-clicking an item in the Global tree, then selecting **Add Selector | HTML | *HTMLElementName***.

Editing and deleting global styles

Both, a style's selector as well as its properties can be edited in the Style Repository window.

- To edit a selector, double-click the selector name, then place the cursor into the text field, and edit.
- For information about defining and editing a style's property values, see Setting CSS Property Values. (The style properties can be displayed in three possible views. These views and how to switch between them are described in Views of Property Definitions.

To delete a global style, select the style and click the **Reset** button in the Style Repository toolbar.

Changing the precedence of global styles

Global styles that are assigned in the Style Repository window are placed as CSS rules in the `/html/head/style` element. In the HTML file, they would look something like this:

```
<html>
   <head>
      <style>
         <!--
         h1     { color: blue;
                  font-size: 16pt;
                }
         h2     { color: blue;
                  font-size: 14pt;
                }
         .main  { color: green; }
         -->
      </style>
   </head>
   <body/>
</html>
```

The order in which the global styles are listed in Authentic View and the HTML document corresponds to the order in which they are listed in the Global tree of the Style Repository. The order in Authentic View and the HTML document has significance. If two selectors select the same node, then the selector which occurs lower down the list of global styles has precedence. For example, in the HTML document having the partial listing given above, if there were an element `<h1 class="main">`, then two global styles match this element: that with the `h1` selector and that with the `.main` selector. The color property of `.main` selector will apply because it occurs after the `h1` selector in the style listing. The font-size of the `h1` style will, however, apply to the `<h1>` element because there is no selector with a higher precedence that matches the `<h1>` element and has a font-size property.

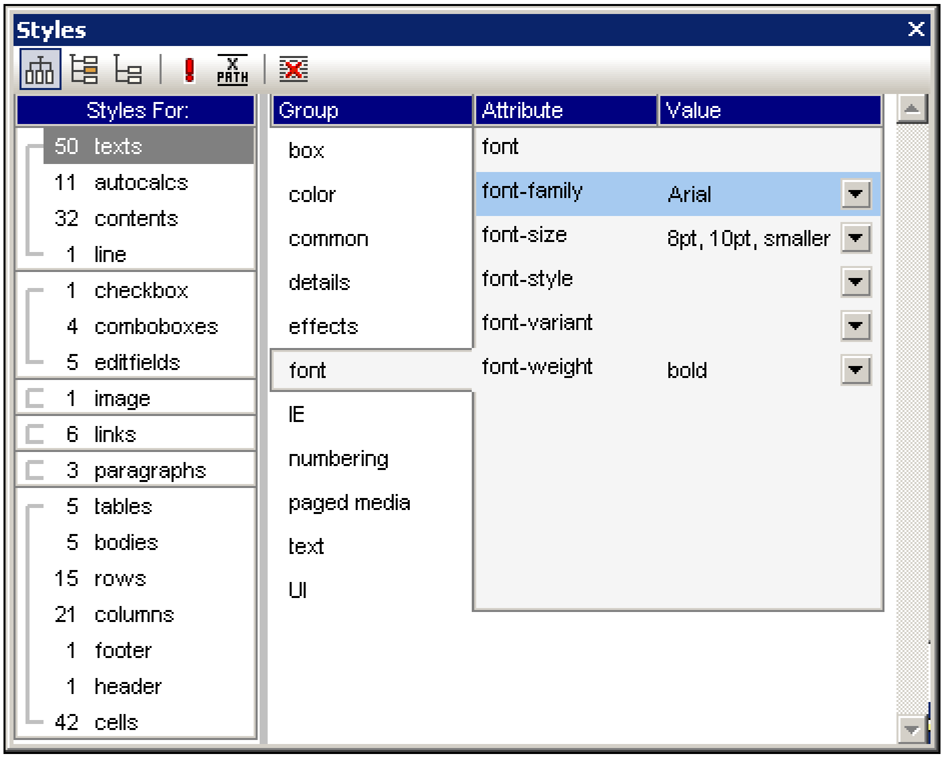
To change the precedence of a global style, select that style and use the **Move Up** and **Move Down** buttons in the Style Repository toolbar to reposition that global style relative to the other global styles in the tree. For example, if the .main global style were moved to a position before the h1 style, then the color property of the h1 style would have precedence over that of the .main style.

Defining CSS Styles Locally

When styles are defined locally, the style rules are defined directly on the component. These local rules have precedence over both global style rules and style rules in external CSS stylesheets that select that component. Locally defined styles are CSS styles and are defined in the Styles entry helper. (This is as opposed to global styles, which are defined in the Style Repository entry helper.)

Defining a style locally consists of two parts:

1. The component or components to be styled are selected in the design (Design View). You can select multiple by keeping the Shift key depressed while selecting components. These components are each of a particular component type. In the selection you make, all components of a single component type are listed together by component type (for example: `50 texts` in the screenshot below).



Attribute	Value
font	
font-family	Arial
font-size	8pt, 10pt, smaller
font-style	
font-variant	
font-weight	bold

2. After making the selection in Design View, you select the component type (in the Styles For) column. If there is more than one component for that component type, then styles will be applied to all these components. How to make a selection for local styling is described in Selecting SPS Components to Style.
3. After selecting the components to style in the Styles For column of the Styles window, the styles for that selection are defined in the Property Definitions column. How to do this is described in the section Setting CSS Property Values.

Selecting SPS Components to Style

Any component in the SPS design (except node tags) can be selected for the definition of a style. Components that can be styled are: (i) a static SPS component such as an Auto-Calculation or a text string; or (ii) a predefined format (represented in the Design View by its start and end tags).

Each SPS component may:

- be of a single component type (for example, a horizontal line component is of the *line* component type; a `(contents)` placeholder is of the *content* component type; a combo box is of the *combobox* component type);
- have structurally mandatory component subtypes (for example, a table component will be of the component type *table*, and will have the mandatory component subtypes *body* , *row*, *column*, and *cell*, and optionally, the *header* and *footer* component subtypes.

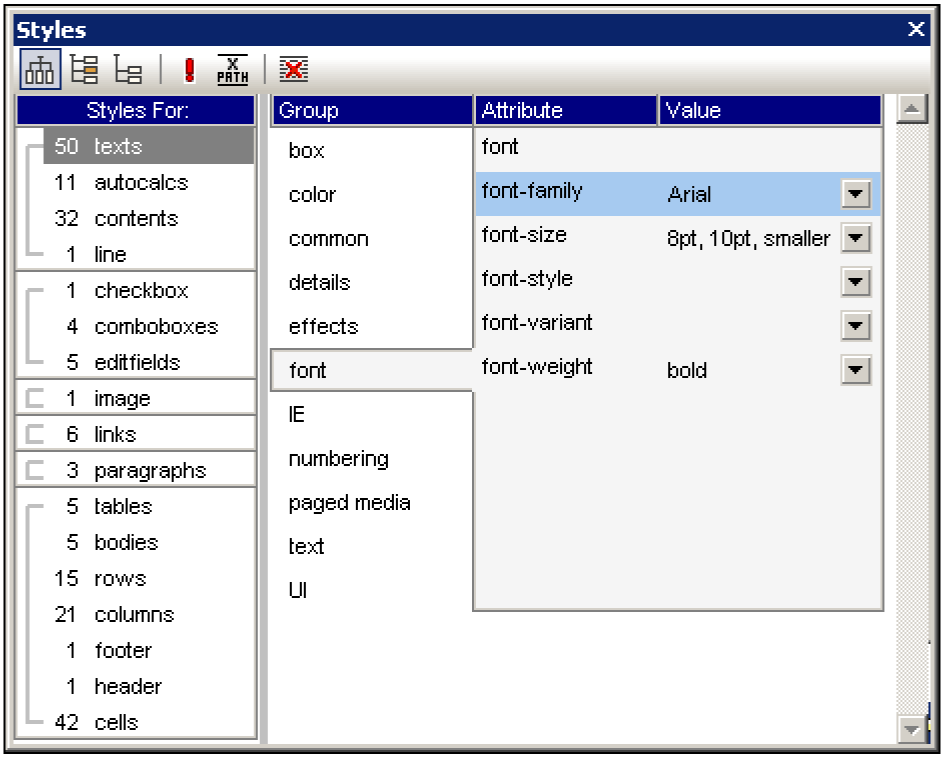
The component or components to style are selected in two steps:

1. Select the SPS component in the design (Design View).
2. Select a component type from the contained component types; this selection is done in the Styles For column of the Styles entry helper.

These two steps are described in detail below.

Selecting the SPS component

When an SPS component is selected in the design (by clicking it), its component type is listed in the Styles For column of the Styles Entry Helper. If multiple components are selected in the design, all components of one component type are listed together in the Styles For column of the Styles entry helper (*screenshot below*).



Styles For:	
50	texts
11	autocalcs
32	contents
1	line
1	checkbox
4	comboboxes
5	editfields
1	image
6	links
3	paragraphs
5	tables
5	bodies
15	rows
21	columns
1	footer
1	header
42	cells

Group:
- box
- color
- common
- details
- effects
- font
- IE
- numbering
- paged media
- text
- UI

Attribute	Value
font	
font-family	Arial
font-size	8pt, 10pt, smaller
font-style	
font-variant	
font-weight	bold

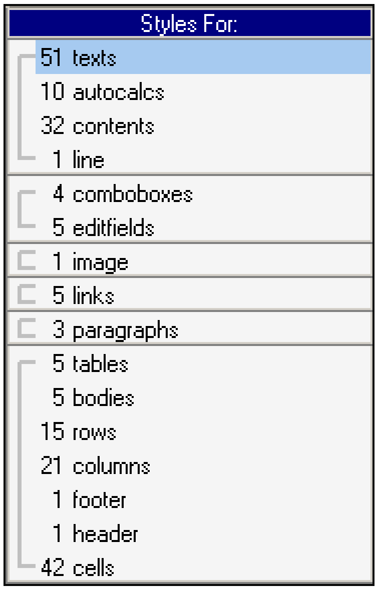
In the Styles For column, the selected component types are organized into the following categories (each category separated from the next by a line):

- ***Textual components.*** These include: static text strings entered directly in the SPS (*texts*); Auto-Calculations (*autocalcs*); dynamic text which is included in the SPS using the `( contents)` placeholder (*contents*); and horizontal lines inserted directly in the SPS (*lines*).
- ***Data-entry devices.*** These include: input fields (*editfields*); multiline input fields (*multiline editfields*);combo-boxes (*comboboxes*); check boxes (*checkboxes*); radio buttons (*radiobuttons*); and buttons (*buttons*). *See Using Data-Entry Devices*.
- ***Images.*** These are images inserted in the SPS via the Insert | Image command.
- ***Bookmarks and links.*** Both bookmarks and hyperlinks are indicated as *links*. *See Bookmarks and Hyperlinks*.
- ***Predefined formats.*** All predefined formats (such as `div`, `p`, `h1`, and `pre`) are indicated as *paragraphs*. *See Predefined Formats*.
- ***Table components.*** These include the structural components of a table from the *table* component type down to the *cell* component type. Each subtype is differentiated and listed separately.

Note: The conditional template and condition components are not listed because they are filters. Not being present in the output, they do not need to be styled.

Selecting the component type for styling

When a component in the SPS design is selected, it is listed by its type in the Styles For column of the Styles Entry Helper. If multiple components are selected, all instances of a single component type within that selection are listed together and can be styled in one go. In the Styles For column, you can select any one of the listed component types and define styling for all instances of this component type. For example, in the screenshot below, the 51 text components have been selected. You can now define styling in the Styles Entry Helper for all the selected 51 instances of static text strings. This selection method is useful if a single style definition is required for all instances of a component type within a component.



Styles For:

51 texts
10 autocalcs
32 contents
1 line

4 comboboxes
5 editfields

1 image

5 links

3 paragraphs

5 tables
5 bodies
15 rows
21 columns
1 footer
1 header
42 cells

After selecting the required component type, you can define the required style.

Note: If a component type instance is inserted into the design after a style has been defined for that component type, then this instance must either be styled separately or the style definition for the component type must be redone with the newly inserted instance included in the selection.

Selecting a single component for styling

To define styling for a single component, click the required component to select it. In the case of static text, placing the cursor anywhere within the text string suffices to select it.

How Styles Are Applied to Components

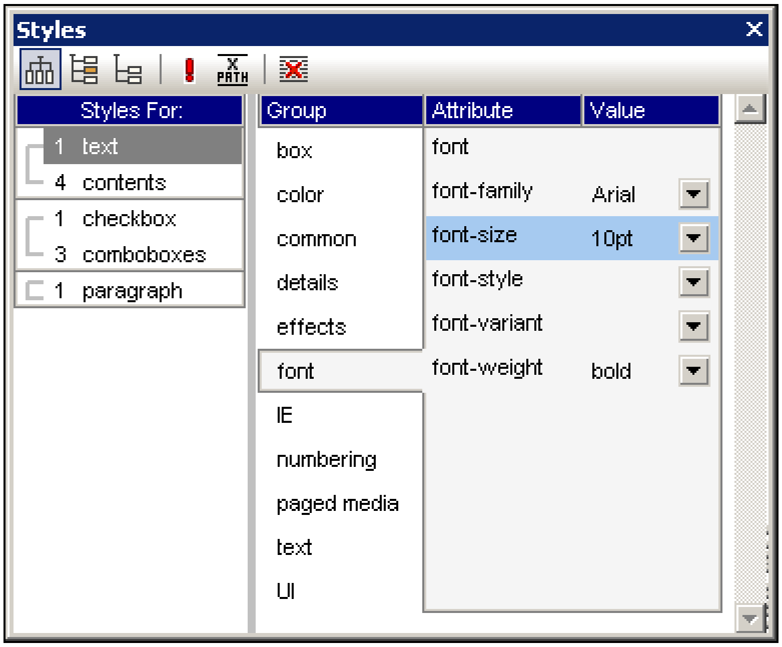
The CSS styles that are applied via the Styles Entry Helper are applied to certain components on the block level and to other components on the inline level. Knowing at which level styles are applied to a component (block or inline) will help you to define styles efficiently. For example, defining vertical margins (the `margin-top` and `margin-bottom` properties) for inline styles will have no effect on the output.

The table below shows how styles are applied to each SPS component type.

Component type	Style application
Static text	Inline
Auto-Calculations	Inline
XML node content created as `(contents)`	Inline
Links	Inline application to content of link. Link itself has no styling.
Predefined formats	Applied to the predefined format element, which are all block elements.
Horizontal lines	Block
XML nodes created as data-entry device	Block
Images	Block
Tables and table sub-components	Block

Setting CSS Property Values

Style properties are defined in the Styles Entry Helper (*screenshot below*) for the selected component or components. The selection is made in two steps. First, the component is selected in the SPS. This causes the descendant component types and any associated predefined formats to appear in the Styles For column of the Styles Entry Helper (*see screenshot below*). Second: In the Styles For column, the descendant component type is selected. In the screenshot below, the *paragraph* component type (the predefined format) is selected. Now style properties can be defined for the predefined format. If, in the screenshot below, the *3 comboboxes* entry had been selected, style properties could have been defined for all three combo boxes in one go.



Attribute	Value
font	
font-family	Arial
font-size	10pt
font-style	
font-variant	
font-weight	bold

Style property groups

The available style properties are CSS properties and are defined in 11 groups:

Style Group	Properties
`box`	Border, margin, and padding settings.
`color`	Color of node content; background properties.
`common`	Includes `class`, `display`, `position`, `float`, `z-index` among others.
`details`	Height, width, and vertical alignment properties.
`effects`	The `clip`, `overflow`, and `visibility` properties.
`font`	Font specifications, such as family, size, style, weight.
`IE`	Internet Explorer-specific properties.
`numbering`	List markers, counters, and quotes.
`paged media`	Settings for page-breaks, orphans, and widows.

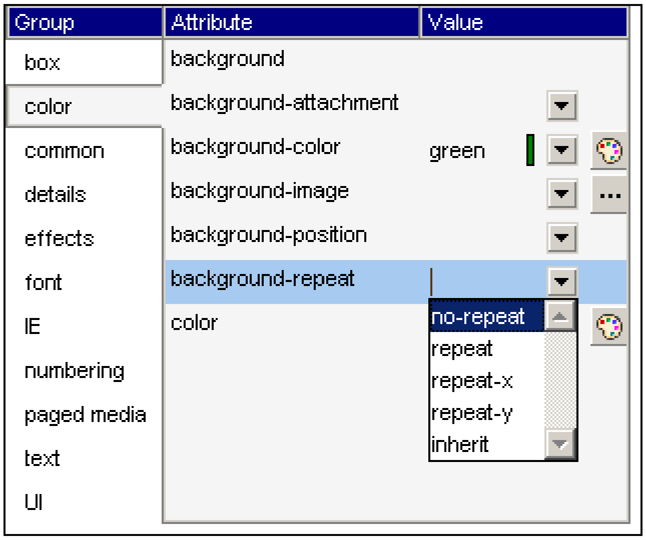
Style Group	Properties
`text`	Text properties such as `text-align`, `text-decoration`, and `text-transform`, as well as other text-related properties such as `letter-spacing` and `word-spacing`.
`UI`	Cursor style setting for user interface.

Note: The `visibility`, `display`, `float`, and `position` properties are not applied in Design View and Authentic View.

Entering property values

Property values can be entered in one, two, or three ways, depending on the property:

- Entered directly in the Value column. To do this, select a property, double-click in its Value column, enter the value using the keyboard, and press **Enter** or click anywhere in the GUI.
- By selecting a value from the dropdown list of the combo box for that property. Click the down arrow of the combo box to drop down the list of property-value options. In the screenshot below, the options for the `background-repeat` property are displayed. Select the required value from the dropdown list.
- By using the Entry Helper at the right-hand side of the Value column for that property. Two types of entry helper are available, and these are available only for properties to which they are relevant: (i) a color palette for selecting colors (in the screenshot below, see `color` and `background-color` properties), and (ii) a dialog for browsing for files (in the screenshot below, see the `background-image` property).



Group	Attribute	Value
box	background	
color	background-attachment	
common	background-color	green
details	background-image	
effects	background-position	
font	background-repeat	
IE	color	
numbering		
paged media		
text		
UI		

Modifying or deleting a property value

If a property value is entered incorrectly or is invalid, both the property and the value are displayed in red. To modify the value, use any of the applicable methods described in the previous section, Entering Property Values.

To delete a property value, double-click in the Value column of the property, delete the value using the **Delete** and/or **Backspace** key, and then press **Enter**.

8.3 Style Properties Via XPath

Styles can be assigned to design components via XPath expressions. This enables property values to be taken from XML data or from the XPath expression itself. Using the `doc()` function of XPath 2.0, nodes in any accessible XML document can be addressed. Not only can style definitions be pulled from XML data; this feature also enables style choices that are conditional upon the structure or content of the XML data. For example, using the `if...else` statement of XPath 2.0, two different background colors can be selected depending on the position of an element in a sequence. Thus, when these elements are presented as rows in a table, the odd-numbered rows can be presented with one background color while the even-numbered rows are presented with another. Also, depending on the content of a node, the presentation can be varied.

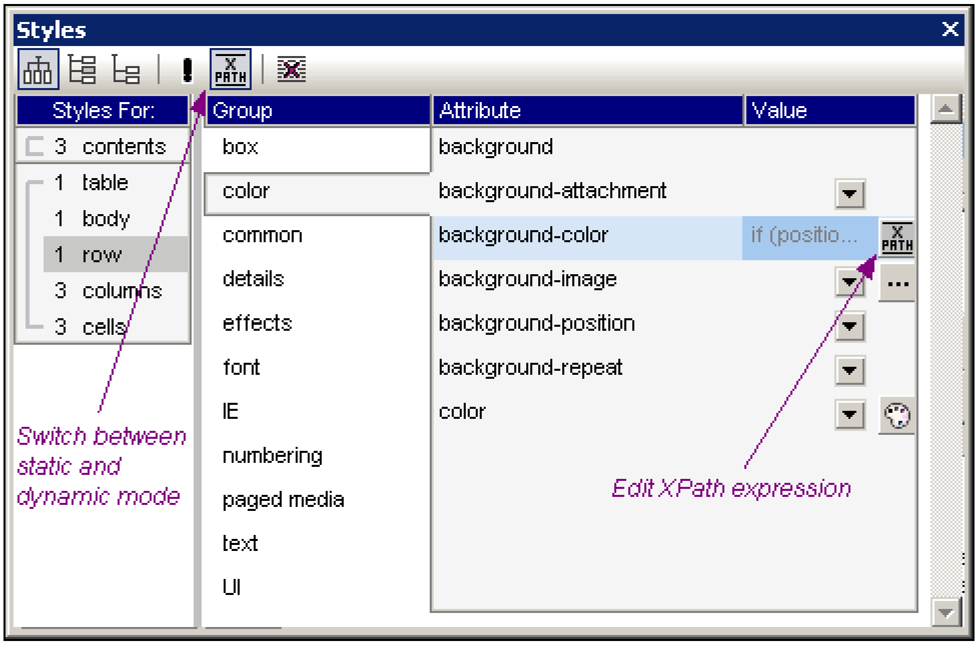
Properties for which XPath expressions are enabled
XPath expressions can be entered for the following styling properties:

- All properties available in the Styles entry helper
- The *Common*, *Event*, and *HTML* groups of properties in the Properties entry helper

Static mode and dynamic (XPath) mode for property values
For those properties where XPath expressions are enabled, two mode are available:

- Static mode, where the value of the property is entered directly in the entry helper. For example, for the background-color of a design component, the value `red` can be entered directly in the entry helper.
- Dynamic, or XPath mode, where an XPath expression is entered. The expression is evaluated at runtime, and the result is entered as the value of the property. For example, for the background color of a design component, the following XPath expression can be entered: `/root/colors/color1`. At runtime, the content of the node `/root/colors/color1` will be retrieved and entered as the value of the background-color property.

Switching between static and dynamic (XPath) modes
For each property for which XPath expressions are enabled, static mode is selected by default. To switch a property to dynamic (XPath) mode, select that property and click the XPath icon in the toolbar of the entry helper (*screenshot below*).



If a static value was present for that property, it is now cleared and the mode is switched to dynamic. The Edit XPath Expression dialog appears. It is in this dialog that you enter the XPath expression for the property. Click **OK** when finished.

After you enter an XPath expression for the property, an Edit XPath expression button appears in the Value column for that property (*screenshot above*). Click this button to subsequently edit the XPath expression. If you wish to switch back to static mode, click the XPath icon in the toolbar. This will clear the XPath expression and switch the property to static mode.

Note: There are two important points to note. First, only one mode (static or dynamic), and the value/expression for that mode, is active at any time. Any value/expression that previously existed for the other mode is cleared; so switching to the other mode will present that mode with an empty entry field. (In order to go back to a previous value/expression, use the Undo command.) Second, if you reselect a property after further editing the SPS, then that property will be opened in the mode it was in previously.

Creating and editing the XPath definition

The XPath definition is created and edited in the Edit XPath Expression dialog. This dialog is accessed in two ways:

- Each time you switch to the dynamic mode of a property from static mode (by clicking the XPath icon in the toolbar of the entry helper), the Edit XPath Expression dialog appears. You can now create the XPath expression. (Note that clicking the toolbar icon when already in dynamic mode switches the mode to static mode; it does not pop up the Edit XPath Expression dialog.)
- The Edit XPath Expression dialog also pops up when you click the Edit XPath Expression button in the Value field of a property that already has an XPath expression defined for it. The dialog will contain the already defined XPath expression for that property, which you can now edit.

After you enter or edit the XPath expression in the entry field, click **OK** to finish.

Values returned by XPath expressions

The most important benefits of using XPath expressions to set a property value are that: (i) the property value can be taken from an XML file (instead of being directly entered); and/or (ii) an XPath expression can test some condition relating to the content or structure of the XML document being processed, and accordingly select a value. XPath expressions return values in the following two categories:

- *XML node content*
  The XPath expression can address nodes in: (i) the XML document being processed by the SPS, or (ii) any accessible XML document. For example the expression `Format/@color` would access the `color` attribute of the `Format` child of the context node. The value of the `color` attribute will be set as the value of the property for which the XPath expression was defined. A node in some other XML document can be accessed using the `doc()` function of XPath 2.0. For example, the expression `doc('Styles.xml')//colors/color-3` would retrieve the value of the element `color-3` in the XML document `Styles.xml` and set this value as the value of the property for which the XPath expression was defined.
- *XPath expression*
  The value of the property can come from the XPath expression itself, not from the XML document. For example, the background color of an element that is being output as a row can be made to alternate depending on whether the position of the row is odd-numbered or even-numbered. This could be achieved using the XPath 2.0 expression: `if (position() mod 2 = 0) then 'red' else 'green'`. Note that the return value of this expression is either the string `red` or the string `green`, and it will be set as the value of the property for which the XPath expression was defined. In the example just cited, the property values were entered as string literals. Alternatively, they could come from an XML document, for example: `if (position() mod 2 = 0) then doc('Styles.xml')//colors/color-1 else doc('Styles.xml')//colors/color-2`. Conversely, the XPath expression could be a straightforward string, for example: `'green'`. However, this is the same as entering the static value `green` for the property.

Chapter 9

Additional Editing Procedures

9 Additional Editing Procedures

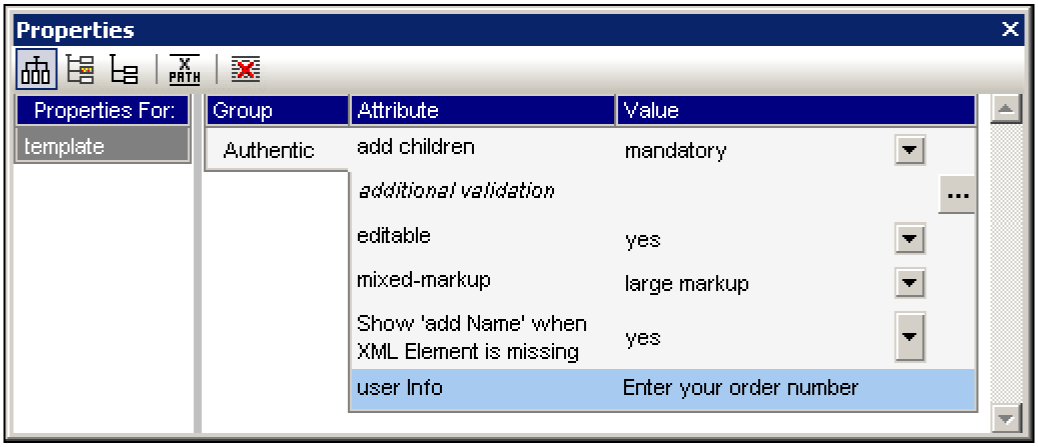
Additional to the content editing and presentation procedures, StyleVision provides a number of useful features that can be used in the SPS. These procedures are listed below and described in detail in the sub-sections of this section.

- Authentic Node Properties. Individual nodes in the XML document have Authentic View-specific properties. Nodes can be defined to be non-editable, to be displayed with markup tags, to display user information on mouseover, etc.
- Additional Validation. A node can be tested using an XPath expression to return a Boolean value that determines whether user input for that node is valid. This test is in addition to document validation against a schema.
- Input Formatting (Formatting Numeric Datatypes). The formatting of numeric datatype content, including dates, can be customized using the tools provided by the Input Formatting feature.
- Working with Dates. In Authentic View, a graphical date-picker ensures that dates are entered in the correct XML Schema format. Furthermore, dates can be manipulated and formatted as required.
- Parameters. Parameters are declared at the global SPS level with a default value. These values can then be overridden at runtime by values passed to the stylesheet from the command line.,
- Unparsed Entity URIs. URIs can be stored in unparsed entities in the DTD on which an XML document is based. The Unparsed Entity URI feature enables images and hyperlinks to use these URIs as target URIs.
- Using Scripts. StyleVision contains a JavaScript Editor in which JavaScript functions can be defined. These functions are then available for use as event handlers anywhere within the SPS, and will take effect in the output HTML document.

9.1 Authentic Node Properties

Authentic node properties are properties you set for the display of a node **in Authentic View**. For example, a node can be displayed with XML markup tags, be defined to be non-editable, and/or to have user information displayed when the cursor is placed over the node. Authentic node properties can be set on various SPS components. What properties are available for a component depends upon the type of component it is.

To assign Authentic node properties, select the required component in the design. Then, in the *Authentic* group of properties in the Properties entry helper (*screenshot below*), specify the required Authentic node settings.



Properties

Properties For: template

Group	Attribute	Value
Authentic	add children	mandatory
	additional validation	
	editable	yes
	mixed-markup	large markup
	Show 'add Name' when XML Element is missing	yes
	user Info	Enter your order number

Defining Authentic Properties

The following node settings can be made to control the behavior of individual nodes in the Authentic View display.

Add children

This setting is available when the selected node is an element. It allows you to define what child elements of the selected element are inserted when the selected element is added. The options are: all child elements, mandatory child elements, and no child element.

Content is editable

Defines whether the node is editable or not. By default the node is editable. This setting is available when the selected node is an element, attribute, or contents. Auto-Calculation results cannot be edited because the value is computed with the XPath expression you enter for the Auto-Calculation; this option is therefore not available for Auto-Calculations.

Mixed markup

This setting is available when the selected node is an element or attribute, and enables you to specify how individual nodes will be marked up in the mixed markup mode of Authentic View. The following options exist: large markup (tags with node names); small markup (tags without node names); and no markup.

Show "add Name" when XML Element is missing

Determines whether a prompt ("Add [element/attribute name]") will appear in Authentic View when the selected element or attribute is missing. By default, the prompt will be displayed. This setting is available when the selected node is an element or an attribute.

User info

Text entered in this text box appears as a tooltip when the mouse pointer is placed over the node. It is available when the selected node is an element, attribute, contents, or an Auto-Calculation. If both the element/attribute node as well as the contents node has User Info, then the User Info for the contents node is displayed as the tooltip when the mouse is placed over the node.

Maximum number of database records to be displayed

This node setting applies to nodes that represent a DB table, i.e. `Row` elements that are children of a top-level table element. The value entered here specifies the number of records in the table that will be displayed.

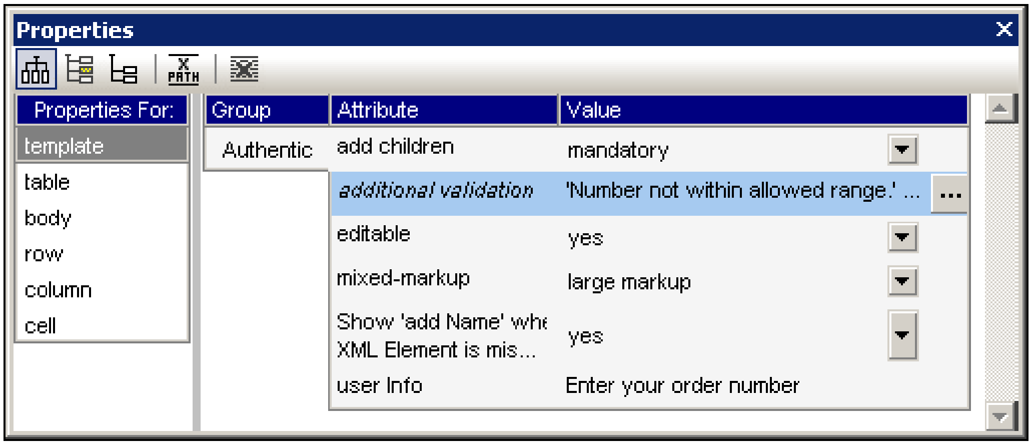
9.2 Additional Validation

The Additional Validation setting is available when the selected component is an element or attribute node, contents (`contents` placeholder), a data-entry device, or an Auto-Calculation. You can set an XPath expression to define the validity of the XML value of the node or Auto-Calculation. An XML value that falls outside this defined range is invalid. If the XML value of the node is invalid, this is made known to the Authentic View user by means of an error message when the XML document is validated (**F8**). The error message that is displayed is the text you enter into the Error message field of the Additional Validation setting.

Setting Additional Validation

To set additional validation, do the following:

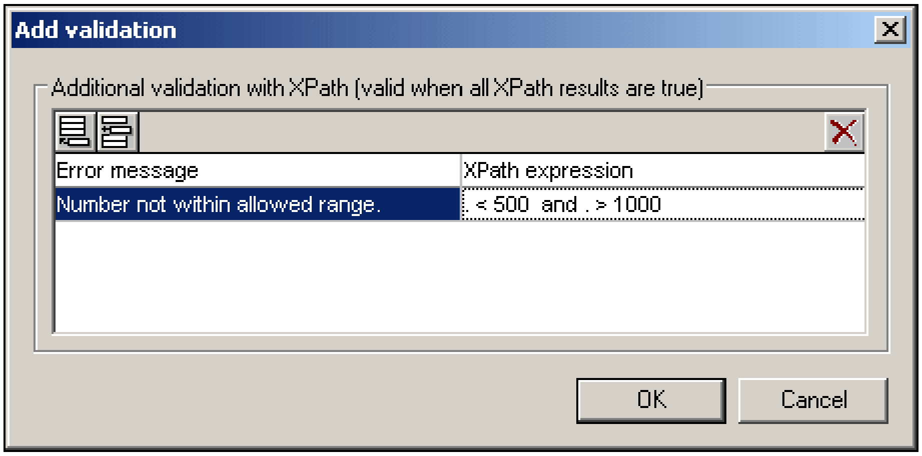
1. Select the component for which additional validation is required.
2. In the Properties entry helper, select the *Authentic* group of properties, and click the Edit button ... of the Additional Validation property (screenshot below). This pops up the Additional Validation dialog.



Properties For:
template
table
body
row
column
cell

Group	Attribute	Value
Authentic	add children	mandatory
	additional validation	'Number not within allowed range.' ...
	editable	yes
	mixed-markup	large markup
	Show 'add Name' whe XML Element is mis...	yes
	user Info	Enter your order number

3. In the Additional Validation dialog (*screenshot below*), add a row for an Additional Validation entry by clicking the **Add** button near the top left of the pane.



Additional validation with XPath (valid when all XPath results are true)

Error message	XPath expression
Number not within allowed range.	. < 500 and . > 1000

4. In the XPath expression column, enter an XPath expression to define the validity range of the XML data in that component.

5. Enter an error message to display when the data is invalid.
6. Click **OK** to finish.

9.3 Input Formatting (Formatting Numeric Datatypes)

Input Formatting enables the contents of numeric XML Schema datatype nodes (*see list below*) to be displayed in a format other than the lexical representation of that datatype. (For example, the lexical representation of an `xs:date` datatype node is `YYYY-MM-DD`., with an optional timezone component, such as `+02:00`.) The Input Formatting is displayed in Authentic View and, depending on the formatting definition, may also be available for display in the HTML output. Input Formatting can also be used to format the result of an Auto-Calculation if the result of the Auto-Calculation is in the lexical format of one of the numeric datatypes (*see list below*) for which Input Formatting is available.

In the sub-sections of this section, we describe:

- how the Input Formatting mechanism works, and
- the syntax for defining the input formatting.

Note: Input Formatting does not change the format in which the data is stored in the XML document. In the valid XML document, the data is always stored in the lexical format appropriate to the datatype of the node. Input Formatting is applied to the display in Authentic View and, optionally (if available), to the display in the output.

Numeric datatypes for which Input Formatting is available
Input Formatting is available for the following datatypes:

- `xs:decimal`; `xs:integer`; the 12 built-in types derived from `xs:integer`
- `xs:double` and `xs:float` when values are between and including 0.000001 and 1,000,000. Values outside this range are displayed in scientific notation (for example: `1.0E7`), and cannot have Input Formatting applied to them.
- `xs:date`; `xs:dateTime`: `xs:duration`
- `xs:gYear`; `xs:gYearMonth`; `xs:gMonth`; `xs:gMonthDay`; `xs:gDay`

The Input Formatting Mechanism

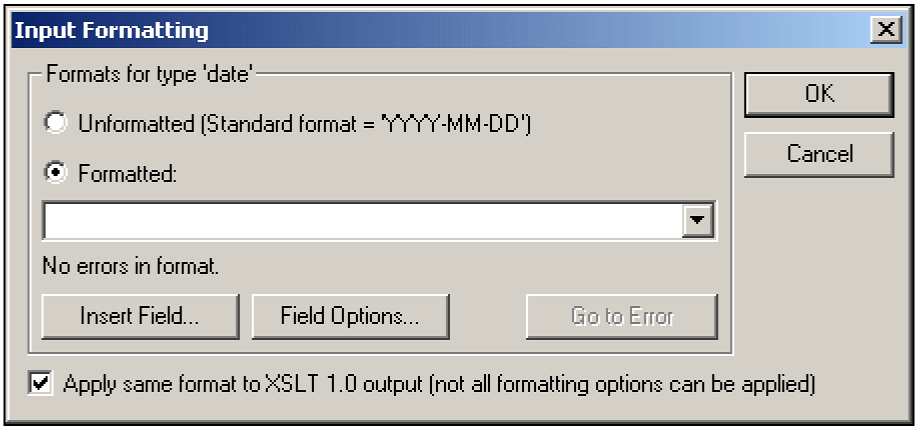
Input formatting can be applied to:

- A numeric datatype node, such as `xs:decimal` or `xs:date` that is present in the SPS **as contents or an input field**.
- An Auto-Calculation that evaluates to a value which has the lexical format of a numeric datatype.

Defining Input Formatting

To define Input Formatting for a node or Auto-Calculation in the SPS, do the following:

1. Select the `contents` placeholder or input field of the node, or the Auto-Calculation.
2. In the Properties entry helper, select the `content` item, and then the *Content* group (or *AutoCalc* group) of properties. Now click the Edit button ... of the `Input Formatting` property. The Input Formatting dialog appears (*screenshot below*). It is different according to whether the selected component was a node or an Auto-Calculation. If the selected component was a node, then a dialog like the one below appears.



Formats for type 'date'

○ Unformatted (Standard format = 'YYYY-MM-DD')

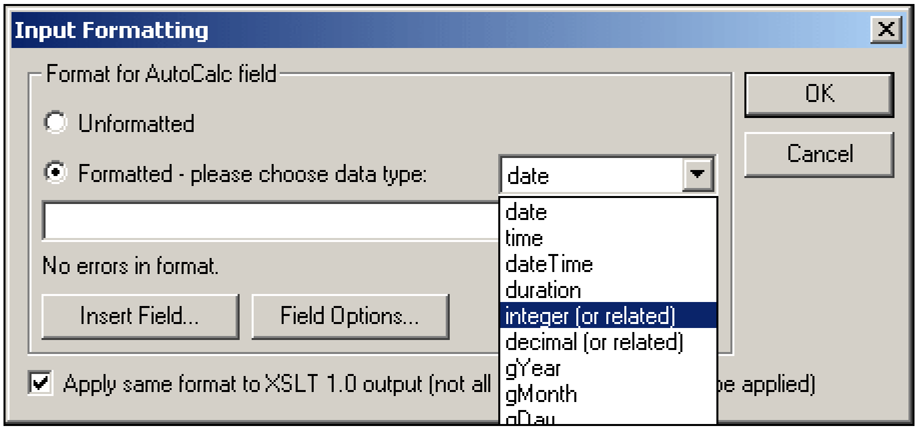
● Formatted:

No errors in format.

☑ Apply same format to XSLT 1.0 output (not all formatting options can be applied)

Note that the dialog contains the line: `Formats for type 'date'` and that the standard format for the `xs:date` datatype is given. For a node of some other datatype, this information would be correspondingly different.

If the selected component was an Auto-Calculation, the following dialog appears.



Format for AutoCalc field

○ Unformatted

● Formatted - please choose data type: date

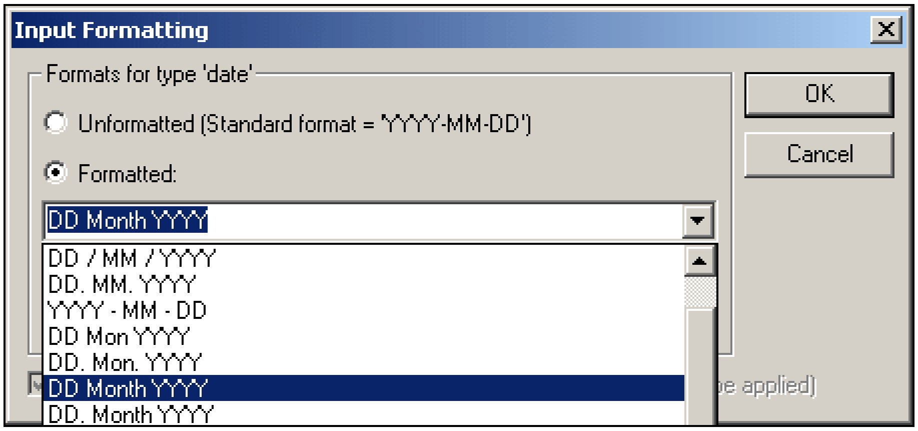
No errors in format.

☑ Apply same format to XSLT 1.0 output (not all ... be applied)

3. You now specify whether the display of the component's value is to be unformatted or formatted. Do this by selecting the appropriate radio button. (If the value is unformatted, it has the standard formatting for the datatype of the selected node or the datatype of

the Auto-Calculation result. If you specify Input Formatting for an Auto-Calculation, you have to additionally select (from a dropdown list) the datatype of the expected result.

4. Enter the Input Formatting definition. This definition can be entered in three ways: (i) by selecting from a dropdown list of available options for that datatype (*see screenshot below*); (ii) by entering the definition directly in the input field; and (iii) by using the **Insert Field** and **Field Options** entry helpers to build the definition correctly. See Input Formatting Syntax for a full description of the various formatting options.



Input Formatting

Formats for type 'date'

○ Unformatted (Standard format = 'YYYY-MM-DD')

● Formatted:

DD Month YYYY

DD / MM / YYYY
DD. MM. YYYY
YYYY - MM - DD
DD Mon YYYY
DD. Mon. YYYY
DD Month YYYY
DD. Month YYYY

Errors in syntax

If there is an error in syntax, the following happens:

- The definition is displayed in red.
- An error message, also in red, is displayed below the input field.
- The **OK** button in the Input Formatting dialog is disabled.
- The **Go to Error** button in the Input Formatting dialog is enabled. Clicking it causes the cursor to be placed at the point in the format definition where the syntax error is.

Mismatch of data and datatype formats

If the data entered in an XML node does not match the lexical format of that node's datatype, or if the result of an Auto-Calculation does not match the lexical format of the expected datatype, then the formatting will be undefined and will not be displayed correctly in the output.

Applying Input Formatting to the output

The Input Formatting that you define applies to Authentic View, which is supported in the Enterprise and Professional editions.

Some Input Formatting definitions—not all—can also, additionally, be applied to HTML output. To do this, check the Apply Same Format to XSLT Output check box. If this option is not checked, or if it is not available, then only Authentic View will display the Input Formatting, while the output will display the value in the standard format for the datatype of the component (the lexical format).

Input Formatting Syntax

The syntax for Input Formatting is:

```
([prefix character/s]field[suffix
   character/s][{field-option1,field-option2,...}])+
```

where **prefix character/s** and **suffix character/s** are optional specifiers used to control alignment and the display of positive/negative symbols;
field can be any datatype-specific formatting or text; and
{field-option(s)} is an optional qualifier, that enables additional formatting options.

Explanation of definition syntax
The Input Formatting definition is constructed as follows:

- The definition is composed of one or more fields. For example, the definition `DD Month YYYY` has three fields.
- Fields can be run together, or they can be separated by the following characters: space, hyphen, comma, colon, period, or by a text string in single or double quotes. For e xample, in the definition: `DD-Month' in the year 'YYYY`, the fields `DD` and `Month` are separated by a hyphen, and the fields `Month` and `YYYY` are separated by a text string enclosed in single quotes.
- A field can have optional prefix and/or suffix character/s. For example: `<+###,##0.00`.
- A field can have one or more optional field-options. The field-option/s for each field must be contained in a single set of curly braces, and must follow the field without any intervening space. Multiple field-options for a single field are separated by ", " (comma). For example, in the definition: `DD Month{uc,ro} YYYY`, the curly-brace-enclosed `uc` and `ro` are field-options for the field `Month`.

Examples
Example of Input Formatting for an `xs:decimal` datatype:

```
"$"(##0.00)
```

Examples of the output would be:

```
$ 25.00
$ 25.42
$267.56
```

Example of Input Formatting for an `xs:date` datatype:

```
DD Month{uc,ro} YYYY
```

where `uc` and `ro` are field-options for making the `Month` field uppercase and read-only, respectively

An example of the output would be:

```
24 SEPTEMBER 2003
```

Field types
A field type represents a component of the data and the way that component is to be formatted. The formatting inherent in the field type can be modified further by prefix and suffix modifiers as well as by field options. The following tables list the available field types. Note that, in the drop-down menu of the Input Formatting dialog, there are type-specific and field-only Input Formatting definitions. You can select one of these and modify them as required by adding prefix modifiers, suffix modifiers, and/or field options.

Field Type	Explanation
#	Space if no digit at position
0	Zero if no digit at position
,	Digit separator
Y	Year
y	year (base = 1930); see Note below
MM	Month, must have length of 2
DD	Day, must have length of 2
W	Week number
d	Weekday number (1 to 7)
i	Day in the year (1 to 366)
hh	Hour (0 to 23), must have length of 2
HH	Hour (1 to 12), must have length of 2
mm	Minute, must have length of 2
ss	Second, must have length of 2
AM	AM or PM
am	am or pm
AD	AD or BC
ad	ad or bc
CE	CE or BCE
ce	ce or bce

Field Type	Explanation
Weekday	Weekday (Sunday, Monday...)
WEEKDAY	Weekday (SUNDAY, MONDAY...)
weekday	Weekday (sunday, monday...)
Wkd	Weekday (Sun, Mon...)
WKD	Weekday (SUN, MON...)
wkd	Weekday (sun, mon...)
Month	Month (January, February...)
MONTH	Month (JANUARY, FEBRUARY...)
month	Month (january, february...)
Mon	Month (Jan, Feb...)
MON	Month (JAN, FEB...)
mon	Month (jan, feb...)

Notes on field length and entry length
The following points relating to the length of data components should be noted:

Length of date fields: When fields such as `MM`, `DD`, `HH`, `hh`, `mm`, and `ss` are used, they must have a length of 2 in the definition. When the `y` or `Y` fields are used, the number of `y` or `Y` characters in the definition determines the length of the output. For example, if you specify `YYY`, then the output for a value of `2006` would be `006`; for a definition of `YYYYYY`, it would be `002006`. See also Base Year below.

Extending field length: The `*` (asterisk) symbol is used to extend the length of a non-semantic numeric field (integers, decimals, etc). In the case of decimals, it can be used on either or both sides of the decimal point. For example, the input formatting `*0.00*` ensures that a number will have zeroes as specified in the formatting if these digit locations are empty, as well as any number of digits on both sides of the decimal point.

Entry lengths in Authentic View: The display in Authentic View of the contents of a node is based on the Input Formatting definition for that node. Therefore, the Authentic View user will not be able to insert more characters than are allowed by the Input Formatting definition. This is a useful way to restrict input in Authentic View. Note, however, that if the length of a **pre-existing** value in the XML document exceeds the length specified in the formatting definition, then the entire value is displayed.

Prefix and suffix modifiers

Prefix and suffix modifiers are used to modify the textual alignment and positive/negative representations of fields. The following table lists the available prefix and suffix modifiers.

Prefix	Suffix	Explanation
<		Left aligned; default for text. For numbers, which are aligned right by default, this is significant if there are a fixed number of leading spaces.
>		Right aligned; default for numbers.
?		Minus symbol adjacent to number if negative; nothing otherwise. This is the default for numbers.
<?		Minus symbol left-aligned if negative; nothing otherwise. Number left-aligned, follows minus sign.
<?>		Minus symbol left-aligned if negative; nothing otherwise. Number right-aligned.
-	-	Minus symbol adjacent to number if negative; space otherwise. Located before number (prefix), after number (suffix).
<-	>-	Minus symbol if negative; space otherwise. Number and sign adjacent. Left-aligned (prefix); right-aligned (suffix).
<->		Minus symbol left-aligned if negative; space otherwise. Number right-aligned.
+	+	Plus or minus sign always, located adjacent to number; before number (prefix), after number (suffix).
<+	>+	Plus or minus sign always, located adjacent to number; left-aligned (prefix), right-aligned (suffix).
<+>		Plus or minus sign always, left-aligned; number right-aligned.
(	)	Parentheses if negative; space otherwise. Adjacent to number.
<(		Parentheses if negative; space otherwise. Adjacent to number. Left-aligned.
<(>		Parentheses if negative; space otherwise. Left parentheses left-aligned; number and right parentheses adjacent and right-aligned.
[	]	Parentheses if negative; nothing otherwise. Adjacent to number.
*	*	Extendable number of digits to left (prefix) or to right (suffix)
_	_	Space
^	^	Fill character (defined in options)
	th	Ordinality of number in EN (st, nd, rd, or th)
	TH	Ordinality of number in EN (ST, ND, RD, or TH)

Field options

Field options enable advanced modifications to be made to fields. The following options are available:

Option	Explanation
uc	Make uppercase
lc	Make lowercase
left	Left aligned
right	Right aligned
ro	Read (XML) only; no editing allowed
edit	The field is editable (active by default)
dec=<char>	Specify a character for the decimal point (default is point)
fill=<char>	Specify fill character
base=<year>	Base year for year fields (*see note below*)
pos	Show only positive numbers; input of negative numbers allowed

Note on Base Year

When using **short year formats** (such as `yy` and `YY`), the base year specifies a cut-off for a century. For example, the base year field option could be used in the definition `DD-MM-YY{base=1940}`. If the user enters a value that is equal to or greater than the last two digits of the base year, which are considered together as a two-digit positive integer, then the century is the same as that of the base year. If the value entered by the user is less than the integer value of the last two digits, then the century is the century of the base year plus one. For example if you set `base=1940`, then if the Authentic View user enters `50`, the value entered in the XML document will be `1950`; if the user enters `23`, the value entered in the XML document will be `2023`.

Note the following points:

- Although two digits are commonly used as the short year format, one-digit and three-digit short year formats can also be used with a base year.
- Datatypes for which short year formats can be used are: `xs:date`, `xs:dateTime`, `xs:gYear`, and `xs:gYearMonth`.
- If the Input Formatting is being set for an Auto-Calculation component, make sure that the correct datatype is selected in the Input Formatting dialog. (The selected date datatype should be that of the result to which the Auto-Calculation evaluates.)
- If the `yy` field type is used, the default base year is 1930. Explicitly setting a base year overrides the default.
- If the `YY` field type is used without any base year being set, then the Authentic View user will be able to modify only the last two digits of the four-digit year value; the first two digits remain unchanged in the XML document.

9.4 Working with Dates

If the source document contains nodes that take date values, using the `xs:date` or `xs:dateTime` datatypes in the underlying XML Schema makes available the powerful date and time manipulation features of XPath 2.0 (*see examples below*). StyleVision supports the `xs:date` or `xs:dateTime` datatypes by providing:

1. A graphical date picker to help Authentic View users enter dates in the correct lexical format of the datatype for that node.
2. A wide range of date formatting possibilities via the Input Formatting feature.

These StyleVision features are described in the sub-sections of this section: Using the Date-Picker and Formatting Dates. In the rest of the introduction to this section, we show how XPath 2.0 can be used to make calculations that involve dates.

Note: Date and time data cannot be manipulated with XPath 1.0. However, with XPath 1.0 you can still use the Date Picker to maintain data integrity and use Input Formatting to provide date formatting.

Date calculations with XPath 2.0
Data involving dates can be manipulated with XPath 2.0 expressions in Auto-Calculations. Given below are a few examples of what can be achieved with XPath 2.0 expressions.

- The XPath 2.0 functions `current-date()` and `current-dateTime()` can be used to obtain the current date and date-time, respectively.
- Dates can be subtracted. For example: `current-date() - DueDate` would return an `xdt:dayTimeDuration` value; for example, something like `P24D`, which indicates a positive difference of 24 days.
- Time units can be extracted from durations using XPath 2.0 functions. For example: `days-from-duration(xdt:dayTimeDuration('P24D'))` would return the integer `24`.

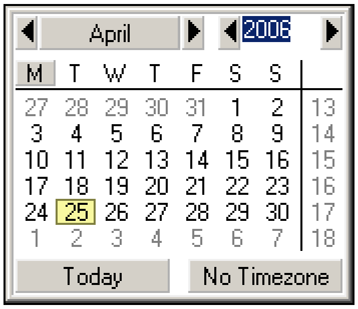
Here is an XPath 2.0 expression in an Auto-Calculation. It calculates a 4% annual interest on an overdue amount on a per-day basis and returns the sum of the principal amount and the accumulated interest:

```
if    (current-date() gt DueDate)
then  (round-half-to-even(InvoiceAmount +
         (InvoiceAmount*0.04 div 365 *
            days-from-duration((current-date() - DueDate))), 2))
else  InvoiceAmount
```

Such a calculation would be possible with XPath 2.0 only if the `DueDate` element were defined to be of a date type such as `xs:date` and the content of the element is entered in its lexically correct form, that is, `YYYY-MM-DD[±HH:MM]` , where the timezone component (prefixed by ±) is optional. Using the Date Picker ensures that the date is entered in the correct lexical form.

Using the Date-Picker

The Date Picker (*screenshot below*) is a graphical calendar in Authentic View for entering dates in the correct lexical format for nodes of `xs:date` and `xs:dateTime` datatype.



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

The lexical format is entered appropriately according to the datatype.

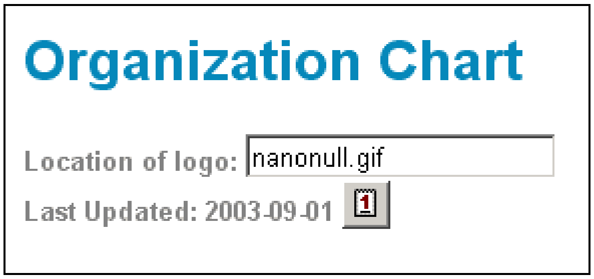
- For `xs:date`, the format of the entry is `YYYY-MM-DD[±HH:MM]`, where the timezone component (prefixed by ±) is optional according to the XML Schema specification. A value for the timezone component can be selected in the Date Picker.
- For `xs:dateTime`, the format of the entry is `YYYY-MM-DDTHH:MM:SS[±HH:MM]`. The timezone component (prefixed by ±) is optional according to the XML Schema specification. A value for the timezone component can be selected by the user.

Inserting and deleting a Date Picker in the design
A Date Picker can be inserted in the SPS design for any node that is of datatype `xs:date` or `xs:dateTime` and when that node is created either as contents or as an input field. A Date Picker can be inserted in one of two ways:

- By default when a node of datatype `xs:date` or `xs:dateTime` is created in the SPS. To set this default, toggle the Auto-Add Date Picker feature ON. Do this by selecting/de-selecting the command **Authentic | Auto-add Date Picker**. Alternatively, click the icon in the toolbar. When the Auto-Addition of the Date Picker is switched on, the Date Picker is inserted when any element of datatype `xs:date` or `xs:dateTime` is created as either contents or an input field, or changed to either of these two components.
- By clicking **Insert | Date Picker** when the cursor is at the desired location within the `xs:date` or `xs:dateTime` node in the SPS. This command can be accessed via the **Insert** menu, or via the context menu when the cursor is within the `xs:date` or `xs:dateTime` node.

When the Date Picker is inserted, the Date Picker icon appears at that location. To delete a Date Picker, use the **Delete** or **Backspace** buttons.

Using the Date Picker in Authentic View
In Authentic View, the Date Picker appears as an icon (*screenshot below*).

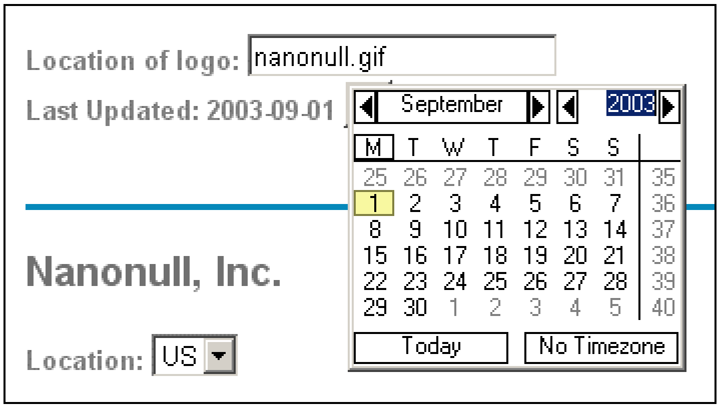


Organization Chart

Location of logo: nanonull.gif

Last Updated: 2003-09-01

To modify the date, click the icon. This pops up the Date Picker (*screenshot below*). To enter a new date, select the required date in the Date Picker. The date will be entered in the correct lexical format according to that node's datatype.



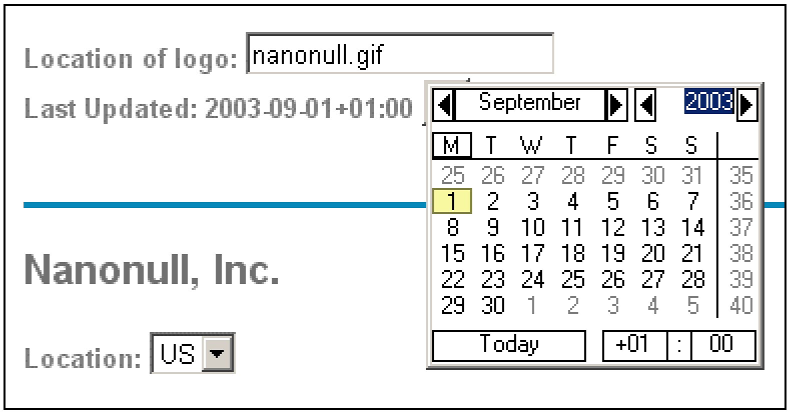
Location of logo: nanonull.gif

Last Updated: 2003-09-01

Nanonull, Inc.

Location: US

To enter a timezone, click the Timezone button, which is set to a default of No Timezone. The timezone will be entered in the lexical format appropriate to the node's datatype (*screenshot below*).



Location of logo: nanonull.gif

Last Updated: 2003-09-01+01:00

Nanonull, Inc.

Location: US

Formatting Dates

A date in an XML document is saved in the format specific to the datatype of its node. For example, the value of an `xs:date` node will have the format `YYYY-MM-DD[±HH:MM]`, while the value of an `xs:dateTime` node will have the format `YYYY-MM-DDTHH:MM:SS[±HH:MM]`. These formats are said to be the lexical representations of that data. By default, it is the lexical representation of the data that is displayed in Authentic View and the output. However, in the SPS, the Input Formatting feature can be used to display dates in alternative formats in Authentic View and, in some cases, optionally in the output.

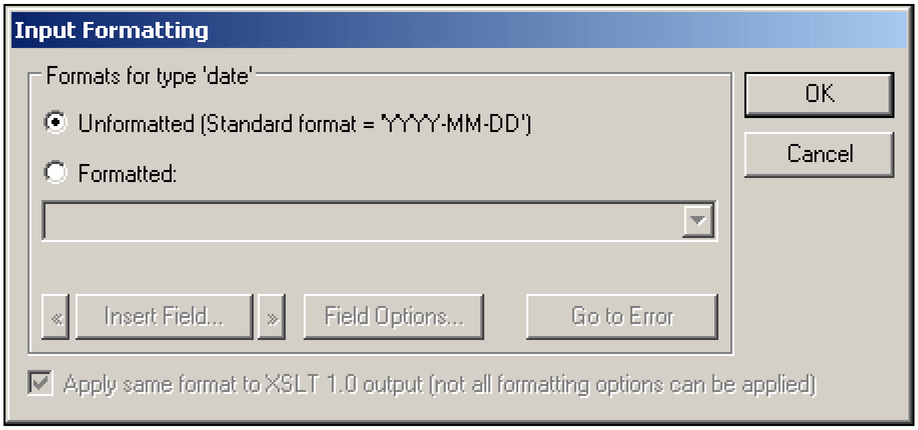
Input Formatting for dates can be used to define custom formats for nodes and Auto-Calculations of the following datatypes:

- `xs:date`
- `xs:dateTime`
- `xs:duration`
- `xs:gYear`
- `xs:gYearMonth`
- `xs:gMonth`
- `xs:gMonthDay`
- `xs:gDay`

Using Input Formatting to format date nodes

To format dates alternatively to the lexical format of the date node, do the following:

1. Select the `contents` placeholder or input field of the node. Note that input formatting can only be applied to nodes created **as contents or an input field**.
2. In the Properties entry helper, select the `content` item, and then the *Content* group of properties. Now click the Edit button ... of the `Input Formatting` property. This pops up the Input Formatting dialog (*screenshot below*).



Input Formatting

Formats for type 'date'

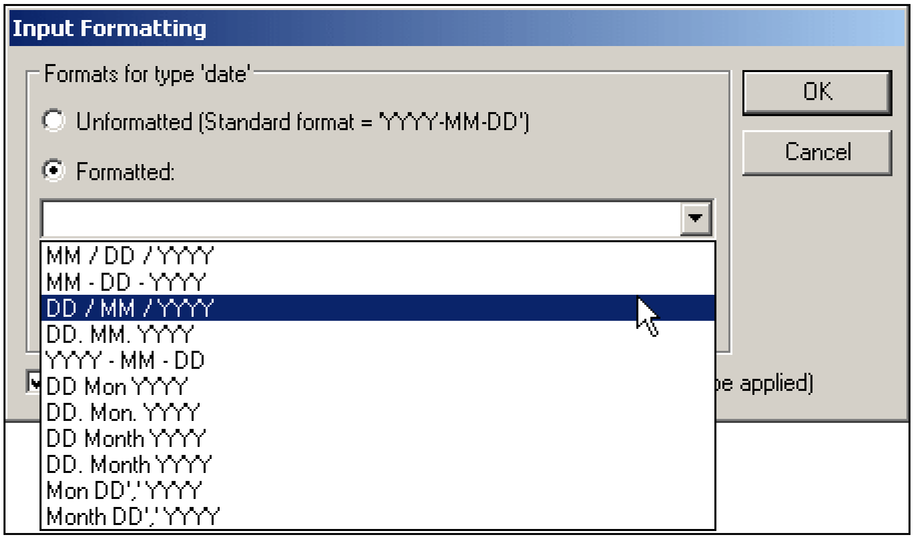
● Unformatted (Standard format = 'YYYY-MM-DD')

○ Formatted:

☑ Apply same format to XSLT 1.0 output (not all formatting options can be applied)

By default, the Unformatted radio button (the standard lexical format for the node's datatype) is selected.

3. To define an alternative format, select the Formatted radio button.
4. You can now select a predefined date format from the drop-down list of the combo box (*screenshot below*), or define your own format in the input field of the combo box. See [Input Formatting Syntax]() for details about the syntax to use when defining your own format.



Input Formatting

Formats for type 'date'

○ Unformatted (Standard format = 'YYYY-MM-DD')

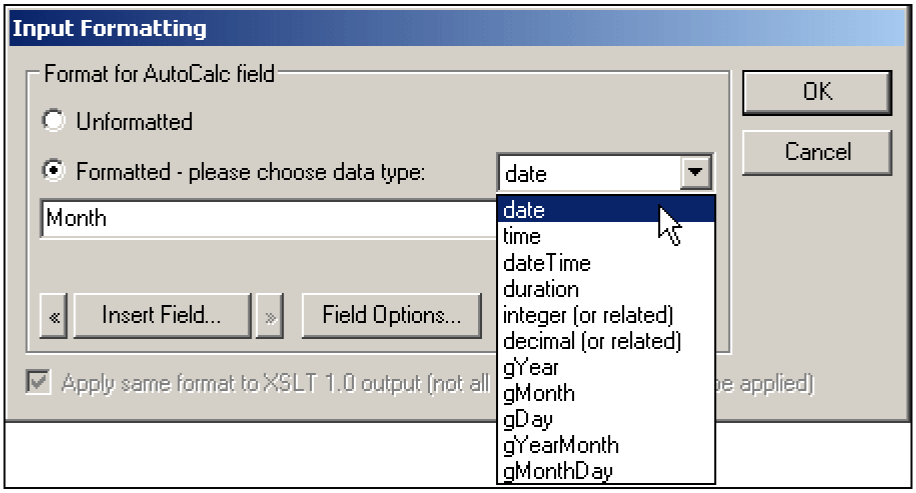
● Formatted:

MM / DD / YYYY
MM - DD - YYYY
DD / MM / YYYY
DD. MM. YYYY
YYYY - MM - DD
DD Mon YYYY
DD. Mon. YYYY
DD Month YYYY
DD. Month YYYY
Mon DD',' YYYY
Month DD',' YYYY

Using Input Formatting to format Auto-Calculations

When Auto-Calculations evaluate to a value that is a lexical date format, Input Formatting can be used to format the display of the result. Do this as follows:

1. Select the Auto-Calculation in the design.
2. In the Properties entry helper, select the `content` item, and then the *AutoCalc* group of properties. Now click the Edit button ... of the `Input Formatting` property. This pops up the Input Formatting dialog (*screenshot below*).



Input Formatting

Format for AutoCalc field

○ Unformatted

● Formatted - please choose data type: date

Month

date
time
dateTime
duration
integer (or related)
decimal (or related)
gYear
gMonth
gDay
gYearMonth
gMonthDay

☑ Apply same format to XSLT 1.0 output (not all e applied)

By default, the Unformatted radio button is selected.

3. To define an alternative format, select the Formatted radio button.
4. In the combo box to the right of the Formatted radio button, select the date datatype to which the Auto-Calculation will evaluate. You can now select a predefined date format from the drop-down list of the definition combo box (available options depend on the selected datatype), or define your own format in the input field of the combo box. See Input Formatting Syntax for details about the syntax to use when defining your own format.

Applying Input Formatting to the output

The Input Formatting that you define applies to Authentic View. Additionally, some Input Formatting definitions—not all—can also be applied to HTML output. To do this, check the Apply Same Format to XSLT Output check box. If this option is not checked or if it is not available, then only Authentic View will display the Input Formatting; the output will display the value in its lexical format (for nodes) or, in the case of Auto-Calculations, in the format to which the Auto-Calculation evaluates.

9.5 User-Declared Parameters

In an SPS, user-declared parameters are declared globally with a name and a default string value. Once declared, they can be used in XPath expressions anywhere in the SPS. The default value of the parameter can be overridden for individual XSLT transformations by passing the XSLT stylesheet a new global value via the command line.

Use of parameters

User-declared parameters are useful in the following situations:

- If you wish to use one value in multiple locations or as an input for several calculations. In this case, you can save the required value as a parameter value and use the parameter in the required locations and calculations.
- If you wish to pass a value to the stylesheet at processing time. In the SPS (and stylesheet), you use a parameter with a default value. At processing time, you pass the desired value to the parameter via the command line.

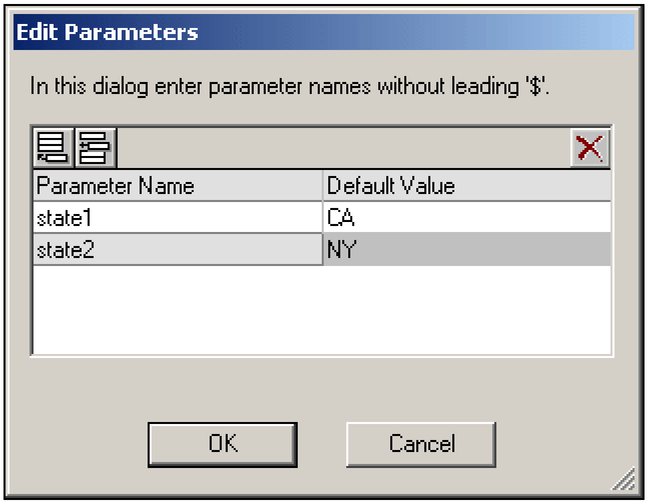
Usage mechanism

Working with user-declared parameters in the SPS consists of two steps:

1. Declaring the required parameters.
2. Referencing the declared parameters.

Declaring parameters

All user-defined parameters are declared and edited in the Edit Parameters dialog (*screenshot below*). The Edit Parameters dialog is accessed via: (i) the **Edit | Edit Stylesheet Parameters** command, (ii) the Edit button ... of the Parameters entry in the Design Tree entry helper, and (iii) the **Parameters** button in the Edit Database Filters dialog (**Edit | Edit DB Filter**).



In this dialog enter parameter names without leading '$'.

Parameter Name	Default Value
state1	CA
state2	NY

Declaring a parameter involves giving it a name and a string value—its default value. If no value is specified, the default value is an empty string. The default value will be used each time the parameter is referenced, and it is overridden only if a new value is passed for that parameter on the command line.

To declare a parameter, do the following:

1. In the Edit Parameters dialog, append or insert a new parameter by clicking the Append

or Insert buttons. A new line appears.

2. Enter the name of the parameter. Parameter names must begin with a letter, and can contain the characters `A` to `Z`, `a` to `z`, `0` to `9`, and the underscore.
3. Enter a default value for that parameter. The value you enter is accepted as a text string.

You can insert any number of parameters and modify existing parameters at any time while editing either the SPS or Authentic View.

Note: The Edit Parameters dialog contains all the user-defined parameters in an SPS.

Referencing declared parameters

Parameters can be referenced in XPath expressions by prefixing a `$` character before the parameter name. For example, you could reference a parameter in the XPath expression of an Auto-Calculation (e.g. `concat('www.', $company, '.com')`). If your SPS is DB-based, then you can also use parameters as the values of DB Filter criteria. The DB parameters, however, are declared and edited in the Edit Parameters dialog.

Note: While it is an error to reference an undeclared parameter, it is not an error to declare a parameter and not reference it.

9.6 Unparsed Entity URIs

If you are using a DTD and have declared an unparsed entity in it, you can use the URI associated with that entity for image and hyperlink targets in the SPS. This is useful if you wish to use the same URI multiple times in the SPS. This feature makes use of the XSLT function `unparsed-entity-uri` to pass the URI of the unparsed entity from the DTD to the output, and is therefore only available in the outputs (HTML); not in Authentic View.

Using this feature requires that the DTD, XML document, and SPS documents be appropriately edited, as follows:

1. In the DTD, (i) the unparsed entities must be declared, and (ii) the notations for the unparsed entity types must be declared.
2. In the XML document, the relevant nodes must be given values that are the names of the required unparsed entities.
3. In the SPS, when images and hyperlinks are inserted, they must correctly access the dynamic node values as unparsed entities.

Declaring an Unparsed Entity

Declaring an unparsed entity consists of two parts:

- Declaring a name, a URI, and a type for the unparsed entity in the DTD.
- Declaring a notation for that entity type in the DTD.

Both these steps require the editing of the DTD. You can enter the declarations either in the external DTD subset or the internal DTD subset.

Declaring the name, URI and type of the unparsed entity
In the example below, the unparsed entities are declared in the internal DTD subset, that is in the XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "UEURIDoc.dtd" [
<!ENTITY Picture SYSTEM "nanonull.gif" NDATA GIF>
<!ENTITY AltovaURI SYSTEM "http://www.altova.com" NDATA LNK>
]>
<document>
      ...
</document>
```

Two unparsed entities have been declared in the above fragment: `Picture` and `AltovaURI`. The URI for each is located between quotes, and the datatypes are respectively, `GIF` and `LNK`.

Declaring notations for the entity types
The XML document fragment listed above references a DTD called `UEURIDoc.dtd`, located in the same folder as the XML file. The notations for both datatypes (`GIF` and `LNK`) are declared in the DTD file, that is, in the external subset. The entire external DTD subset (in `UEURIDoc.dtd`) is given below, with the notations at the bottom.

```
<!-- Element declarations -->
<!ELEMENT document (header, para, img, link)>
<!ELEMENT header (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT img EMPTY>
 <!ATTLIST img
   src CDATA #REQUIRED>
<!ELEMENT link (#PCDATA)>
 <!ATTLIST link
   href CDATA #REQUIRED>

<!-- Notation Declarations -->
<!NOTATION GIF PUBLIC "urn:mime:img/gif">
<!NOTATION LNK PUBLIC "urn:mime:text/plain">
```

Now that the entities and the notations for them have been declared, the next step is to use the unparsed entities in the XML document and SPS.

Using an Unparsed Entity

After having declared an unparsed entity in the DTD (external or internal subset), the entity must be referenced in one or more nodes in the XML document before it can be used in the SPS. In this section, we describe how:

- An unparsed entity is referenced in a node in the XML document.
- Images and hyperlinks are inserted in the SPS so that XML node content is used as unparsed entities.

Referencing unparsed entities in the XML document content

Given below is a cut-down listing of the XML document `UEURIDoc.xml`, which shows how the unparsed entities declared in the previous section are used inside the `img` and `link` elements.

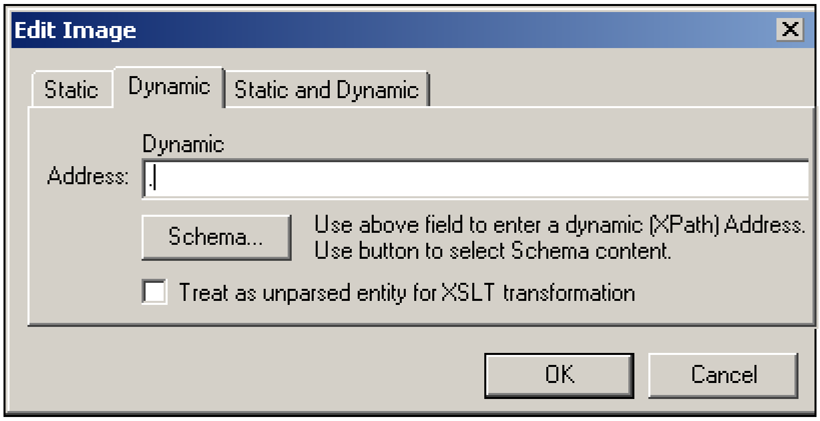
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "UEURIDoc.dtd" [
<!ENTITY Picture SYSTEM "nanonull.gif" NDATA GIF>
<!ENTITY AltovaURI SYSTEM "http://www.altova.com" NDATA LNK>
]>
<document>
   <header>Example of How to Use Unparsed Entity URIs</header>
   <para>...</para>
   <img src="Picture"/>
   <link href="AltovaURI">Link to the Altova Website.</link>
</document>
```

In the listing above, the `Picture` and `AltovaURI` entities are used as values of the `//img/@src` and `//link/@href` attributes. Note that they are used without the ampersand and semi-colon delimiters that are use for parsed general entities.

Inserting images and hyperlinks to use unparsed entities

Images and hyperlinks that reference unparsed entity URIs are used as follows:

1. Insert the image and hyperlink via the **Insert** menu.
2. In the Edit dialog for each, select the Dynamic tab properties (*screenshot below*), and enter an XPath expression that selects the node containing the name of the unparsed entities. In the XML document example given above, these nodes would be, respectively, the `//img/@src` and `//link/@href` nodes.



Dynamic

Address: .

Use above field to enter a dynamic (XPath) Address.
Use button to select Schema content.

☐ Treat as unparsed entity for XSLT transformation

3. Then check the Treat as Unparsed Entity check box at the bottom of the dialog. This causes the content of the selected node to be read as an unparsed entity; if an

unparsed entity of that name is declared, the URI associated with that unparsed entity is used to locate the resource (image or hyperlink).

When the stylesheet is processed, the URI associated with the entity name is substituted for the entity name. Note that if the URI is a relative URI, the XSLT processor expands it to an absolute URI applying the base URI of the DTD. So if the unparsed entity is associated with the relative URI "`nanonull.gif`", then this URI will be expanded to `file:///c:/someFolder/nanonull.gif`, where the DTD is in the folder `someFolder`.

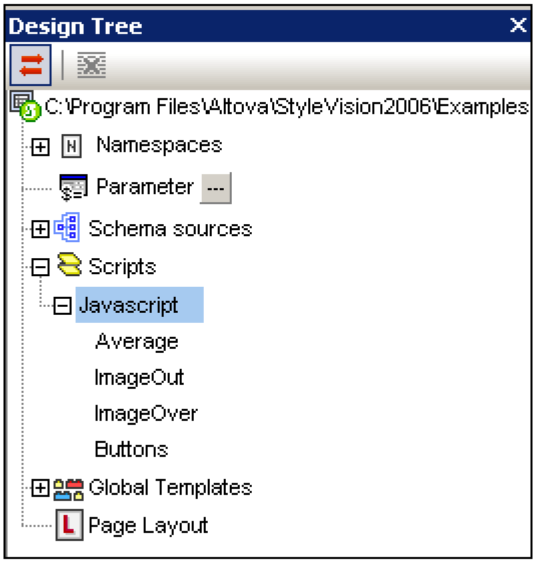
9.7 Using Scripts

In StyleVision, you can define JavaScript functions for each SPS in a JavaScript editor (available as a tab in the Design View). The function definitions created in this way are stored in the header of the HTML document and can be called from within the body of the HTML document. Such functions are useful when:

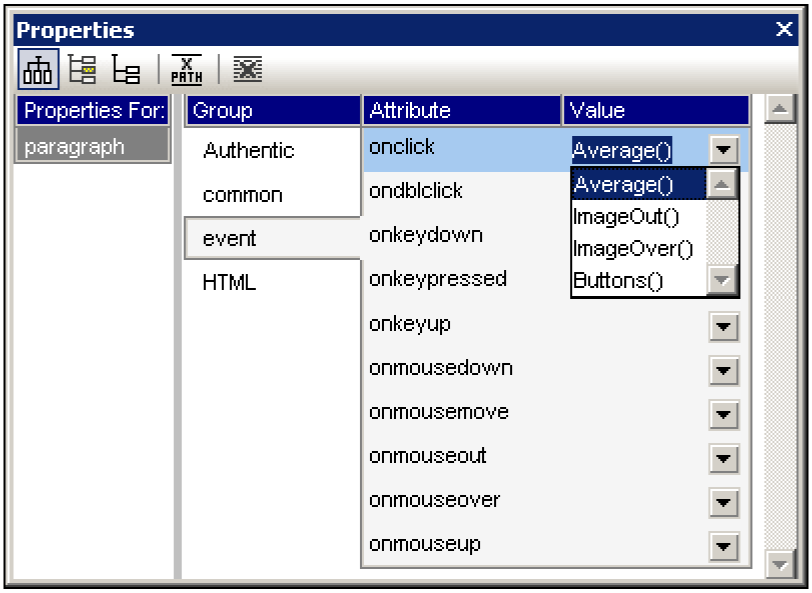
- You wish to achieve a complex result using multiple script statements. In this case it is convenient to write all the required scripts, as separate functions, in one location (the header) and refer to the functions subsequently in the design document.
- You wish to use a particular script at multiple locations in the design document.

How to define functions in the JavaScript Editor is described in the sub-section Defining JavaScript Functions.

In the GUI, all JavaScript functions which are defined for a given SPS in the JavaScript Editor are listed in the Design Tree window under the Scripts entry (*screenshot below*). The screenshot below indicates that four JavaScript functions, `Average`, `ImageOut`, `ImageOver`, and `Buttons`, are currently defined in the active SPS.



The functions defined in the JavaScript Editor are available as event handler calls within the GUI. When a component in the design document is selected, any of the defined functions can be assigned to an event handler property in the Event property group in the Properties Entry Helper (*screenshot below*). How to assign a JavaScript function to an event handler is described in the section Assigning Function to Event Handlers.

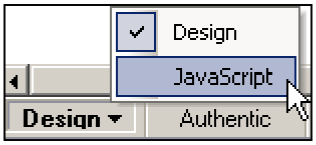


Note: Scripts are applicable in the HTML output only. They are not applicable in Authentic View.

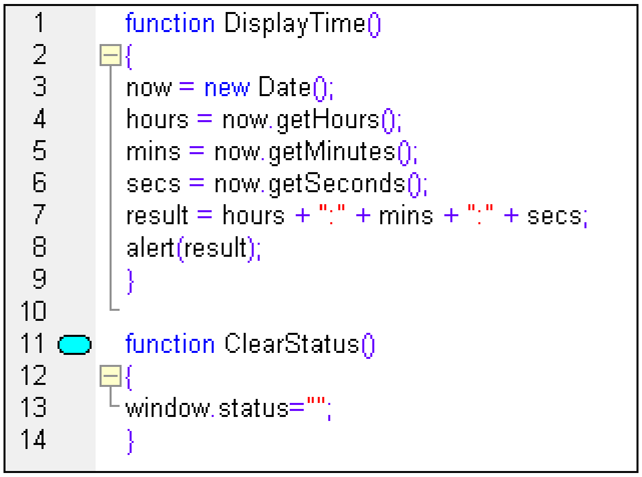
Defining JavaScript Functions

To define JavaScript functions, do the following:

1. In Design View, switch to the JavaScript Editor by clicking the Design View tab and selecting JavaScript (*screenshot below*).



2. In the JavaScript Editor, type in the function definitions (*see screenshot below*).



```
function DisplayTime()
{
now = new Date();
hours = now.getHours();
mins = now.getMinutes();
secs = now.getSeconds();
result = hours + ":" + mins + ":" + secs;
alert(result);
}

function ClearStatus()
{
window.status="";
}
```

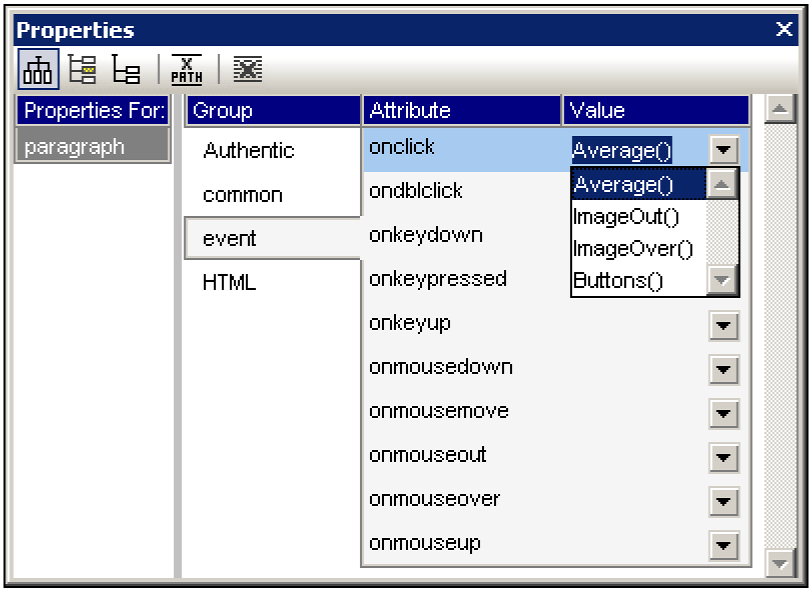
The screenshot above shows the definitions of two JavaScript functions: `DisplayTime` and `ClearStatus`. These have been described for the active SPS. They will be entered in the header of the HTML file as follows:

```
<script language="javascript">

<! -- function DisplayTime()
{
    now = new Date();
    hours = now.getHours();
    mins = now.getMinutes();
    secs = now.getSeconds();
    result = hours + "." + mins + "." + secs;
    alert(result)
}

function ClearStatus()
{
    window.status="";
}
-->

</script>
```

These functions can now be called from anywhere in the HTML document. In StyleVision, all the defined functions are available as options that can be assigned to an event handler property in the *Event* property group in the Properties Entry Helper. See Assigning Function to Event Handlers for details.

Assigning Functions as Event Handlers

In the StyleVision GUI, you can assign JavaScript functions as event handlers for events that occur on the HTML renditions of SPS components. These event handlers will be used in the HTML output. The event handler for an available event—such as `onclick`—is set by assigning a global function as the event handler. In the Properties Entry Helper, global functions defined in the JavaScript Editor are available as event handlers in the dropdown boxes of each event in the *Events* property group for the selected component (*screenshot below*).



To assign a function to an event handler, do the following:

1. Select the component in the SPS for which the event handler is to be defined. The component can be a node or content of any kind, dynamic or static.
2. In the Properties Entry Helper select the *Event* group. This results in the available events being displayed in the Attribute column (*screenshot above*).
3. In the Value column of the required event, click the down arrow of the combo box. This drops down a list of all the functions defined in the JavaScript Editor.
4. From the dropdown list, select the required function as the event handler for that event.

Chapter 10

Authentic View

10 Authentic View

In StyleVision, you can preview the Authentic View of an XML document that has been assigned as the Working XML File to the currently open StyleVision. This preview is available when you click the Authentic Preview tab in the Main Window of StyleVision. In order for the Authentic Preview to be enabled, you must assign the currently open StyleVision Power Stylesheet a Working XML File (**File | Assign Working XML File**).

This section provides:

- An overview of the interface
- A description of the toolbar icons specific to Authentic View
- A description of viewing modes available in the main Authentic Preview window
- A description of the context menus available at various points in the Authentic View of the XML document
- A detailed description of how to use various Authentic View features

Additional sources of Authentic View information are:

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

10.1 Interface

This section describes the Authentic View user interface. It contains the following sections:

- Overview of the GUI
- Authentic View toolbar icons
- Authentic View main window
- Authentic View entry helpers
- Authentic View context menus

Overview of the GUI

The Authentic Preview provides you with menu commands, toolbar icons, and context menus with which to edit the XML document that is displayed in the Main Window.

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

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

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

Status Bar
The Status Bar displays the XPath to the currently selected node. In the Authentic Preview of StyleVision, the XPath to the currently selected node is indicated in the Schema Tree, where the currently selected node is highlighted in gray. The XPath in Authentic Preview is not displayed in a status bar.

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

Authentic View Toolbar Icons

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

In the description below, related icons are grouped together.

Show/hide XML markup
Markup tags can be turned on or off in Authentic Preview.

Hide markup.

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

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

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

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

Append row to table

Insert row in table

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

Move current row up by one row

Move current row down by one row

Delete the current row

Please note: These commands apply only to **dynamic SPS tables**. They should not be used inside static SPS tables.

DB Row Navigation icons

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

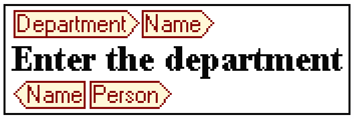
This icon opens the Edit Database Query dialog in which you can enter a query. Authentic View displays the queried record/s.

Authentic View Main Window

There are four viewing modes in Authentic View: Large Markup; Small Markup; Mixed Markup; and Hide All Markup. These modes enable you to view the document with varying levels of markup information. In Authentic Preview of StyleVision only two markup modes are available: Hide Markup and Show Large (Full) Markup.

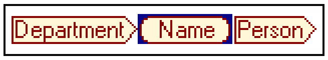
To switch between modes, use the commands in the **Authentic** menu or the icons in the toolbar (see the previous section, Authentic View toolbar icons).

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



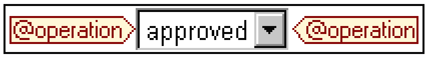
Enter the department

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



To expand the contracted element/attribute, double-click the contracted tag.

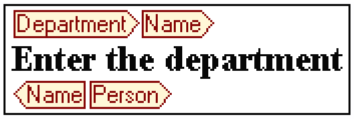
In large markup, attributes are recognized by the symbol `@` in the start and end tags of the attribute:



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

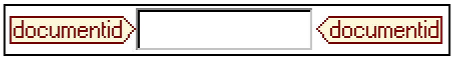
Content display
In Authentic View, content is displayed in two ways:

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

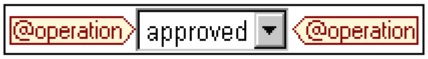


Enter the department

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

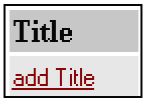


In the case of the other data-entry devices, your selection produces a corresponding XML value, which is specified in the StyleVision Power Stylesheet. Thus the selection "approved" in the display example below could map to an XML value of "1", or to "approved", or anything else; while "not approved" in the display could map to "0", or "not approved", or anything else.



Optional nodes

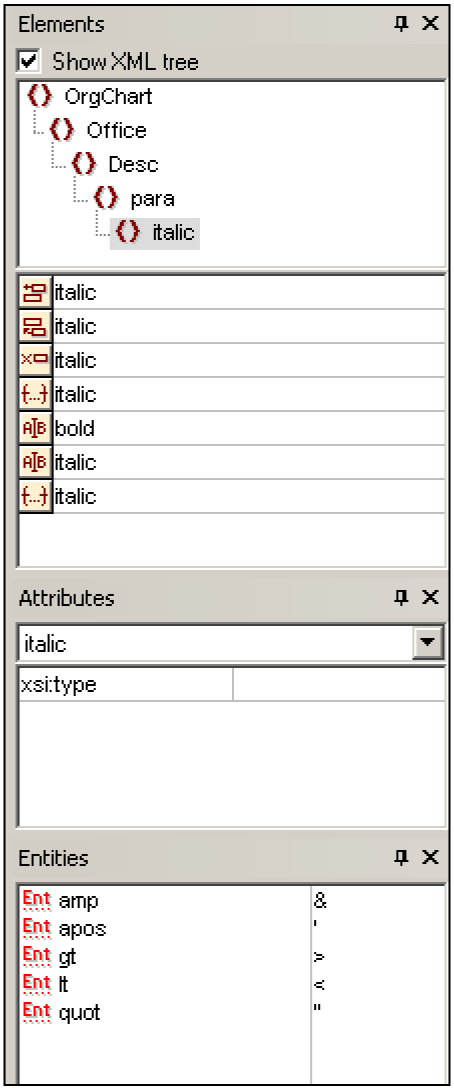
When an element or attribute is **optional** (according to the referenced schema), a prompt of type "add [*element/attribute*]" is displayed:



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

Authentic View Entry Helpers

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



Ent amp	&
Ent apos	'
Ent gt	>
Ent lt	<
Ent quot	"

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

Each of the entry helpers is described separately below.

Elements Entry Helper

The Elements Entry Helper consists of two parts:

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

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

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

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

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

Remove Element
Removes the element and its content.

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

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

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

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

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

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

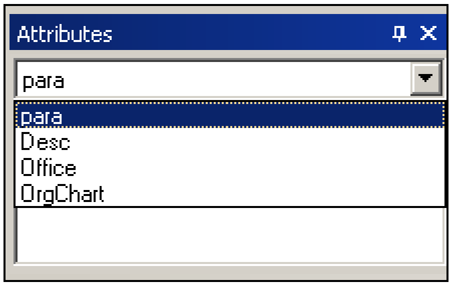
If a text range is selected rather than an element, applying an element to the selection will create the applied element at that location with the selected text range as its content. Applying an element when the cursor is an insertion point is not allowed.

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

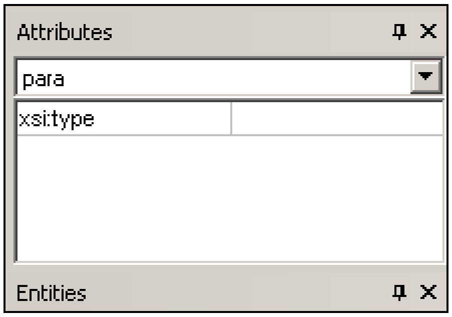
Clear Element (when insertion point selected)
This icon appears when the cursor is placed within an element that is a child of a mixed-content element. Clicking the icon clears the inline element.

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

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



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



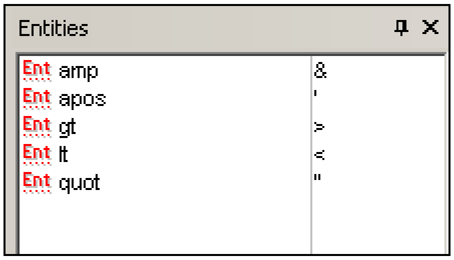
xsi:type	

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

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

Entities Entry Helper

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



Ent amp	&
Ent apos	'
Ent gt	>
Ent lt	<
Ent quot	"

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

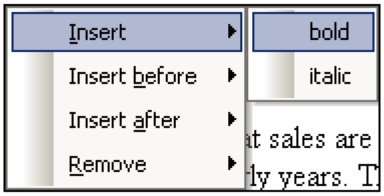
You can also **define your own entities** in Authentic View: see Define Entities in the Editing in Authentic View section.

Authentic View Context Menus

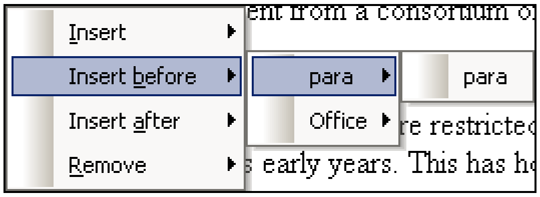
Right-clicking on some selected document content or node pops up a context menu with commands relevant to the selection or cursor location.

Inserting elements

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



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



Most of the commands available in the context menu are explained in Authentic View entry helpers.

Remove node

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

Insert entity

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

Insert CDATA Section

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

Remove

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

Clear

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

10.2 Editing in Authentic View

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

The section explains the following:

- The Date Picker is a graphical calendar that enters dates in the correct XML format when you click a date. See Using the Date Picker.
- An entity is shorthand for a special character or text string. You can define your own entities, which allows you to insert these special characters or text strings by inserting the corresponding entities. See Defining Entities for details.
- What image formats can be displayed in Authentic View.

To learn how to use all the features of Authentic View, please do the Authentic View Tutorial using either XMLSpy or Authentic Desktop. The Authentic View Tutorial is available with these products.

Tables in Authentic View

The three table types fall into two categories: SPS tables (static and dynamic) and XML tables.

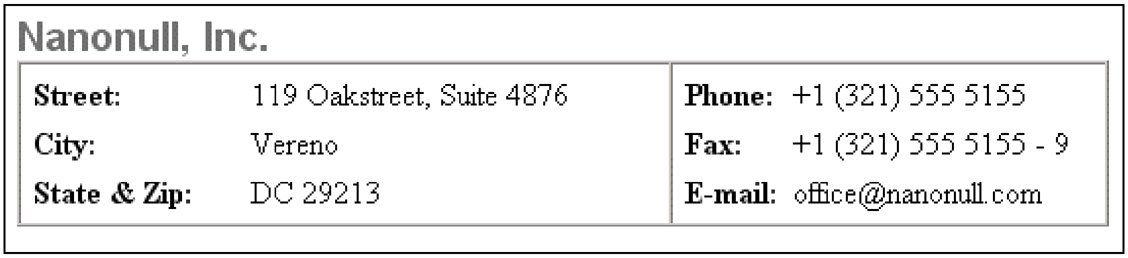
SPS tables are of two types: static and dynamic. SPS tables are designed by the designer of the StyleVision Power Stylesheet to which your XML document is linked. You yourself cannot insert an SPS table into the XML document, but you can enter data into SPS table fields and add and delete the rows of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

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

SPS Tables

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

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

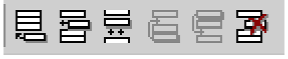


Nanonull, Inc.

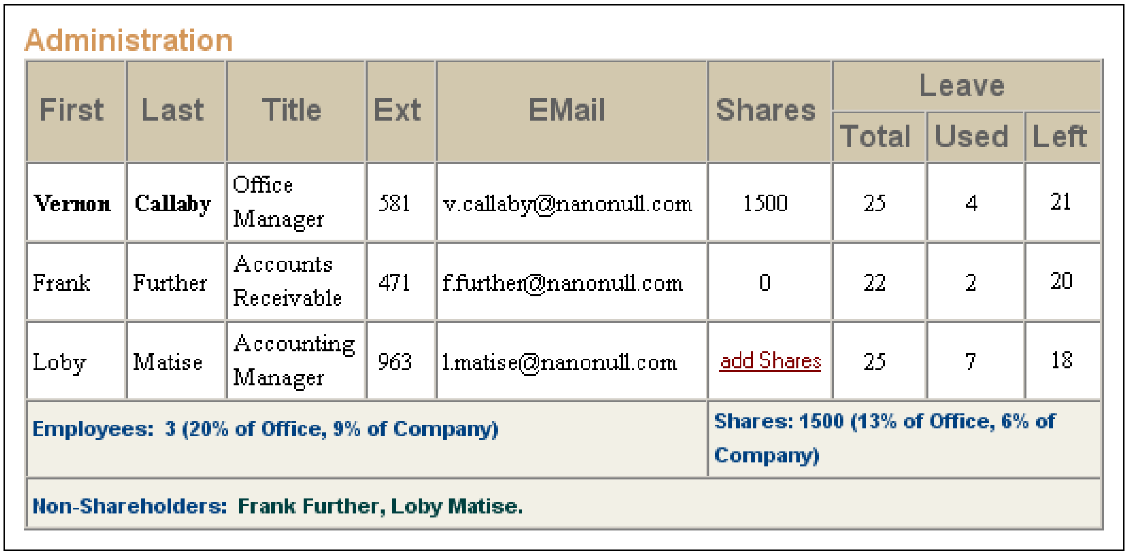
Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

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

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



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



Administration

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

To move among cells in the table, use the Up, Down, Left, and Right arrow keys. To move forward from one cell to the next, use the **Tab** key. Pressing the **Tab** key in the last cell of a row creates a new row.

XML Tables

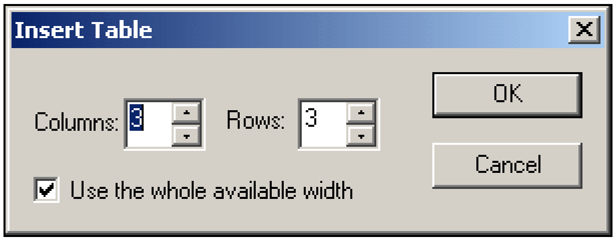
XML tables can be inserted by you, the user of Authentic View. They enable you to insert tables anywhere in the XML document where they are allowed, which is useful if you need to insert tabular information in your document. These tables will be printed out as tables when you print out directly from Authentic View. If you are also generating output with XSLT stylesheets, discuss the required output with the designer of the StyleVision Power Stylesheet.

Note that you can insert XML tables only at allowed locations. These locations are specified in the schema (DTD or XML Schema). If you wish to insert a table at additional locations, discuss this with the person designing the StyleVision Power Stylesheet.

Working with XML tables
There are three steps involved when working with XML tables: inserting the table; formatting it; and entering data. The commands for working with XML tables are available as icons in the toolbar (see XML table editing icons). Currently, XML tables cannot be inserted in the Authentic Preview of StyleVision.

Inserting tables
To insert an XML table:

1. Place your cursor where you wish to insert the table, and click the icon. (Note that where you can insert tables is determined by the schema.) This opens the Insert Table dialog (shown below).

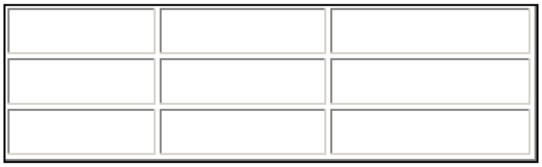


Insert Table

Columns: 3 Rows: 3

☑ Use the whole available width

2. Select the number of columns and rows, and specify whether you wish the table to extend the entire available width. For the specifications given in the dialog box shown above, the following table is created.

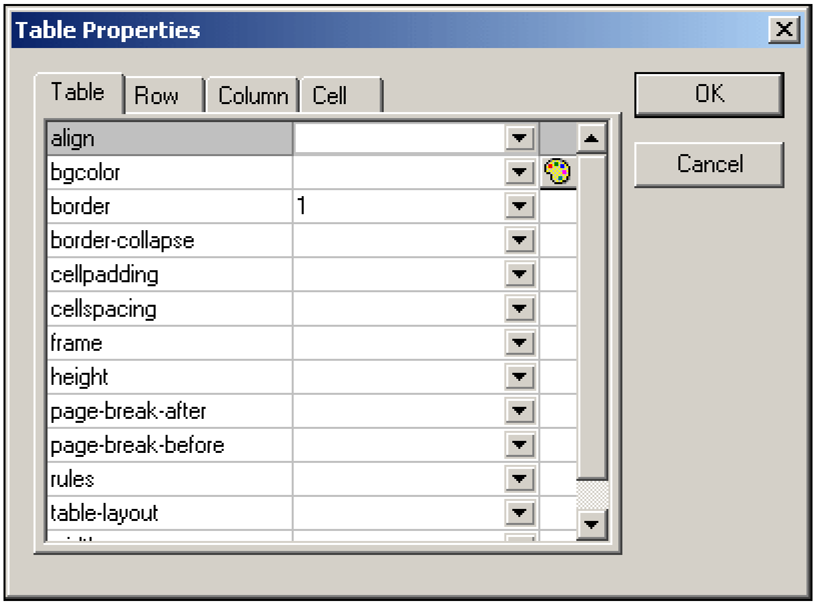


You can add and delete columns, create row and column joins later. Create the broad structure first.

Please note: All modifications to table structure must be made by using the **Table** menu commands. They cannot be made by changing attribute values in the Attribute Entry Helper.

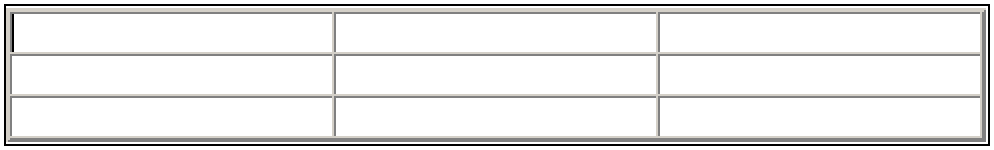
Formatting tables and entering data
To format your table:

1. Place the cursor anywhere in the table and click the (Table Properties) icon. This opens the Table Properties dialog (*see screenshot*), where you specify formatting for the table, or for a row, column, or cell.

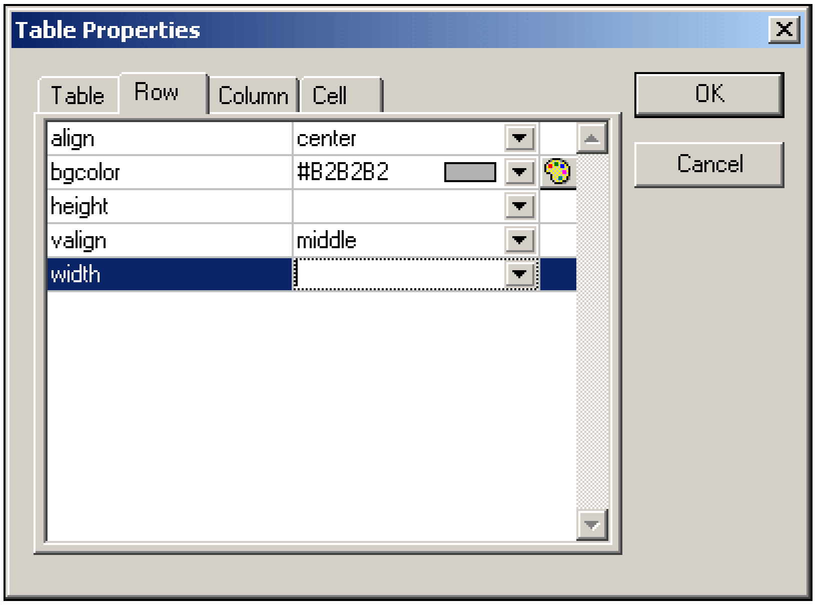


align	
bgcolor	
border	1
border-collapse	
cellpadding	
cellspacing	
frame	
height	
page-break-after	
page-break-before	
rules	
table-layout	

2. Set the cellspacing and cellpadding properties to "0". Your table will now look like this:

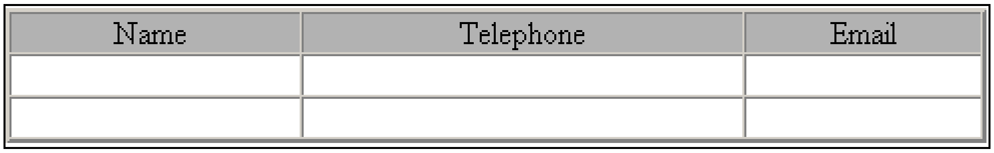


3. Place the cursor in the first row to format it, and click the (Table Properties) icon. Click the **Row** tab.



align	center
bgcolor	#B2B2B2
height	
valign	middle
width	

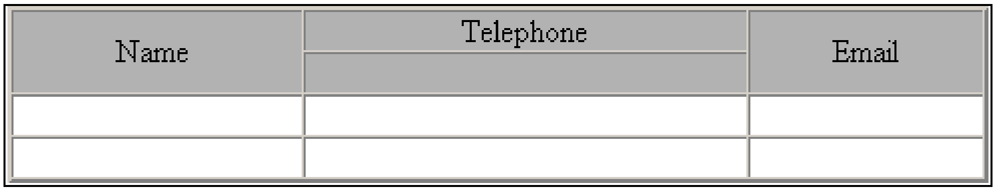
Since the first row will be the header row, set a background color to differentiate this row from the other rows. Note the Row properties that have been set in the figure above. Then enter the column header text. Your table will now look like this:



Name	Telephone	Email

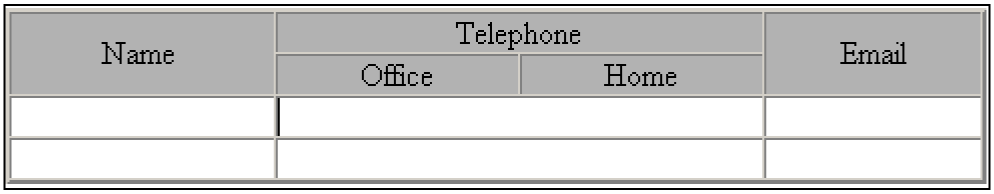
Notice that the alignment is centered as specified.

4. Now, say you want to divide the "Telephone" column into the sub-columns "Office" and "Home", in which case you would need to join cells. Place the cursor in the "Telephone" cell, and click the SPLIT (Split vertically) icon. Your table will look like this:



Name	Telephone	Email

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



Name	Telephone		Email
	Office	Home	

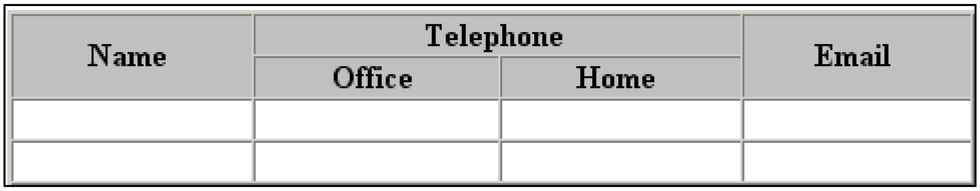
Now you will have to vertically split each cell in the "Telephone" column.

You can also add and delete columns and rows, and vertically align cell content, using the table-editing icons. The XML table editing icons are described in the User Reference, in the section titled "XML Table Icons".

Moving among cells in the table
To move among cells in the XML table, use the Up, Down, Right, and Left arrow keys.

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

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



Name	**Telephone**		**Email**
	Office	**Home**	

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

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

XML Table Editing Icons

The commands required to edit XML tables are available as icons in the toolbar, and are listed below. Note that no corresponding menu commands exist for these icons and that they are not active in StyleVision.

For a full description of when and how XML tables are to be used, see XML tables.

Insert table

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

Delete table

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

Append row

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

Append column

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

Insert row

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

Insert column

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

Join cell left

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

Join cell right

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

Join cell below

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

Join cell above

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

Split cell horizontally

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

Split cell vertically

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

Align top

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

Center vertically

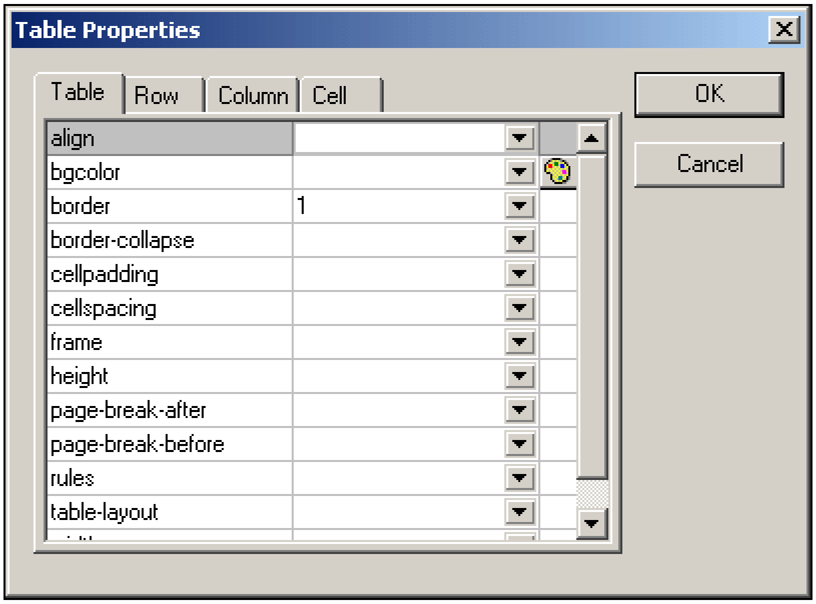
This command centers the cell contents.

Align bottom

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

Table properties

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



align	
bgcolor	
border	1
border-collapse	
cellpadding	
cellspacing	
frame	
height	
page-break-after	
page-break-before	
rules	
table-layout	

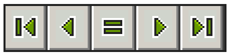
Editing a DB

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

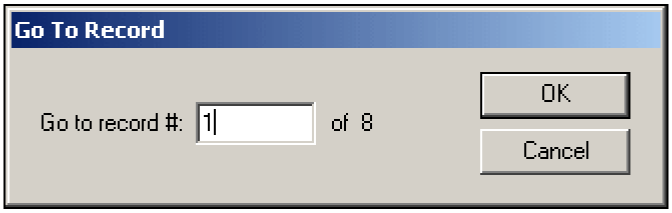
- The number of records in a DB table that are displayed in Authentic View may have been deliberately restricted by the designer of the StyleVision Power Stylesheet in order to make the design more compact. In such cases, only that limited number of records is initially loaded into Authentic View. Using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

Navigating a DB Table

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



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



Go to record #: 1 of 8

To navigate a DB table, click the required button.

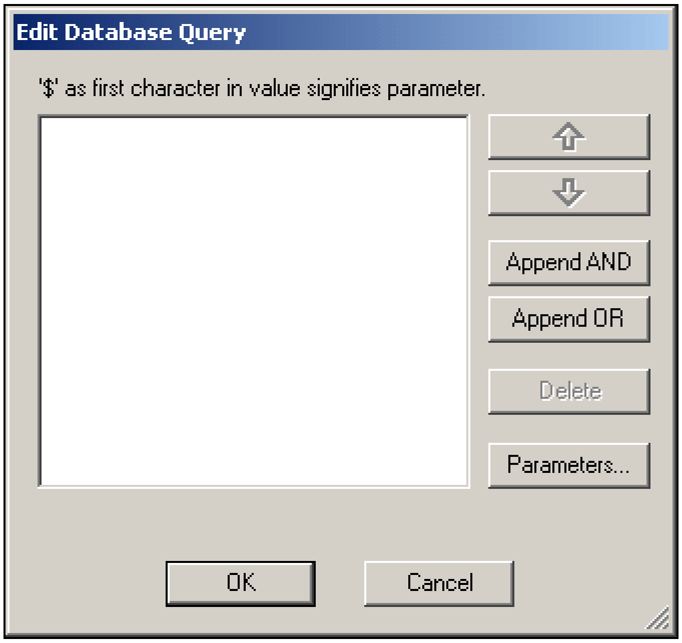
DB Queries

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

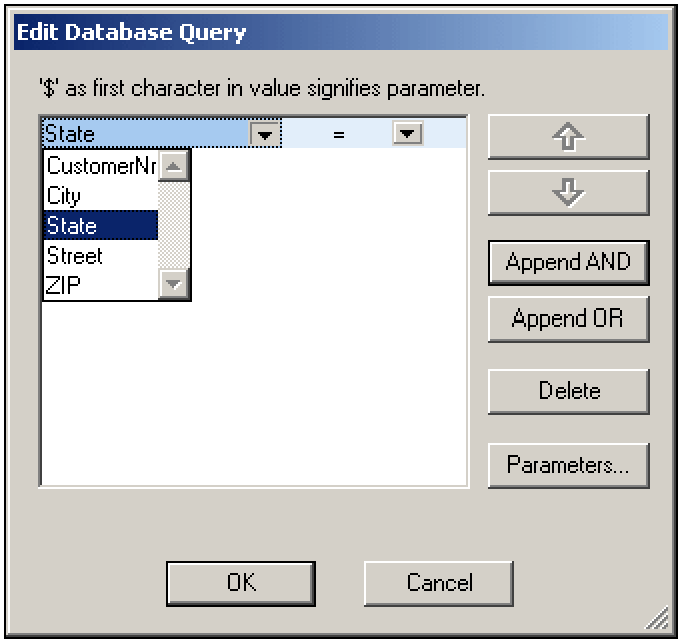
Please note: If you get a message saying that too many tables are open, then you can reduce the number of tables that are open by using a query to filter out some tables.

To create and submit a query:

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



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



'$' as first character in value signifies parameter.

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

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

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

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

Using parameters with DB Queries

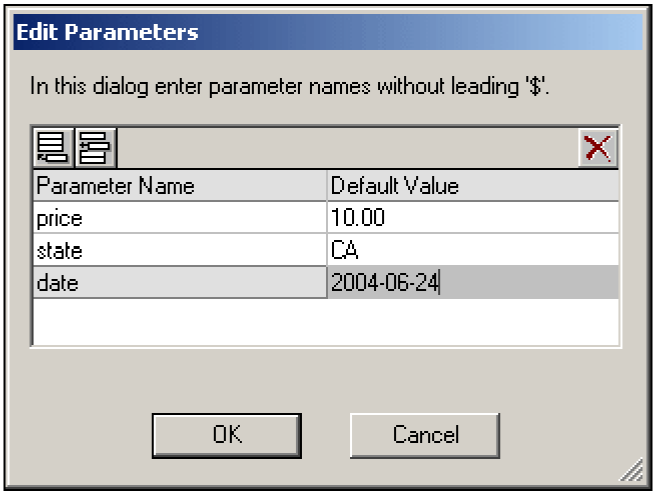
You can enter the name of a **parameter** as the value of an expression when creating queries. Parameters are variables that can be used instead of literal values in queries. You first declare the parameter and its value, and then use the parameter in expressions. This causes the value of the parameter to be used as the value of that expression. The parameters that you add in the Edit Parameters dialog can be parameters that have already been declared for the stylesheet. In this case, the new value overrides the value in the stylesheet.

Parameters are useful if you wish to use a single value in multiple expressions.

Declaring parameters from the Edit DB Query dialog

To declare parameters:

1. Click the **Parameters...** button in the Edit Database Query dialog. This opens the **Edit Parameters** dialog (*see screenshot*).



Edit Parameters

In this dialog enter parameter names without leading '$'.

Parameter Name	Default Value
price	10.00
state	CA
date	2004-06-24

2. Click Append or Insert .
3. Type in the name and value of the parameter in the appropriate fields.

Please note: The Edit Parameters dialog contains **all** the parameters that have been defined for the stylesheet. While it is an error to use an undeclared parameter in the StyleVision Power Stylesheet, it is not an error to declare a parameter and not use it.

Using parameters in queries

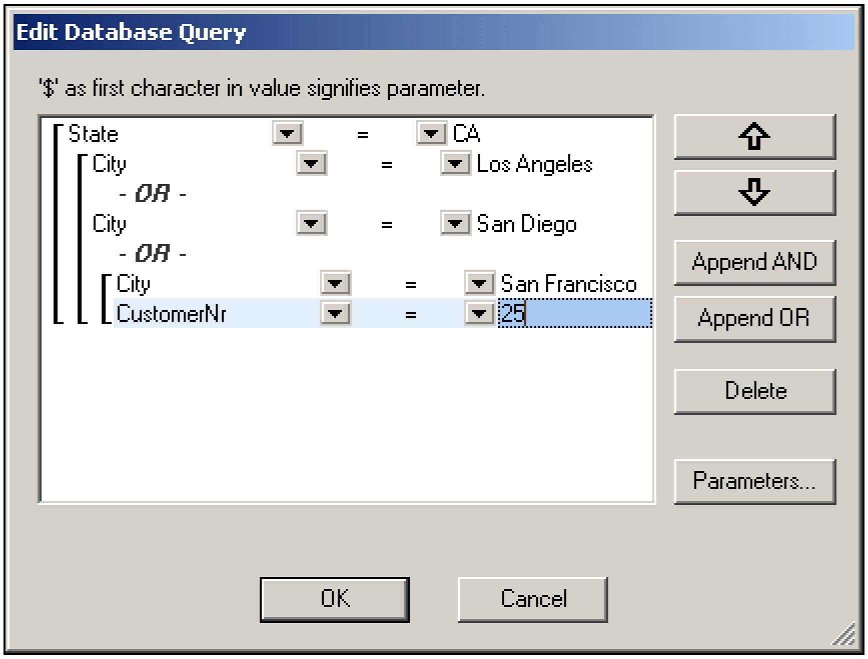
To enter the name of a parameter as the value of an expression:

- Type `$` into the value input field followed (without any intervening space) by the name of the parameter in the Edit Database Query dialog.

Please note: If the parameter has already been declared, then the entry will be colored green. If the parameter has not been declared, the entry will be red, and you must declare it.

Re-ordering criteria in DB Queries

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



Edit Database Query

'$' as first character in value signifies parameter.

```
State          =  CA
City           =  Los Angeles
  - OR -
City           =  San Diego
  - OR -
City           =  San Francisco
CustomerNr     =  25
```

The DB Query shown in the screenshot above may be represented in text as:

```
State=CA AND (City=Los Angeles OR City=San Diego OR (City=San
Francisco AND CustomerNr=25))
```

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

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

The following points should be noted:

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

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

Modifying a DB Query

To modify a DB Query:

1. Click the Query button . The Edit Database Query dialog box opens. You can now edit the expressions in any of the listed criteria, add new criteria, re-order criteria, or delete criteria in the DB Query.
2. Click **OK**. The data from the DB is automatically re-loaded into StyleVision so as to reflect the modifications to the DB Query.

Modifying a DB Table

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

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

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

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

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

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

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

Please note:

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

Deleting a record

To delete a record:

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

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

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

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

Working with Dates

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

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

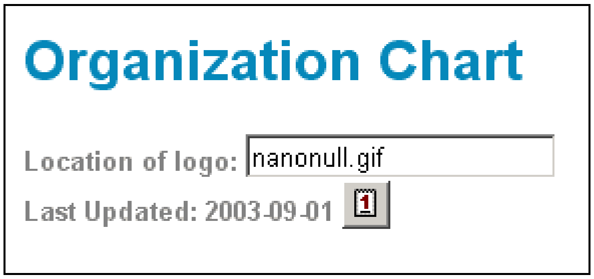
The method the Authentic View user will use is defined in the SPS. Both methods are described in the two sub-sections of this section.

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

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

Date Picker

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

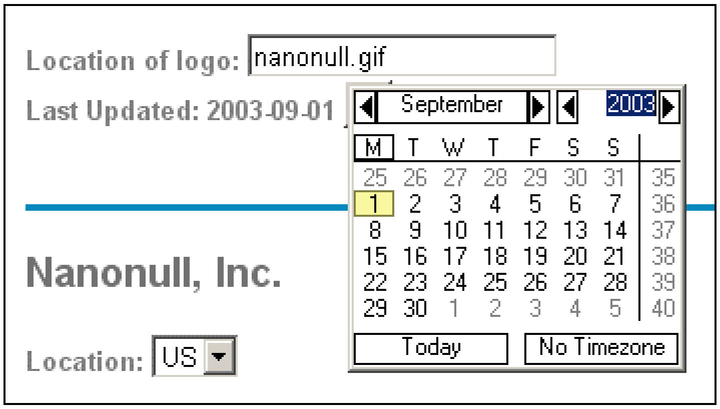


Organization Chart

Location of logo: nanonull.gif

Last Updated: 2003-09-01

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



Location of logo: nanonull.gif

Last Updated: 2003-09-01

Nanonull, Inc.

Location: US

September 2003

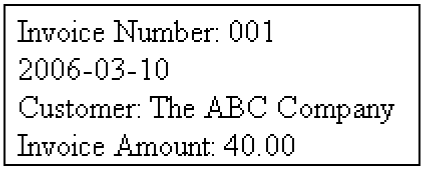
M	T	W	T	F	S	S	
25	26	27	28	29	30	31	35
1	2	3	4	5	6	7	36
8	9	10	11	12	13	14	37
15	16	17	18	19	20	21	38
22	23	24	25	26	27	28	39
29	30	1	2	3	4	5	40

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

Text Entry

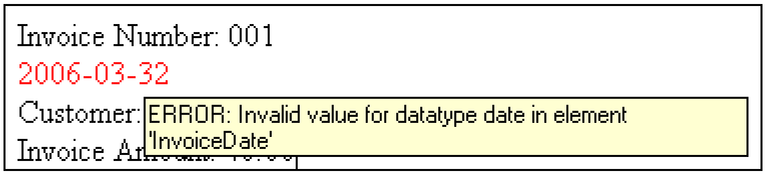
For date fields that do not have a Date Picker (*see screenshot*), you can edit the date directly by typing in the new value.

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



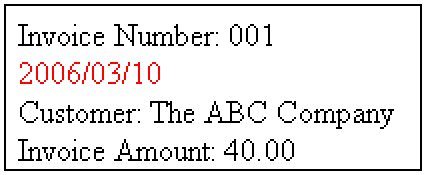
Invoice Number: 001
2006-03-10
Customer: The ABC Company
Invoice Amount: 40.00

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



Invoice Number: 001
2006-03-32
Customer: ERROR: Invalid value for datatype date in element 'InvoiceDate'
Invoice Am

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



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

Defining Entities

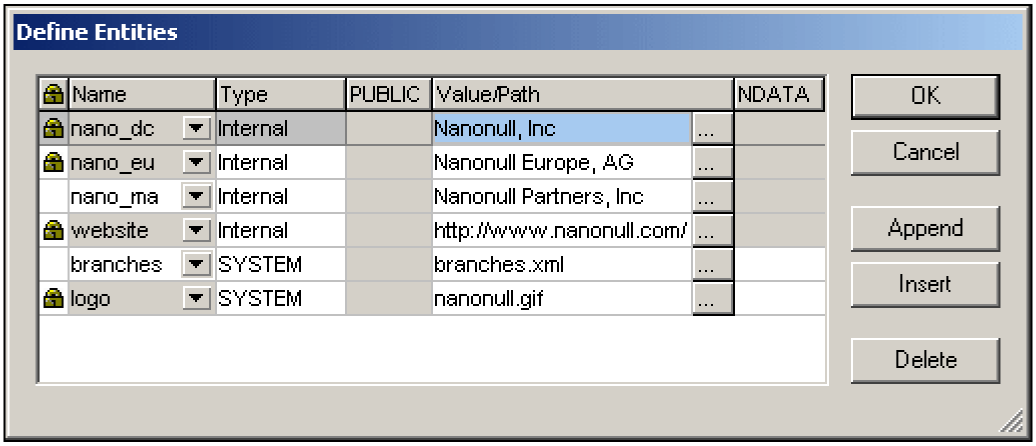
You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the **Insert Entity** submenu of the context menu. When you double-click on an entity in the Entities Entry Helper,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

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

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

To define an entity:

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



🔒	Name	Type	PUBLIC	Value/Path	NDATA
🔒	nano_dc	Internal		Nanonull, Inc	
🔒	nano_eu	Internal		Nanonull Europe, AG	
	nano_ma	Internal		Nanonull Partners, Inc	
🔒	website	Internal		http://www.nanonull.com/	
	branches	SYSTEM		branches.xml	
🔒	logo	SYSTEM		nanonull.gif	

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

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

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

6. The NDATA entry tells the processor that this entity is not to be parsed but to be sent to the appropriate processor. The NDATA field should therefore be used with unparsed entities only.

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

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

Limitations of entities

- An entity contained within another entity is not resolved, either in the dialog, Authentic View, or XSLT output, and the ampersand character of such an entity is displayed in its escaped form, i.e. `&amp;`.
- External entities are not resolved in Authentic View, except in the case where an entity is an image file and it is entered as the value of an attribute of type `ENTITY` or `ENTITIES`. Such entities are resolved when the document is processed with an XSLT generated from the SPS.

Images in Authentic View

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

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

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

Keystrokes in Authentic View

Enter (Carriage Return) Key

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

Please note: The **Return** key does **not** insert a carriage return/line feed, i.e. it does not jump to a new line. This is the case even when the cursor is inside a text node, such as paragraph.

Chapter 11

Reference

11 Reference

This section contains a complete description of StyleVision toolbars, Design View, and menu commands. It is divided into the following broad parts:

- A description of all the toolbars with their icons, as well as a description of how to customize the views of the toolbars.
- Descriptions of symbols used in Design View and of the Edit XPath Expression dialog.
- All menu commands.

While the User Reference section contains a description of individual commands, the mechanisms behind various StyleVision features are explained in detail in the various Procedures sections:

- High-Level Procedures
- Content Editing Procedures
- Presentation Procedures
- Additional Editing Procedures

For command line usage, see Command Line Interface: StyleVisionBatch.

11.1 Toolbars

A number of StyleVision commands are available as toolbar shortcuts, organized in the following toolbars:

- Formatting
- Table
- Authentic
- Template Filter
- Standard

The icons in each toolbar are listed in the sub-sections of this section, each with a brief description of the corresponding command.

Positioning the toolbars

A toolbar can float freely on the screen or can be placed in a toolbar area along any edge of the GUI. Toolbars are most commonly placed along the top edge of the GUI, just below the Menu bar. However, they can also be placed along the side or bottom edges of the GUI.

To position a toolbar in a toolbar area, do the following:

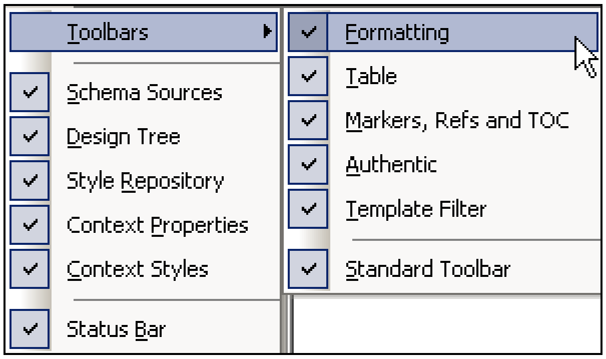
1. Grab the toolbar by its handle (if the toolbar is already in a toolbar area) or by its title bar (if the toolbar is floating).
2. Drag the toolbar to the desired toolbar area, if it exists, and drop it at the desired location in that toolbar area. If no toolbar area exists at the edge along which you wish to place the toolbar, dragging the toolbar to that edge will automatically create a toolbar area there when the toolbar is dropped.

To make a toolbar float freely grab it by its handle, drag it away from the toolbar area, and drop it anywhere on the screen except at an edge or in an existing toolbar area.

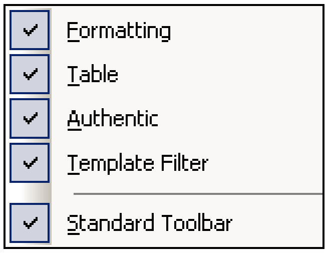
Switching the display of toolbars on and off

The display of individual toolbars can be switched on and off using any of the following three methods:

- In the **View | Toolbars** menu (*screenshot below*), select or deselect a toolbar to, respectively, show or hide that toolbar.



- Right-click any toolbar area to display a context menu (*screenshot below*) that allows you to toggle the display of individual toolbars on and off.

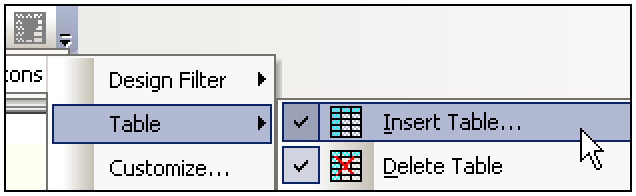


- In the Toolbars tab of the Customize dialog (**Tools | Customize**), toggle the display of individual toolbars on or off by clicking a toolbar's check-box. When done, click the **Close** button to close the dialog.

Adding and removing toolbar buttons

Individual toolbar buttons can be added to or removed from a toolbar, that is, they can be made visible or be hidden. To add or remove a button from a toolbar, do the following:

1. In the toolbar where the button to be added or removed is, click the **More Buttons** button (if the toolbar is in a toolbar area) or the **Toolbar Options** button (if the toolbar is a floating toolbar). The **More Buttons** button is an arrowhead located at the right-hand side of the toolbar (in horizontal toolbar areas) or at the bottom of the toolbar (in vertical toolbar areas). The Toolbar Options button is an arrowhead located at the right-hand side of the floating toolbar.
2. In the **Add or Remove Buttons** menu that pops up, place the cursor over the **Add or Remove Buttons** menu item (*screenshot below*). This rolls out a menu which contains the names of the toolbars in that toolbar area plus the **Customize** menu item (*screenshot below*).



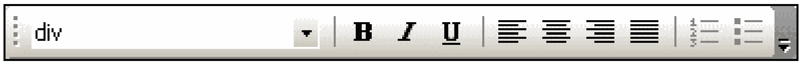
3. Place the cursor over the toolbar that contains the toolbar button to be added or removed (*screenshot above*).
4. In the menu that rolls out (*screenshot above*), click on the name of the toolbar button to add or remove from the toolbar.
5. Clicking the Customize item pops up the Customize dialog.

The **Reset Toolbar** item below the list of buttons in each toolbar menu resets the toolbar to the state it was in when you downloaded StyleVision. In this state, all buttons for that toolbar are displayed.

Note: The buttons that a toolbar contains are preset and cannot be disassociated from that toolbar. The process described above displays or hides the button in the toolbar that is displayed in the GUI.

Formatting

The **Formatting toolbar** (*screenshot below*) contains commands that assign commonly used inline and block formatting properties to the item/s selected in the SPS.



Predefined HTML formats
The HTML format selected from the dropdown list is applied to the selection in Design View. For example, a selection of `div` applies HTML's `<div>` element around the current selection in Design View.

Text properties
The bold, italic, and underline inline text properties can be directly applied to the current selection in Design View by clicking on the appropriate button.

Alignment
Alignment properties (left-aligned, centered, right-aligned, and justified) can be directly applied to the selection in Design View.

Lists
Lists can be inserted at the cursor insertion point, or the selection in the SPS can be converted to a list.

Table

The **Table toolbar** contains commands to structure and format Static SPS Tables in Design View. These commands are shown in the screenshot below (which is that of the menu for adding and removing Table toolbar buttons).

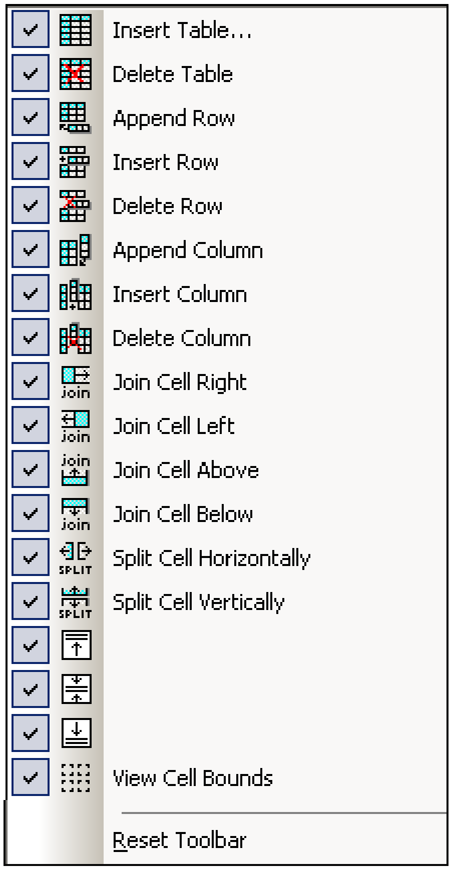


Table operations

The **Insert Table** button inserts a **static table** anywhere in the SPS design. Placing the cursor in a static or dynamic table and clicking **Delete Table** deletes that table.

Row and Column operations

Rows and columns in any SPS table (static or dynamic) canSPS be inserted, appended, or deleted with reference to the cursor location. Rows and columns are inserted before the current cursor location or appended after all rows/columns. The row/column in which the cursor is can also be deleted. These operations are achieved with the **Insert Row/Column**, **Append Row/ Column**, or **Delete Row/Column** buttons.

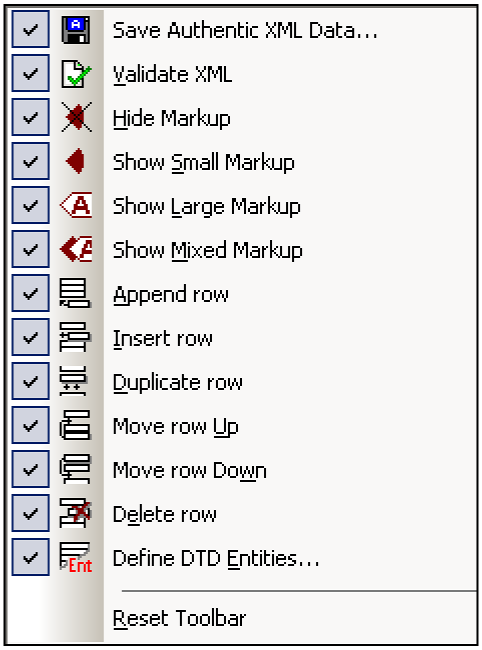
Cell operations

An SPS table cell in which the cursor is located can be joined to any one of the four cells around it. The joining operation is similar to that of spanning table cells in HTML. The buttons to be used for these operations are **Join Cell Right/Left/Above/Below**. Also, an SPS table cell in which the cursor is located can be split, either horizontally or vertically, using the **Split Cell Horizontally** and **Split Cell Vertically** buttons, respectively. SPS table cell content can be aligned vertically at the top, in the middle, and at the bottom. The display of cell borders can be switched on and off with the **View Cell Bounds** toggle.

Authentic

The **Authentic toolbar** contains commands for customizing Authentic View and editing XML documents in Authentic View. These commands are shown in the screenshot below (the menu for adding and removing Authentic toolbar buttons).

All these features are available to the Authentic View user. They enable you, as the SPS designer, to test the SPS using features at the Authentic View users's disposal.



Validating and saving XML documents

While editing an XML document in Authentic View, you can check the validity of the Working XML File by using the **Validate XML** button. Editing changes can be saved to the Working XML File with the **Save Authentic XML Data** button.

Markup tags in Authentic View

In Authentic View, the display of markup tags can be customized. Markup tags can be hidden (**Hide Markup**), can be shown with node names (**Show Large Markup**), without node names (**Show Small Markup**), or with any of these three options for individual nodes (**Show Mixed Markup**).

Editing of dynamic tables in Authentic View

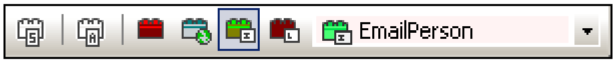
In Authentic View, row operations can be performed on dynamic tables. Rows can be inserted, appended, and duplicated, as well as be moved up and down, using the appropriate toolbar buttons (**Insert Row**, **Append Row**, **Duplicate Row**, **Move Row Up**, **Move Row Down**, and **Delete Row**).

Defining DTD Entities

Entities can be defined at any time while editing the Working XML File in Authentic View. Clicking the **Define DTD Entities** button pops up the Define DTD Entities dialog. (*See Authentic | Define Entities for details of usage.*)

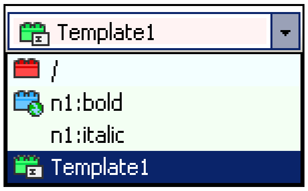
Design Filters

The **Design Filter toolbar** (*screenshot below*) contains commands that enable you to filter which templates are displayed in the design. Each icon in the toolbar is explained below.



Icon	Command	Description
	Show only one template	Shows the selected template only. Place the cursor in a template and click to show that template only.
	Show all template types	Shows all templates in the SPS (main, global, named, and layout) .
	Show/Hide main template	Toggles the display of the main template on and off.
	Show/Hide global templates	Toggles the display of global templates on and off.
	Show/Hide Design Fragments	Toggles the display of Design Fragments on and off.
	Show/Hide layout templates	Toggles the display of layout templates on and off.

The Design Filter combo box (*screenshot below*) displays a list of all the templates in the SPS.



Selecting a template in the combo box causes the template to be selected in the design. The combo box, therefore, enables you to quickly navigate to the desired template in the design, which is useful if the design has several templates, some of which might be currently hidden.

Standard

The **Standard toolbar** contains buttons for commands that provide important file-related and editing functionality. These icons are listed below with a brief description. For a fuller description of a command, click the command to go to its description in the Reference section.

Icon	Command	Shortcut	Description
▾	New from XML Schema / DTD	Ctrl+N	Creates a new SPS document based on a schema. Clicking the dropdown arrow enables you to create the SPS from a DB or an HTML document, or an empty SPS.
📂	Open	Ctrl+O	Opens an existing SPS document.
💾	Save Design	Ctrl+S	Saves the active SPS document.
💾	Save All	Ctrl+Shift+S	Saves all open SPS documents.
🖨	Print	Ctrl+P	Prints the Authentic View of the Working XML file.
🔍	Print Preview		Displays a print preview of the Authentic View of the Working XML File.
✂	Cut	Shift+Del	Cuts the selection and places it in the clipboard.
📄	Copy	Ctrl+C	Copies the selection to the clipboard.
📋	Paste	Ctrl+P	Pastes the clipboard item to the cursor location.
✗	Delete	Del	Deletes the selection.
↶	Undo	Alt+ Backspace	Undoes an editing change. An unlimited number of Undo actions can be performed at a time.
↷	Redo	Ctrl+Y	Redoes an undo.
🔭	Find	Ctrl+F	Finds text in Authentic View and Output Views.
🔭	Find Next	F3	Finds the next occurrence of the searched text.
🌐	Hyperlink		Inserts a static or dynamic hyperlink in the SPS.
🖼	Image		Inserts a static or dynamic image in the SPS.
1	Auto-add Date Picker		Adds a date-picker control for each node that is of XML Schema datatype `xsd:date` or `xsd:dateTime`.
▶	Auto-add DB Control		Adds a DB control for every DB table element created in the SPS.

Icon	Command	Shortcut	Description
XSLT 1.0	XSLT 1.0		Sets XSLT 1.0 as the stylesheet language.
XSLT 2.0	XSLT 2.0		Sets XSLT 2.0 as the stylesheet language.
ABC✓	Spelling		Runs a spelling check on the SPS document.
	Edit DB Filter		Displays a dialog that enables editing of DB Filters for filtering data imported from a DB.
	Clear DB Filter		Clears the DB Filter for the selected element.
	Edit Stylesheet Parameters		Displays the Edit Parameters dialog, in which you can edit stylesheet parameters.

11.2 Design View

The Design View is where the SPS is structured and where presentation properties are assigned. It provides you with a graphical representation of your design. The symbols that are used to denote the various components of the SPS are important for understanding the structure and layout of the SPS. These symbols are explained in the Symbols sub-section of this section. A key mechanism used to access nodes in XML documents is XPath, and a number of StyleVision features use XPath. A dialog used in common by all these features is the Edit XPath Expression dialog, in which you can build XPath expressions. The Edit XPath Expressions dialog is explained in detail in the XPath Dialog sub-section of this section.

Symbols

An SPS design will typically contain several types of component. Each component is represented in the design by a specific symbol. These symbols are listed below and are organized into the following groups:

- Nodes in the XML document
- XML document content
- Data-entry devices
- Predefined formats
- XPath objects
- URI objects

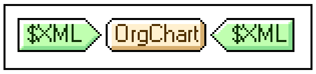
Each of these component types can:

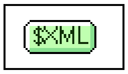
- be moved using drag and drop;
- be cut, copied, pasted, and deleted using (i) the commands in the View menu, or (ii) the standard Windows shortcuts for these commands;
- have formatting applied to it;
- have a context menu pop up when right-clicked.

Nodes in the XML document
Element and attribute nodes in the XML document are represented in the SPS design document by tags. Each node has a start tag and end tag. Double-clicking either the start or end tag collapses that node. When a node is collapsed all its contents are hidden. Double-clicking a collapsed node expands it and displays its content.

The following types of node are represented:

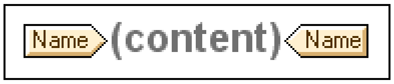
- ***Document node***

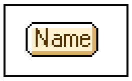




The **document node** (indicated with `$XML`) represents the XML document as a whole. It is indicated with a green `$XML` tag when the schema source is associated with an XML document, and with `$DB` when the schema source is associated with a DB. The document node in the screenshot at left is expanded and contains the `OrgChart` element, which is collapsed. The document node in the screenshot at right is collapsed; its contents are hidden.

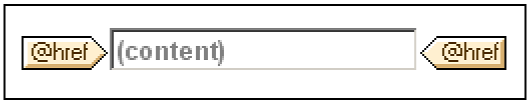
- ***Element node***

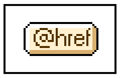




An **element node** is inserted together with all its ancestor elements if the ancestors are not present at the insertion point. In the screenshot above, the `Name` element node is shown expanded *(left)* and collapsed *(right)*.

- ***Attribute node***





An **attribute node** is inserted together with all its ancestor elements if the ancestors are not present at the insertion point. Attribute names contain the prefix `@`. In the screenshot above, the `href` attribute node is shown expanded *(left)* and collapsed *(right)*.

XML document content

XML document content is represented by two placeholders:

- `(contents)`
- `(rest-of-contents)`

The **`contents`** placeholder represents the contents of a single node. All the text content of the node is output. If the node is an attribute node or a text-only element node, the value of the node is output. If the node is an element node that contains mixed content or element-only content, the text content of all descendants is output. In XSLT terms, the `contents` placeholder is equivalent to the `xsl: apply-templates` element with its `select` attribute set for that node..

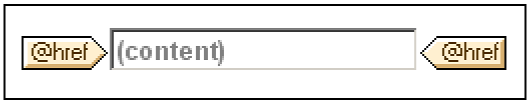
Note: When applied to an element node, the `contents` placeholder does not output the values of attributes of that element. To output attribute nodes, you must explicitly include the attribute in the template (main or global).

The **`rest-of-contents`** placeholder applies templates to the rest of the child elements of the current node. The template that is applied for each child element in this case will be either a global template (if one is defined for that element) or the default template for elements (which simply outputs text of text-only elements, and applies templates to child elements). For example, consider an element `book`, which contains the child elements: `title`, `author`, `isbn`, and `pubdate`. If the definition of `book` specifies that only the `title` child element be output, then none of the other child elements (`author`, `isbn`, and `pubdate`) will be output when this definition is processed. If, however, the definition of `book` includes the `rest-of-contents` placeholder after the definition for the `title` element, then for each of the other child elements (`author`, `isbn`, and `pubdate`), a global template (if one exists for that element), or the default template for elements, will be applied.

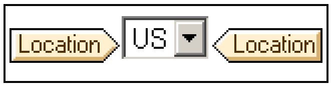
Data-entry devices

In order to aid the Authentic View user edit the XML document correctly and enter valid data, data-entry devices can be used in the design. You can assign any of the following data-entry devices to a node:

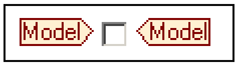
- ***Input fields (single line or multi-line)***



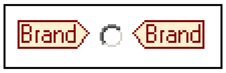
- ***Combo boxes***



- ***Check boxes***

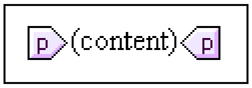


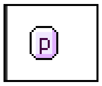
- ***Radio buttons***



These tags can be collapsed and expanded by double-clicking an expanded and the collapsed tag, respectively. For a detailed description of how each of these data-entry devices is used, see Data-Entry Devices.

Predefined formats
Predefined formats are shown in mauve tags, which can be expanded/collapsed by double-clicking.

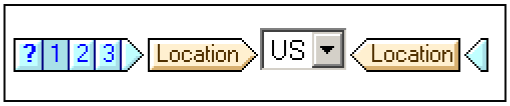




The screenshot above shows tags for the predefined format `p ( para)` , expanded *(at left)* and collapsed *(at right)*. To apply a predefined format, highlight the items around which the predefined format is to appear (by clicking a component and/or marking text), and insert the predefined format.

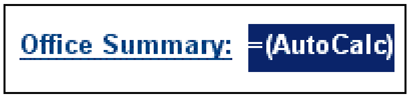
XPath objects
StyleVision features two mechanisms that use XPath expressions:

- ***Conditional templates***



Condition tags are blue. The start tag contains cells. The leftmost cell contains a question mark. Other cells each contain either (i) a number, starting with one, for each `when` condition; and/or (ii) an asterisk for the optional `otherwise` condition. A condition branch can be selected by clicking it. The number of the selected condition branch is highlighted in the start tag, and the template for that branch is displayed (within the start and end tags of the condition). The XPath expression for the selected condition branch is also highlighted in the Design Tree. Note that tags for conditions cannot be expanded/collapsed.

- ***Auto-Calculations***

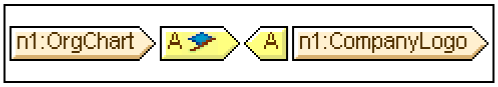


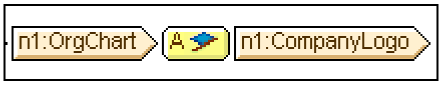
Auto-Calculations are represented in Design View by the `=( AutoCalc)` object (*see screenshot above*). The XPath expression for the selected Auto-Calculation is highlighted in the Design Tree. The dialog to edit the Auto-Calculation is accessed via the Properties entry helper.

URI objects
There are three URI-based objects that can be inserted in a design:

- ***Images***
  If an image is inserted in the SPS design and can be accessed by StyleVision, it becomes visible in Design View. If it cannot be accessed, its place in the SPS is marked by an image placeholder.

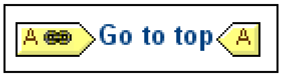
- ***Bookmarks (Anchors)***

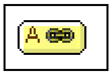




Bookmark tags are yellow and indicated with the character A (*screenshots above*). A bookmark is created with the command **Insert | Bookmark**, and can be empty or contain content. Content must always be inserted after the anchor is created. Anchor tags can be expanded (*screenshot above left*) or collapsed (*screenshot above right*).

- ***Links***

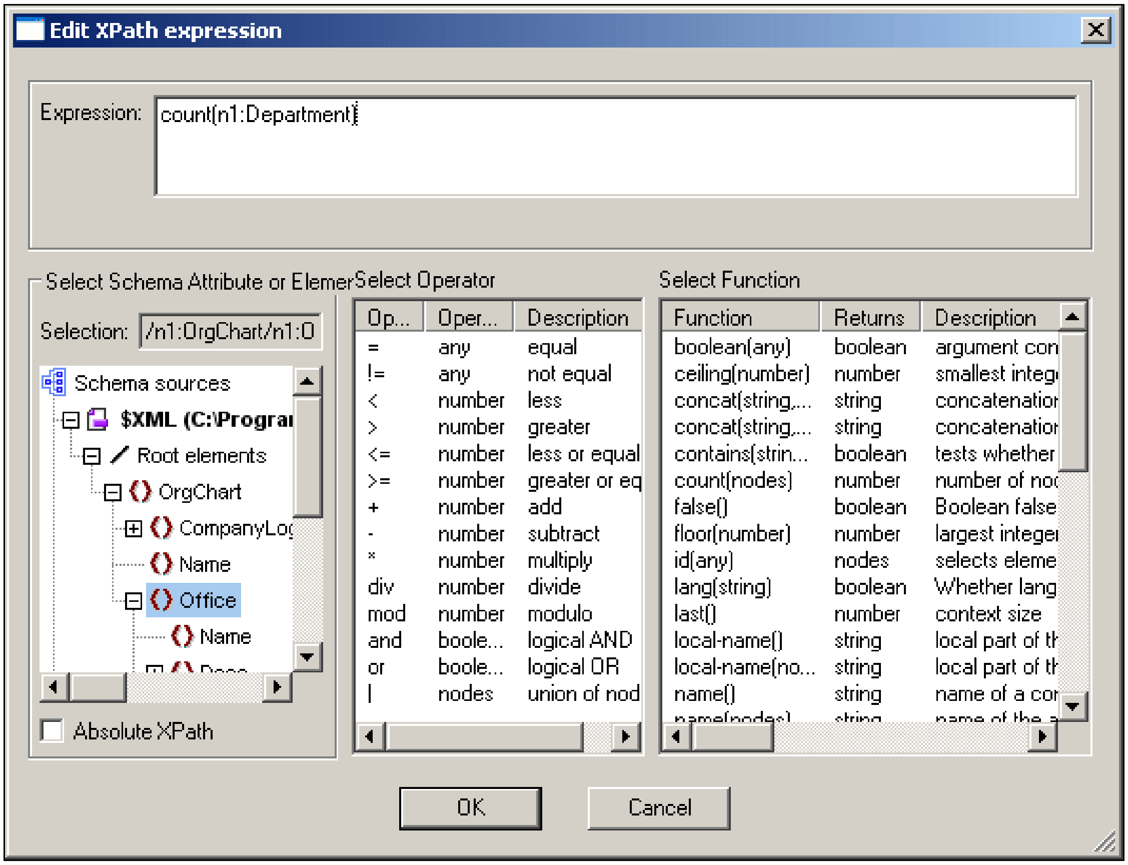




Link tags are yellow and indicated with the character A (*screenshots above*). A link is created with the command **Insert | Hyperlink**. The link item can be created before or after the link is created. If an item is to be created as a link, it should be selected and the link created around it. Link tags can be expanded (*screenshot above left*) or collapsed (*screenshot above right*).

XPath Dialog

The **Edit XPath Expression** dialog (*screenshot below*) is used to edit and assign XPath expressions for a range of features.



Expression: count(n1:Department)

Selection: /n1:OrgChart/n1:O

□ Absolute XPath

Op...	Oper...	Description
=	any	equal
!=	any	not equal
<	number	less
>	number	greater
<=	number	less or equal
>=	number	greater or eq
+	number	add
-	number	subtract
*	number	multiply
div	number	divide
mod	number	modulo
and	boole...	logical AND
or	boole...	logical OR
|	nodes	union of nod

Function	Returns	Description
boolean(any)	boolean	argument con
ceiling(number)	number	smallest integ
concat(string,...	string	concatenatior
concat(string,...	string	concatenatior
contains(strin...	boolean	tests whether
count(nodes)	number	number of noc
false()	boolean	Boolean false
floor(number)	number	largest intege
id(any)	nodes	selects eleme
lang(string)	boolean	Whether lang
last()	number	context size
local-name()	string	local part of th
local-name(no...	string	local part of th
name()	string	name of a cor

In the Edit XPath Expression dialog, you can (i) enter an expression in the Expression text box via the keyboard, or (ii) you can insert nodes, operators, and functions by double-clicking them from their respective lists. If the syntax of the XPath expression is correct, the expression is displayed in black; if incorrect, it is displayed in red. The lists for operators and functions automatically displays XPath 1.0 operators and functions or XPath 2.0 operators and functions according to the XSLT version selected for the SPS (XPath 1.0 for XSLT 1.0, and XPath 2.0 for XSLT 2.0).

The Edit XPath Expression dialog helps you to build XPath expressions in the following ways.

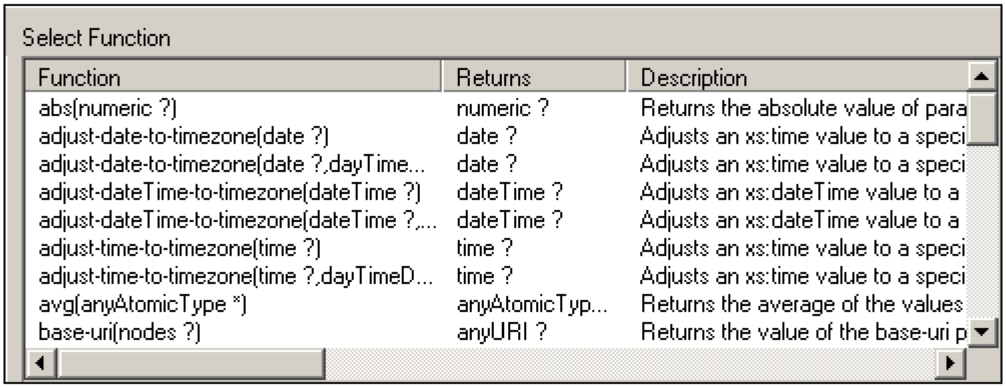
- **Context node**
  The node within which the Condition or Auto-Calculation will be inserted is shown in the Selection text box in the Select Schema Attribute or Element pane. The XPath expression is created relative to, or within the context of, this node. The Condition or Auto-Calculation is inserted at a location within this context, and the XPath expression will be evaluated with this node as its context.

- **Inserting a node from the schema in the expression**
  In the Select Schema Attribute or Element pane, the entire schema is displayed. You can insert a node from the schema into the XPath expression by double-clicking the required node. If the Absolute XPath check box is not checked, the selected node will be inserted with a location path expression that is relative to the context node. For

example, in the screenshot above, the `Location` element, which is a child of the `Office` element (the context node), has been inserted with a location path that is relative to the context node (that is, as `Location`). If the Absolute XPath check box were checked, the `Location` node would have been inserted as `/OrgChart/Office/Location`.

- **Inserting XPath operators**
  The Select Operator pane automatically lists XPath 1.0 or XPath 2.0 operators according to whether XSLT 1.0 or XSLT 2.0 has been selected as the XSLT version for the SPS. To insert an operator in the XPath expression, double-click the required operator.

- **Inserting XPath functions**
  The Select Function pane (*screenshot below*) is at the right of the Edit XPath Expression dialog and automatically lists XPath 1.0 or XPath 2.0 functions according to whether XSLT 1.0 or XSLT 2.0 has been selected as the XSLT version for the SPS. Each function is listed with its signature. If a function has more than one signature, that function is listed as many times as the number of signatures (see `adjust-date-to-timezone` in screenshot below). Arguments in a signature are separated by commas, and arguments can have an occurrence indicators (**`?`** indicates a sequence of zero or one items of the specified type; **`*`** indicates a sequence of zero or more items of the specified type). The functions list also includes the return type of that function and a brief description of the function.



Select Function

Function	Returns	Description
abs(numeric ?)	numeric ?	Returns the absolute value of para
adjust-date-to-timezone(date ?)	date ?	Adjusts an xs:time value to a speci
adjust-date-to-timezone(date ?,dayTime...	date ?	Adjusts an xs:time value to a speci
adjust-dateTime-to-timezone(dateTime ?)	dateTime ?	Adjusts an xs:dateTime value to a
adjust-dateTime-to-timezone(dateTime ?,...	dateTime ?	Adjusts an xs:dateTime value to a
adjust-time-to-timezone(time ?)	time ?	Adjusts an xs:time value to a speci
adjust-time-to-timezone(time ?,dayTimeD...	time ?	Adjusts an xs:time value to a speci
avg(anyAtomicType *)	anyAtomicTyp...	Returns the average of the values
base-uri(nodes ?)	anyURI ?	Returns the value of the base-uri p

To insert a function in the XPath expression, double-click the required function.

The Otherwise check box

The Otherwise check box below the input field for the XPath expression appears when a second or subsequent condition is being added to a conditional template. Checking the Otherwise check box inserts the optional Otherwise condition of a conditional template. For details of how to use the Otherwise condition, see Conditional Templates.

XPath expressions containing carriage returns / linefeeds

You can include carriage returns and/or linefeeds (CR/LFs) in the XPath expression in order to set part of the output on separate lines. However, in order for the CR/LF to be visible in the output, the component containing the XPath expression must be enclosed in the `pre` special paragraph type. An example of such an XPath expression is:

```
translate('a;b;c', ';', codepoints-to-string(13))
```

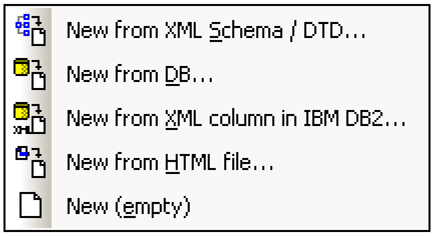
11.3 File Menu

The **File** menu contains commands for working with SPSs and related files. The following commands are available:

- New, to create a new SPS from a variety of sources.
- Open, Close, Close All, to open and close the active file.
- Save Design, Design As, All, which are commands to save the active SPS and all open SPS files.
- Save Authentic XML Data, enabled in Authentic View, it saves changes to the Working XML File.
- Save Generated Files, to save output files that can be generated using the SPS.
- Assign/Unassign Working XML File, to assign/unassign the Working XML File that will be used to generate the previews in StyleVision.
- Assign/Unassign Template XML File, to save output files that can be generated using the SPS.
- Edit Schema Definition in XMLSpy, to open the schema in Altova XMLSpy.
- Properties, to set the encoding of the output documents, the CSS compatibility mode of the browser, and how relative image paths in Authentic View should be resolved.
- Print Preview, Print, enabled in Authentic View and output view, these commands print what is displayed in the previews.
- Most Recently Used Files, Exit, respectively, to select a recently used file to open, and to exit the program.

New

Placing the cursor over the **New** command pops out a submenu (*screenshot below*) that enables you to create a new SPS document of one of four types:



New from XML Schema / DTD...
New from DB...
New from XML column in IBM DB2...
New from HTML file...
New (empty)

- A new SPS based on an XML Schema or DTD (**New from XML Schema / DTD**). The selected schema is added to the Schema Sources tree (in the Schema Sources entry helper) and, in Design View, the SPS is created with an empty main template.
- A new SPS based on an XML Schema generated from a DB you select (**New from DB**). The connection process is described in the section Connecting to a DB and Setting up the SPS. The SPS is created in Design View with an empty main template.
- A new SPS based on a user-defined schema you create node-by-node from an HTML file (**New from HTML File**). The user-defined schema (in the Schema Sources tree) will have a single document element (root element), and the HTML file is loaded in Design View.
- A new empty SPS (**New (empty)**). No schema is added to the Schema Sources tree and an empty main template will be created in Design View.

The new SPS document is given a provisional name of `SPSX.sps`, where `X` is an integer corresponding to the position of that SPS document in the sequence of new documents created since the application was started.

A schema in the Schema Sources entry helper has two immediate sub-items: XML files and Root elements. You select the required XML files and root elements after the schema has been added to the schema sources tree. The selections to be made are

- The Working XML File selection for Authentic View and output previews.
- The Template XML File selection for a starting template the Authentic View user works with when the SPS is opened.
- Root elements (document elements) to select what global elements can be used as document elements in the SPS.

How to make these selections is described in the section, Schema Sources entry helper.

Note: If your schema is not well-formed or is invalid, you may get an error message when you try to load it. In this case, open the schema and try to resolve the problem in it.

Open, Close, Close All

The **Open** (**Ctrl+O**) command 📂 allows you to open an existing SPS file. The familiar Open dialog of Windows systems is opened and allows you to select a file with an extension of `.sps`.

The **Close** command closes the currently active SPS document. Note that while several files can be open, only one is active. The active document can also be closed by clicking the **Close** button at the top right of the Main Window. If you have unsaved changes in the document, you will be prompted to save these changes.

The **Close All** command closes all the open SPS documents. If you have unsaved changes in an open document, you will be prompted to save these changes.

Save Design, Design As, All

The **Save Design (Ctrl+S)** command 💾 saves the currently open document as an SPS file (with the file extension `.sps`).

The **Save Design As** command shows the familiar Save As dialog of Windows systems. You can enter the name with which the active SPS file should be saved and the location where you want it saved. The newly saved file becomes the current file in StyleVision.

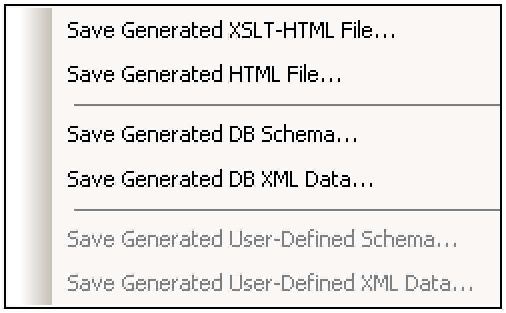
The **Save All (Ctrl+Shift+S)** command 💾 saves all the open SPS documents.

Save Authentic XML Data

In Authentic View, you can edit the Working XML File or DB related to the SPS. The **Save Authentic XML Data** command saves these modifications to the Working XML File or DB. Alternatively to editing the XML file in StyleVision, you can edit an XML document or DB in the Authentic View of Altova XMLSpy or Altova Authentic Desktop.

Save Generated Files

The **Save Generated Files** command pops up a submenu which contains options for saving the following files (*screenshot below*). For perspective on how the generated files fit into the genaeral usage procedure, see Usage Procedure | Generated Files.


Save Generated XSLT-HTML File...
Save Generated HTML File...
Save Generated DB Schema...
Save Generated DB XML Data...
Save Generated User-Defined Schema...
Save Generated User-Defined XML Data...

Save Generated XSLT-HTML File

The Save Generated XSLT-HTML File command generates an XSLT file for HTML output from your SPS. You can use this XSLT file subsequently to transform an XML document to HTML.

Save Generated HTML File

The Save Generated HTML File command generates an HTML file. This operation requires two input files:

- The Working XML File assigned to the currently active SPS file. If no Working XML File has been assigned, the **Save Generated HTML File** command is disabled. For DB-based SPSs, the automatically generated non-editable XML file is the Working XML File (see The DB Schema and DB XML files); you do not assign a Working XML File.
- An XSLT file, which is automatically generated from the currently active SPS file.

Save Generated DB Schema

When you connect to a DB in order to create a DB-based SPS, StyleVision generates and loads a temporary XML Schema based on the DB structure. The Save Generated DB Schema command enables you to save this generated XML Schema. Note that for XML DBs, StyleVision does not generate a schema file; it uses a schema file from the DB or some other external file location. Consequently, this command is not enabled for XML DBs.

Save Generated DB XML Data

The Save Generated DB XML Data command generates and saves an XML file that contains data from the DB in an XML structure conformant with the structure of the XML Schema generated from the DB. If DB Filters have been defined in the StyleVision Power Stylesheet, these are applied to the data import. Note that for XML DBs, StyleVision does not generate an XML file, but uses XML data in the XML columns of the XML DB. Consequently, this command is not enabled for XML DBs.

Save Generated User-Defined Schema

This command is activated when the SPS involves a user-defined schema. The schema you create in the Schema Sources entry helper is saved as an XML Schema with the `.xsd` extension.

Save Generated User-Defined XML Data
The data in the imported HTML file that corresponds to the user-defined schema is saved as an XML file. The corresponding data are the nodes in the HTML document (in Design View) that have been created as XML Schema nodes.

Assign/Unassign Working XML File

A Working XML File is an XML file that is assigned to an SPS in StyleVision in order to preview the Authentic View and output of the XML document in StyleVision. Without a Working XML File, the SPS in StyleVision will not have any dynamic XML data to process. The **Assign Working XML File** command assigns an XML file as the Working XML File to the SPS. Clicking the command, opens a dialog in which you can browse for the Working XML File. If a Working XML File is already assigned, clicking this command and assigning a file replaces the existing assignment with the new assignment.

Unassigning the Working XML File

The **Unassign Working XML File** command removes the assignment from the SPS. This command is enabled only when a Working XML File has been assigned for the active SPS.

Assign/Unassign Template XML File

The **Assign Template XML File** command assigns an XML file to the SPS (SPS). When the SPS is opened in Authentic View, it displays the data present in the assigned XML file according to the design of the SPS. The XML file therefore provides the starting data of a new XML file. In effect it provides the data for an template XML file, which is based on the SPS and can be saved under any name. We therefore call the XML file you assign to the SPS a Template XML File.

Assigning and changing a Template XML File

To assign a Template XML File, click **Authentic | Assign Template XML File**, select the required file, and save the SPS. To change the Template XML File, click **Authentic | Assign Template XML File**, select the new Template XML File, and save the SPS.

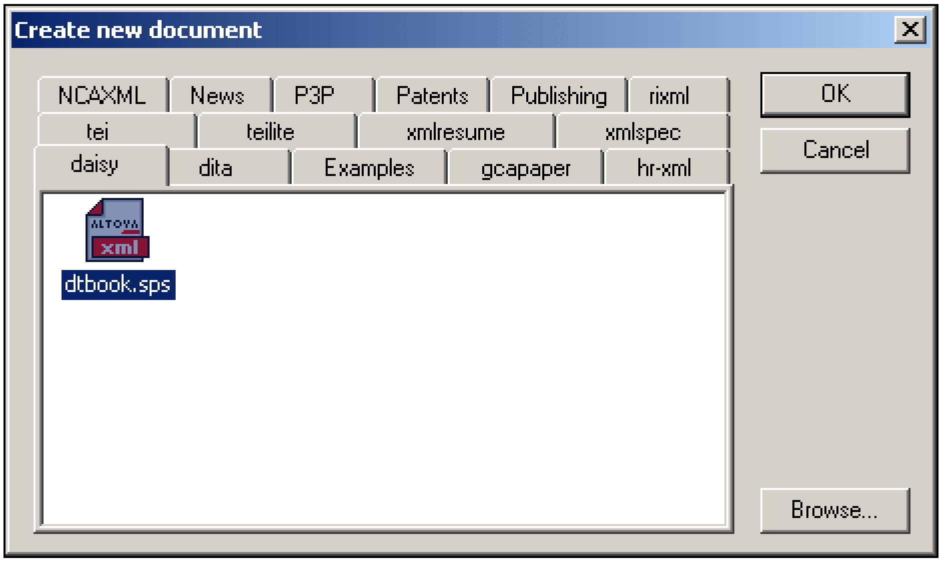
Note:

- The Template XML file must conform to the same schema as that of the SPS to which it is linked.
- The Template XML File can also be assigned and changed using the Template XML File entry of a schema in the Schema Sources entry helper.

Opening a template XML document in Authentic View

An Authentic View user can open a template XML document as follows:

1. In XMLSpy or Authentic Desktop, select **File | New** or **Authentic | New Document...**. This pops up the Create New Document dialog. The tabs in this dialog each represent a folder in the `sps/Template` folder within the XMLSpy or Authentic Desktop application folder. In each tab, the SPS files for the corresponding folder are displayed. These SPS files provide general use templates for popular schemas.



2. Select an Altova-supplied SPS template from one of the tabs or browse for the required SPS (using the **Browse...** button).
3. Click OK. A template based on the SPS and carrying starting data from a Template XML File, if any is associated with the SPS, is opened in Authentic View.

Adding folders and SPS files to the tabbed list in the Create a New Document dialog

You can add your own SPS files to the tabbed list in the Create a New Document dialog. Do this by copying/moving the folders containing the SPS files into the `sps/Template` folder within the XMLSpy or Authentic Desktop application folder. The added folder will then be displayed as a tab in the Create a New Document dialog. The SPS files which are in the folder will be displayed in the tab for that folder.

Unassigning the Template XML File

The **Unassign Template XML File** command removes the assignment from the SPS. This command is enabled only when a Template XML File has been assigned for the active SPS.

Edit Schema Definition in XMLSpy

This command opens the schema file in XMLSpy so that you can edit it. After you have edited and saved the modifications to your schema, you must reload the SPS so that the schema in the SPS is updated with the changes you made. This command is not relevant for DB-based SPSs and is grayed out when a DB-based SPS is opened.

Properties

The **Properties** command pops up the Properties dialog, in which you can set: (i) the encoding of output documents; and (ii) the CSS support level of the HTML and Authentic Views.

Encoding

In the Output Encoding pane you can select the encoding of your output documents. Changing the encoding in this dialog changes the encoding for the currently active SPS. You can also specify the default encoding for all subsequently created SPS documents; this is done in the Encoding tab of the Options dialog.

CSS support

CSS support in versions of Internet Explorer (IE) prior to IE 6.0 was incomplete and in some respects incorrectly interpreted. CSS support was enhanced and corrected in IE 6.0, and further improved in IE 7.0.

In IE 6.0 and later, an HTML document can be displayed either in **compatibility mode** (corresponding to the CSS support level in IE versions prior to IE 6.0), or in **standards-compliant mode** (corresponding to CSS support in IE 6.0 and later). Which mode is used depends on a switch coded in the HTML document. (See CSS Support in IE 6.0 and CSS Support in IE 7.0 for details.)

In an SPS, you can select the desired mode in the Properties dialog (*screenshot above*). The appropriate switch will be generated in the output document, and the specified level of support is immediately available in Authentic View and HTML Preview. Note that new SPS documents are created with Standards-Compliant Mode selected. SPS documents created in versions of Altova StyleVision prior to Altova StyleVision 2007 sp2 will be opened in Compatibility Mode; they can be re-saved in Standards-Compliant Mode (by selecting the Standards-Compliant option in the Properties dialog).

Note: When setting CSS styles in a document, you should be aware of what CSS support level has been set for the document output and you should assign CSS styles accordingly.

Relative image paths

You can set whether relative image paths in Design View and Authentic View should be relative to the SPS or to the XML file.

Print Preview, Print

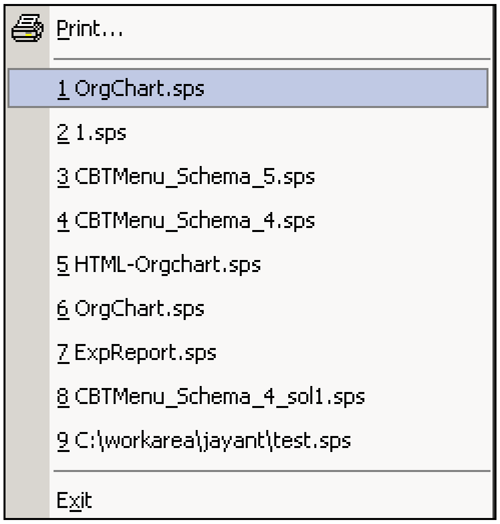
The **Print Preview** command is enabled in Authentic View (*Authentic View is supported in the Enterprise and Professional editions only*). It opens a window containing a preview of the Authentic View of the Working XML File. You can do the following in the Print Preview window:

- Navigate the pages of the preview using the page navigation buttons or by entering the page number in the Page text box.
- Change the zoom factor of the preview pages using the Zoom In and Zoom Out buttons or the combo box to select a zoom factor.
- Print the page using the Print button. You can set page properties by clicking the Page Setup buttons to get the Page Setup dialog.

The **Print** command is enabled in the Authentic View and output preview tabs. It prints out the selected view of the Working XML File according to the page setup for that view. Note that the page setup for Authentic View can be edited in the Page Setup dialog, which you access via the Print Preview window.

Most Recently Used Files, Exit

The list of most recently used files, shows the file name and path information for the nine most recently used files. Clicking one of these entries, causes that file to be opened in a new tab in the Main Window.



To access these files using the **keyboard**, press **ALT+F** to open the File menu, and then the number of the file you wish to open; for example, pressing **1** will open the first file in the list, **2** the second file, and so on.

The **Exit** command is used to quit StyleVision. If you have an open file with unsaved changes, you will be prompted to save these changes.

11.4 Edit Menu

The **Edit** menu contains commands that aid the editing of SPS and Authentic View documents. Besides the standard editing commands, such as Cut (Shift+Del), Copy (Ctrl+C), Paste (Ctrl+V), and Delete (Del), which are not described in this section, the following commands are available:

- Undo, Redo, Select All, to undo or restore your previous actions, and to select all content of the SPS.
- Find, Find Next, Replace, to find text in the SPS, Authentic View, and XSLT stylesheet previews, and to replace text in Authentic View.
- Edit DB Filter, Clear DB Filter, to access the Edit DB Filters dialog and clear DB Filters, respectively.
- Edit Stylesheet Parameters, to edit parameters declared globally for the SPS.
- Collapse/Expand Markup, to collapse and expand SPS design component tags.

Commands are also available via the context menu which appears when you right-click a component or right-click at a cursor insertion point. Additionally, some commands are available as keyboard shortcuts and/or toolbar icons. Note, however, that commands which are not applicable in a particular document view or at a given location are grayed out in the menu.

Undo, Redo, Select All

The **Undo (Ctrl+Z)** command enables you to undo an editing change. An unlimited number of Undo actions is supported. Every action can be undone and it is possible to undo one command after another till the first action that was made since the document was opened.

The **Redo (Ctrl+Y)** command allows you to redo any number of previously undone commands. By using the Undo and Redo commands, you can step backward and forward through the history of commands.

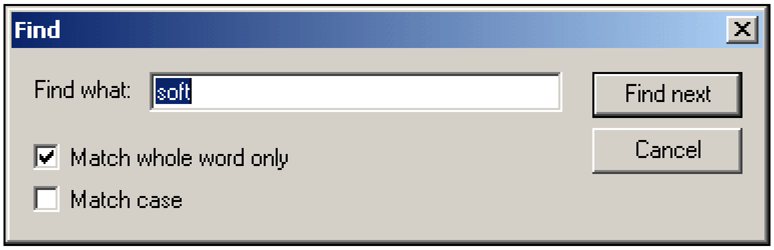
The **Select All** command selects the entire contents of the Design Document window.

Find, Find Next, Replace

The **Find (Ctrl+F)** command allows you to find words or fragments of words in the Design View, JavaScript Editor, Authentic View, and XSLT-for-HTML stylesheet.

Design View and Authentic View

Clicking the **Find** command in Design View or Authentic View pops up the following dialog:



Find what: soft

☑ Match whole word only

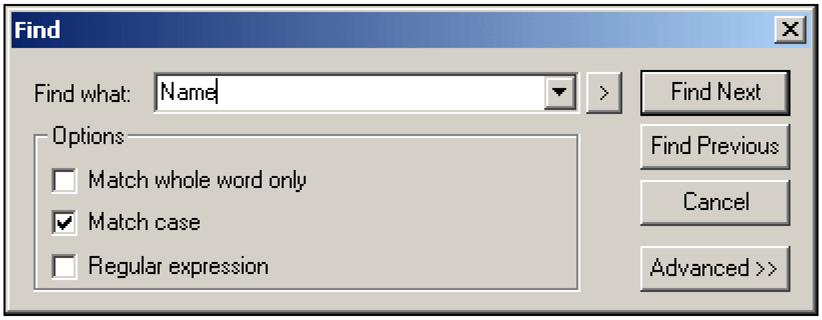
☐ Match case

Note the following:

- In Design View, the static data is searched, but not node names.
- In Authentic View, the dynamic data (XML data) is searched, not text from the static (XSLT) input.
- To match the entry with whole words, check "Match whole word only". For example, an entry of `soft` will find only the whole word `soft`; it will not find, for example, the `soft` in `software`.
- To match the entry with fragments of words, leave the "Match whole word only" check box unchecked. Doing this would enable you, for example, to enter `soft` and `software`.
- To make the search case-insensitive, leave the "Match case" checkbox unchecked. This would enable you to find, say, `Soft` with an entry of `soft`.

XSLT-for-HTML and JavaScript Editor

Clicking the **Find** command in the XSLT-for-HTML or JavaScript Editor tab pops up the following dialog:



Find what: Name

Options

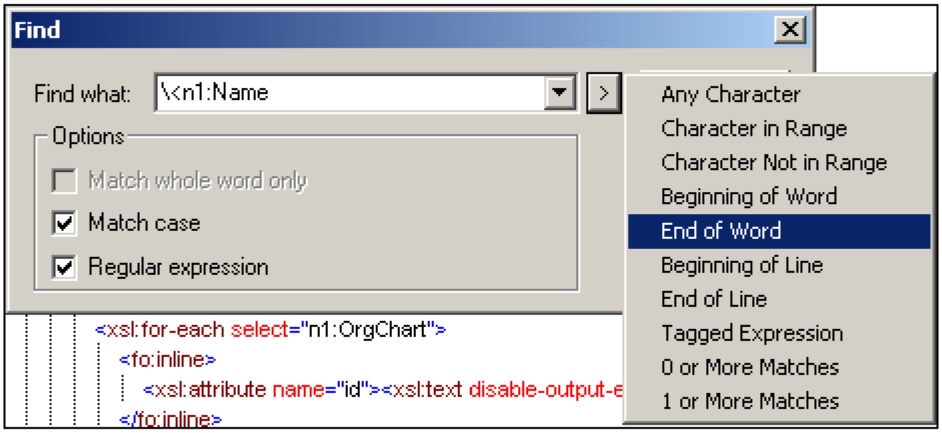
☐ Match whole word only

☑ Match case

☐ Regular expression

The following points should be noted:

- To enter a regular expression as the search term, check the Regular expression check box. You can create a regular expression with the help of a menu that pops out when you click the right-pointing arrowhead near the search term entry field.



Find what: \<n1:Name

Options

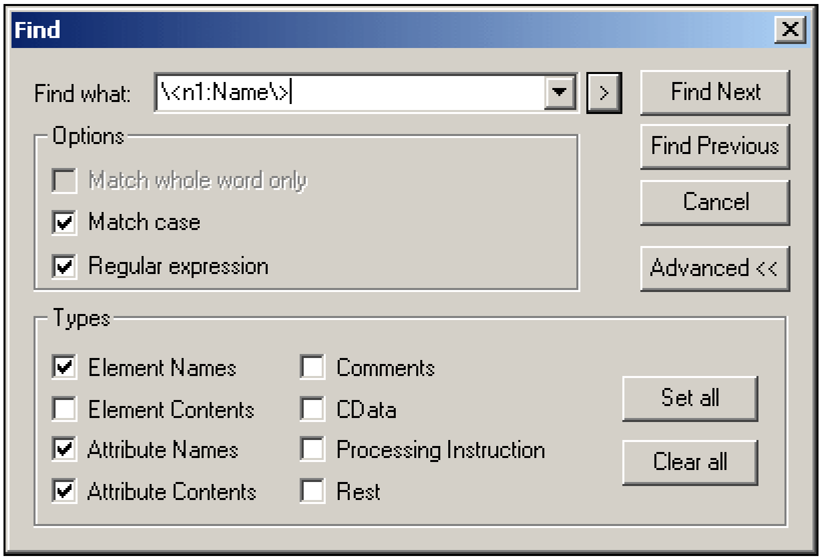
□ Match whole word only

☑ Match case

☑ Regular expression

```
<xsl:for-each select="n1:OrgChart">
  <fo:inline>
    <xsl:attribute name="id"><xsl:text disable-output-e
  </fo:inline>
```

- To set restrictions on what part of the document to search, click the Advanced button. This makes more search options available (*screenshot below*):



Find what: \<n1:Name\>

Options

□ Match whole word only

☑ Match case

☑ Regular expression

Types

☑ Element Names	□ Comments
□ Element Contents	□ CData
☑ Attribute Names	□ Processing Instruction
☑ Attribute Contents	□ Rest

Select the types of document content you wish to search by checking the appropriate check box.

Find Next command

The **Find Next (F3)** command repeats the last Find command to search for the next occurrence of the requested text. See Find for a description of how to use the search function.

Replace (Ctrl+H)

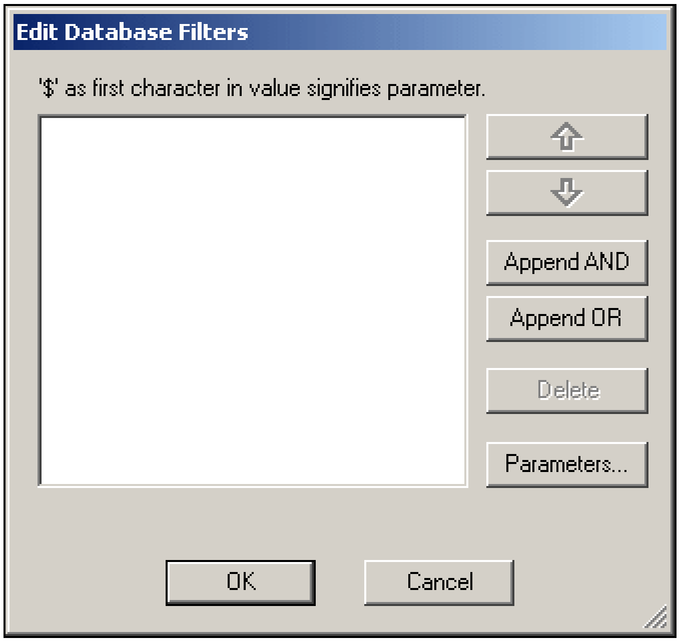
The **Replace** command is enabled in Authentic View (*not supported in Standard edition*) and enables you to search for a text string and replace it with another text string.

Edit DB Filter, Clear DB Filter

The **Edit DB Filter** command allows you to create and edit a filter for a database table (a DB Filter). A DB Filter determines what data from the selected database table is imported and displayed. A DB Filter consists of one or more criteria. When you specify criteria, you use an expression, which is a combination of operators (**=** or **>**) and values (text or numbers). Additionally, criteria can be joined by the logical operators **`AND`** or **`OR`**.

To create or edit a DB Filter, do the following:

1. Select the top-level data table element for which you wish to create or edit a DB Filter. Do this by clicking either the element tag in Design View or the element name in the schema tree.
2. Select **Edit | Edit DB Filter** or click the toolbar icon for the command. This pops up the Edit Database Filters dialog.



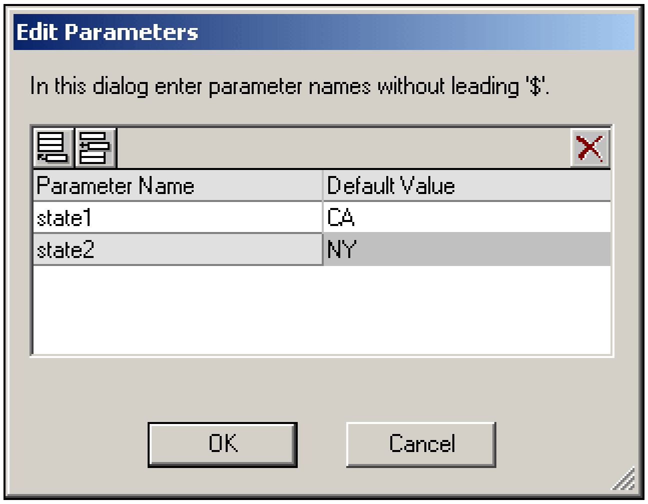
3. To add criteria use the **Append AND** and **Append OR** buttons. To move a criterion up or down, use the arrow buttons. To delete a criterion, use the Delete button.
4. Specify the criteria for the DB Filter. Each criterion consists of three parts: **`Field Name + Operator + Value`**. The options for Field Names and Operators are available in combo boxes. The value of the expression must be keyed in, and may be a parameter (indicated by a preceding `$` character).

Clear DB Filter command

The **Clear DB Filter** command deletes the filter after asking for and receiving a confirmation from you.

Stylesheet Parameters

The **Stylesheet Parameters** command enables you to declare and edit parameters and their default values. The command is available in both the Design Document view and the Authentic Editor View. When you click this command, the Edit Parameters dialog (*shown below*) pops up.



In this dialog enter parameter names without leading '$'.

Parameter Name	Default Value
state1	CA
state2	NY

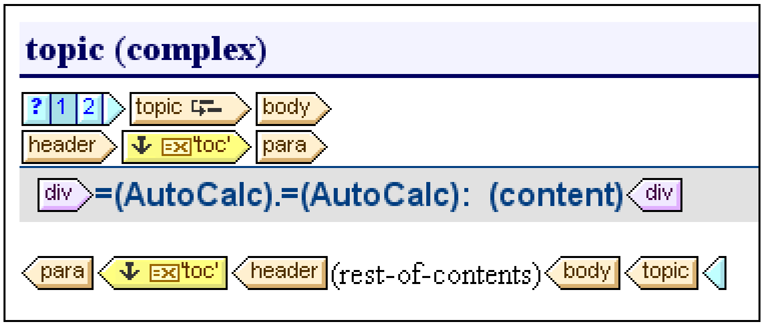
The following points should be noted:

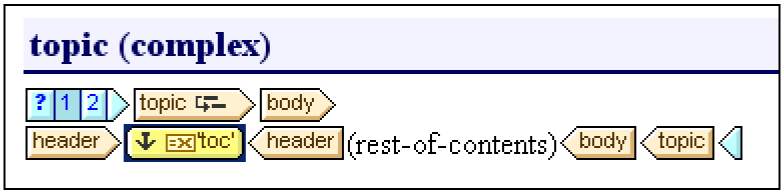
- You can insert, append, edit and delete parameters for the entire stylesheet and for the DB Filters.
- Parameter names must begin with a letter, and can contain the characters `A` to `Z`, `a` to `z`, `0` to `9`, and the underscore.

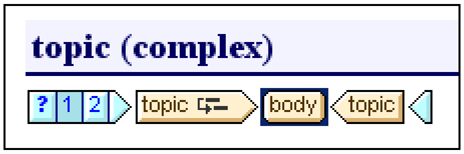
Collapse/Expand Markup

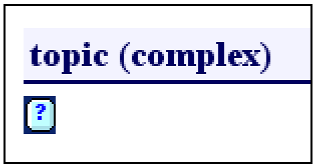
The **Collapse/Expand Markup** command is a toggle command, which collapses and expands the selected tag. It can be applied to any kind of tag: node, predefined format, SPS mechanism, etc. To collapse/expand a tag, double-click the tag; the end tag of an expanded tag may also be double-clicked to collapse that tag.

The screenshots below show how a series of tags are collapsed. Double-clicking a collapsed tag expands it.





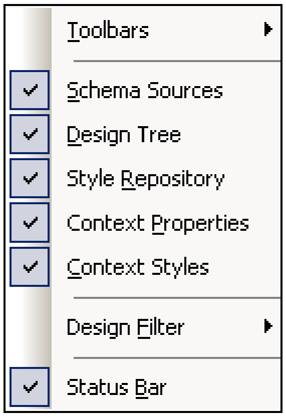




Collapsing a tag can be useful for optimizing the display according to your editing needs.

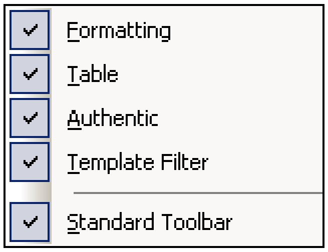
11.5 View Menu

The **View** menu (*screenshot below*) enables you to change the look of the GUI and to toggle on and off the display of GUI components. You can switch the display of individual toolbars, individual design entry helpers, design filters, and the status bar on and off.



Toolbars and Status Bar

Placing the cursor over the **Toolbars** item pops out a submenu (*screenshot below*), which enables you to turn on and off the display of the different toolbars.

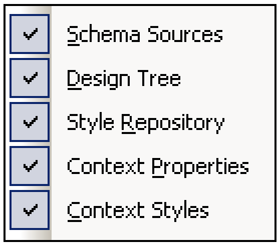


When a toolbar is checked, it is displayed. In the screenshot above all the toolbars are displayed. To toggle on or off the display of a toolbar, click the appropriate toolbar. For a complete description f toolbars, see the section Reference | Toolbars.

Status Bar
The display of the Status Bar, which is located at the bottom of the application window, can be switched on or off by clicking the **Status Bar** toggle command.

Design Entry Helpers

The **View** menu contains toggle commands to switch the display of each of the Entry Helpers in Design View on and off (*screenshot below*).



When an Entry Helper is checked it is displayed in Design View. Click an Entry Helper to set its display on or off, as required. This command is also used to make a hidden Entry Helper visible again.

Design Filter

The **Design Filter** menu item rolls out a sub-menu containing commands that enable you to filter the templates that are displayed in Design View. This is useful if your design is very long or contains several templates. Using the Design Filter mechanism, you can specify what kinds of template to display. The following filter options are available:

Icon	Command	Description
	Show only one template	Shows the selected template only. Place the cursor in a template and click to show that template only.
	Show all template types	Shows all templates in the SPS (main, global, named, and layout) .
	Show/Hide main template	Toggles the display of the main template on and off.
	Show/Hide global templates	Toggles the display of global templates on and off.
	Show/Hide Design Fragments	Toggles the display of Design Fragments on and off.
	Show/Hide layout templates	Toggles the display of layout templates on and off.

Note that these commands are also available as toolbar icons in the Design Filters toolbar.

11.6 Insert Menu

The **Insert** menu provides commands enabling you to insert a variety of design components into the SPS. Some of these commands are available as toolbar icons. Additionally, Insert menu commands are also available via context menus which appear when, in the SPS design, you right-click a selection or a cursor insertion point. In the context menus, commands that are not available at that location in the SPS are disabled.

Note: Since the Insert commands are used for constructing the SPS, they are available in Design View only.

Contents

The **Contents** command inserts a `(content)` placeholder at the cursor location point. There `(content)` placeholder can be inserted within two types of node, **element** and **attribute**, and it indicates that all children of the current node will be processed.

- If the current node is an element node, the node's children element nodes and text nodes will be processed. For the processing of children element nodes, global templates will be used if these exist. Otherwise the built-in template rule for elements will be used. For the processing of text nodes, the built-in template rule for text nodes will be used, the effect of which is to output the text. Effectively, the built-in template rule for elements, outputs the text of all descendant text nodes. It is important to note that the values of attributes will not be output when the `(content)` placeholder is used—unless a global template is defined for the attribute's parent element or one of its ancestors and the attribute is explicitly output, using either the `(content)` placeholder or any other content-rendering component.
- If the current node is an attribute node, the built-in template rule for the attribute's child text node will be used. This template copies the text of the text node to the output, effectively outputting the attribute's value.

The `(content)` placeholder can also be inserted for a node by placing the cursor inside the node tags, right-clicking, and selecting **Insert | Contents**.

Styling the contents

The `(content)` placeholder can be formatted by selecting it and using a predefined format and/or properties in Styles entry helper. This formatting is visible in the design, and, in the output, it will be applied to the contents of the node.

Replacing contents

If another node from the schema tree is dropped into a node containing a `(content)` placeholder, then the existing `(content)` placeholder is replaced by the new node.

Deleting contents

The `(content)` placeholder can be deleted by selecting it and pressing the **Delete** key on the keyboard.

Note: You can create an **empty template rule** by deleting the `(content)` placeholder of a node. An empty template rule is useful if you wish to define that some node have no template applied to it, i.e. produce no output.

Rest of Contents

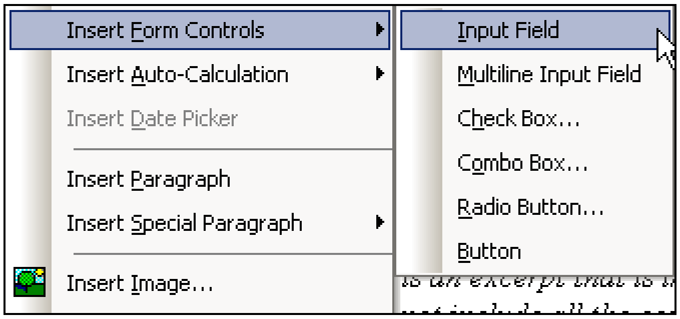
The **Rest of Contents** command inserts the `(rest-of-contents)` placeholder for that node. This placeholder represents the content of **unused child nodes** of the current node; it corresponds to the `xsl:apply-templates` rule of XSLT applied to the unused elements and text nodes of the current element. Note that templates are not applied for child attributes. the `(rest-of-contents)` placeholder can also be inserted for an element by placing the cursor inside the element tags, right-clicking, and selecting **Insert | Rest of Contents**.

Use the `(rest-of-contents)` placeholder in situations where you wish to process one child element in a specific way and apply templates to its siblings. It is important to apply templates to siblings in order to avoid the possibility that the siblings are not processed. This enables you to reach elements lower down in the document hierarchy.

The `(rest-of-contents)` placeholder can be deleted by selecting it and pressing the **Delete** key on the keyboard.

Form Controls

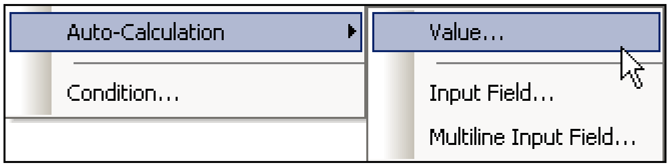
Mousing over the **Form Controls** command rolls out a submenu (*screenshot below*) containing commands to insert various form controls (data-entry devices).



How to create each of these form controls is described in the section Using Data-Entry Devices. After a form control has been created, its properties can be edited by selecting it and then editing the required property in the Properties entry helper.

Auto-Calculation

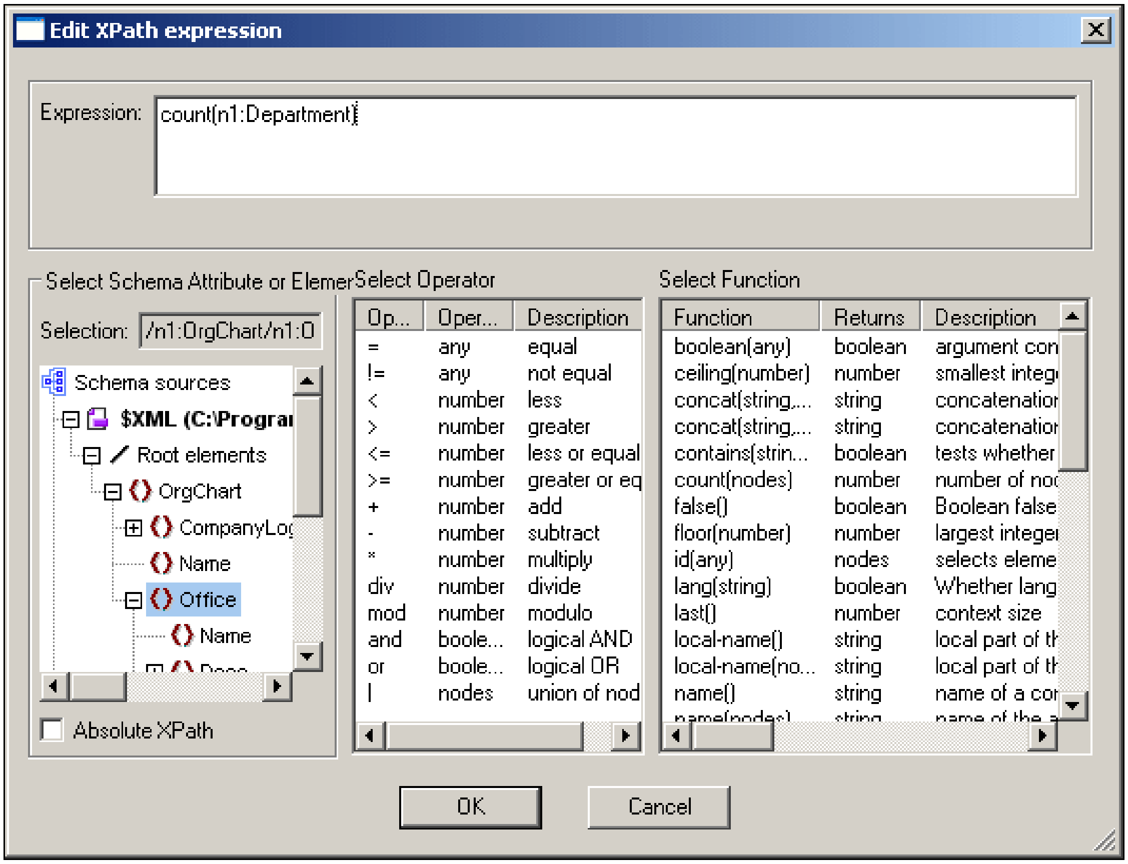
An **Auto-Calculation** uses an XPath expression to calculate a value. This value is displayed at the point where the Auto-Calculation is inserted. An Auto-Calculation can be inserted in the SPS as a text value, input field, or multiline input field. When the cursor is placed over **Insert | Auto-Calculation**, a menu pops out (*screenshot below*), enabling you to choose how the Auto-Calculation should be inserted.



The value of the Auto-Calculation will be displayed accordingly in Authentic View and the output document.

The XPath expression for the Auto-Calculation

On selecting how the Auto-Calculation should be represented, the Edit XPath Expression dialog (*screenshot below*) pops up.



Expression: count(n1:Department)

Selection: /n1:OrgChart/n1:O

Op...	Oper...	Description
=	any	equal
!=	any	not equal
<	number	less
>	number	greater
<=	number	less or equal
>=	number	greater or eq
+	number	add
-	number	subtract
*	number	multiply
div	number	divide
mod	number	modulo
and	boole...	logical AND
or	boole...	logical OR
|	nodes	union of nod

Function	Returns	Description
boolean(any)	boolean	argument con
ceiling(number)	number	smallest integ
concat(string,...	string	concatenatior
concat(string,...	string	concatenatior
contains(strin...	boolean	tests whether
count(nodes)	number	number of noc
false()	boolean	Boolean false
floor(number)	number	largest integer
id(any)	nodes	selects eleme
lang(string)	boolean	Whether lang
last()	number	context size
local-name()	string	local part of th
local-name(no...	string	local part of th
name()	string	name of a cor
name(nodes)	string	name of the a

The context node for the expression being built is highlighted in the schema tree in the pane at extreme left. You can enter the XPath expression directly in the text box, or you can double click an item (in any of the three panes) to insert that item. Nodes inserted from the schema tree in the left-hand pane are inserted relative to the context node (if the Absolute XPath check box is unchecked) or as an absolute expression starting from the document node (if the Absolute XPath check box is checked).

After completing the XPath expression, click **OK** to finish inserting the Auto-Calculation.

Updating an XML node with an Auto-Calculation
A node in an XML document can be updated with the result of an Auto-Calculation. How to do this is described in the section, Updating Nodes with Auto-Calculations.

Date Picker

The **Date Picker** command inserts a Date Picker at the current cursor position. It will be enabled only when the cursor is within an `xs: date` or `xs: dateTime` node and if the element has been created as `( contents)` or an input field.

Paragraph, Special Paragraph

The **Paragraph** command ¶ inserts an HTML paragraph `<p>` element around the selected component. A component is considered selected for this purpose when the entire node is selected (by clicking either of its tags) or when static text is selected. If the cursor is placed within static text, the paragraph element is inserted (start and end tags) at this point.

The **Special Paragraph** command allows you to assign a predefined format to the selected node. The available predefined formats can also be selected from the combo box in the toolbar.

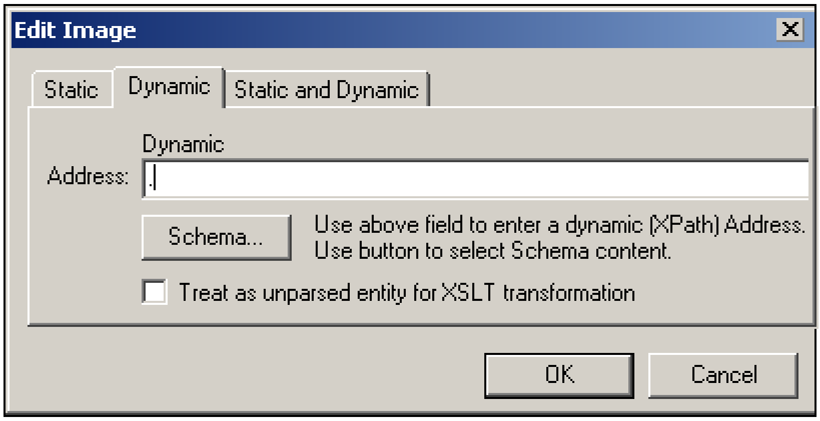
Each paragraph type has particular formatting features that can be used to advantage. Note that the `pre` format type enables carriage returns to be output as such instead of them being normalized to whitespace.

Image

The **Image** command allows you to insert an image using an image location address that either comes from the XML document (dynamic) or is entered by you directly in the SPS (static).

To insert an image, do the following:

1. Click **Insert | Image** or the Insert Image toolbar icon. The Insert Image dialog (*shown below*) appears.



Address: .

Use above field to enter a dynamic (XPath) Address.
Use button to select Schema content.

□ Treat as unparsed entity for XSLT transformation

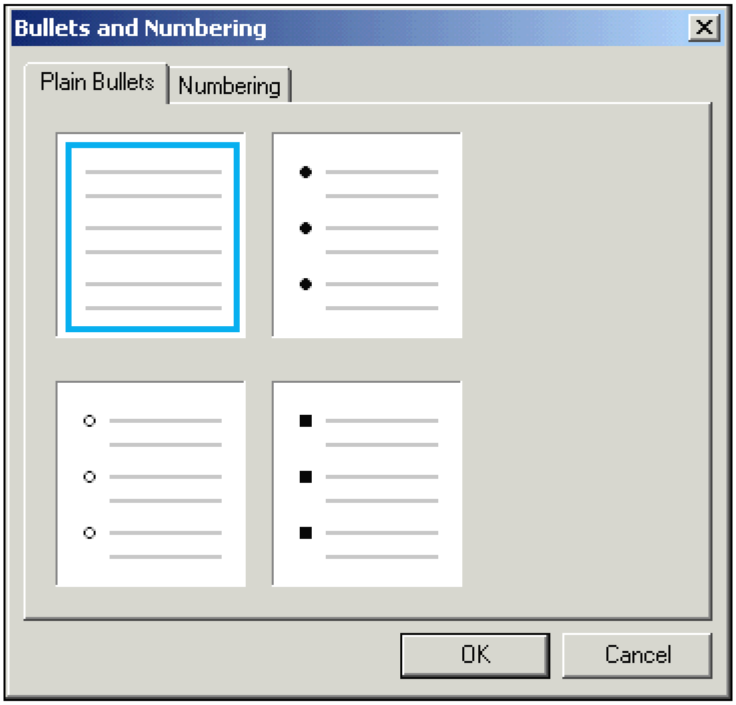
2. Select the required tab (Static, Dynamic, or Static and Dynamic), and enter the address of the image location and/or the XPath expression that locates the image address in the XML document. The screenshot above shows how a Static and Dynamic address is entered.

Using unparsed entities
If the SPS is DTD-based and uses unparsed entities, then, for the dynamic part of an image address, the URI declared as the value of the unparsed entity can be used. For details of how to use unparsed entities, see Unparsed Entity URIs.

Bullets and Numbering

The **Bullets and Numbering** command allows you to create a list, either static or dynamic. The list items of a static list are entered in the SPS, while those of dynamic lists are taken from sibling nodes in the XML document.

Clicking the Bullets and Numbering command pops up the Bullets and Numbering dialog (*screenshot below*), in which you select the required list item marker (for example, bullet, circle, or square). Clicking OK creates the list with that type of list item marker.

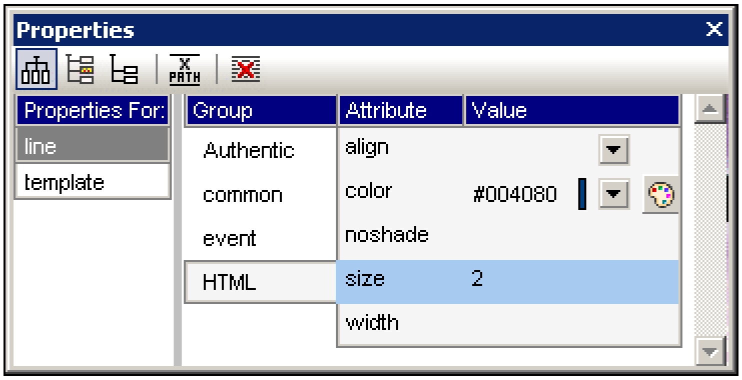


To create a static list, place the cursor at the location in the design where the list is required and click the Bullets and Numbering command. Select the list item marker you want and click **OK**. An empty list item is created at that point with the selected marker. Type in the required text for that list item. Press Enter to create a new list item in that list.

To create a dynamic list, select Bullet and Numbering from the context menu that pops up when you drop a node into the design.

Horizontal Line

The **Horizontal Line** command inserts a horizontal line at the cursor insertion point. This command is not available when an SPS component is selected. To set properties for the horizontal line, select the line in the design, and in the Properties entry helper, select *line*, and specify values for properties in the *line* group (*see screenshot below*).



Properties For:	Group	Attribute	Value
line	Authentic	align	
template	common	color	#004080
	event	noshade	
	HTML	size	2
		width	

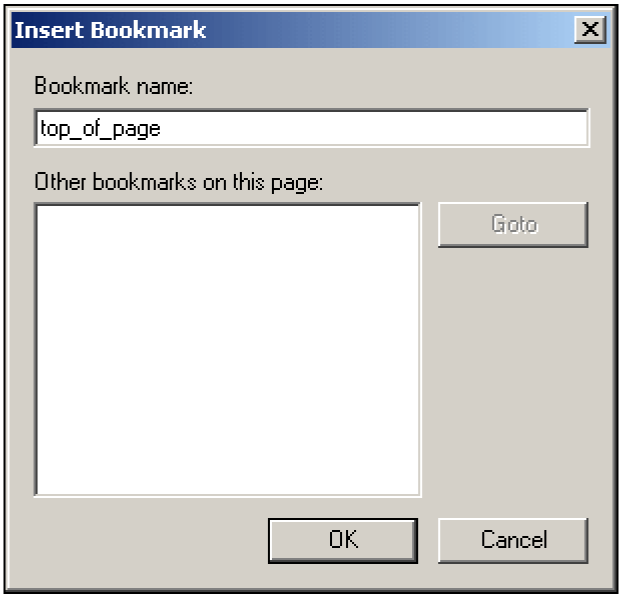
You can specify the following properties for the line: its `color`, `size` (thickness), `width` (in the design), `alignment`, and the `noshade` property.

Bookmark

The **Bookmark** command allows you to insert a bookmark (or anchor) anywhere in the SPS. A bookmark can be referenced by a Hyperlink.

To insert a bookmark, do the following:

1. Place the cursor at the location where you wish to create the bookmark.
2. Select **Insert | Bookmark**, or right-click and select **Insert | Bookmark**.



Bookmark name: top_of_page

Other bookmarks on this page:

3. In the Insert Bookmark dialog, enter a name for the bookmark in the Name field. (If there are more bookmarks on the page, they are displayed in the Other Bookmarks on This Page pane.)
4. Click **OK**. The bookmark is defined.

Note: A bookmark is created for a single, fixed location in the SPS. The name of the bookmark is a static name that you give. It does not come dynamically from the XML document. You cannot assign a dynamic name to a bookmark.

You can edit the name of a bookmark after it has been created. Do this by selecting the bookmark, clicking the Edit button of the bookmark (Properties entry helper, link, *Link* group of properties, bookmark name), and editing the name of the bookmark in the text box of the Edit Bookmark dialog.

Hyperlink

The **Hyperlink** command enables you to insert a link from any part of the output document (HTML) to an anchor within the output document or to an external document or document fragment. Note that links are created only in the output document; linking is not available in Authentic View.

To insert a hyperlink, do the following:

1. Select the SPS component or text fragment to be made into a hyperlink.
2. Click the Hyperlink icon in the toolbar, or select **Insert | Hyperlink**, or right-click and select **Insert | Hyperlink**.
3. In the Insert Hyperlink dialog that appears, specify the document or document fragment you wish to link to. You do this by specifying a URI in one of the following forms:
   - a static address (entered directly; you can select an HTML file via the Browse button, and a fragment in the current document via the Bookmark button). Examples would be: `http://www.altova.com` (static Web page URI); `U:\documentation\index.html` (via Browse button); or `#top_of_page` (via Bookmark button).
   - a dynamic address (which comes from a node in the XML document; you specify the node). An example would be a node such as `//otherdocs/doc1`.
   - a combination of static and dynamic text for an address (you specify the static text and the XML document node). An example would be `www.altova.com -- department/name -- #intropara`.
4. Click OK. The hyperlink is created.

Note: When specifying the node for a dynamic hyperlink entry, you can enter the XPath expression as an absolute XPath expression by checking the Absolute Path check box. If this check box is not checked, the XPath expression for the node you select via the Schema button is entered as being relative to the currently selected component.

Using unparsed entities

For the dynamic part of a hyperlink address, you can use the URI declared for an unparsed entity in the DTD—if you are using a DTD. For details of how to use unparsed entities, see Using unparsed entity URIs.

Removing a hyperlink

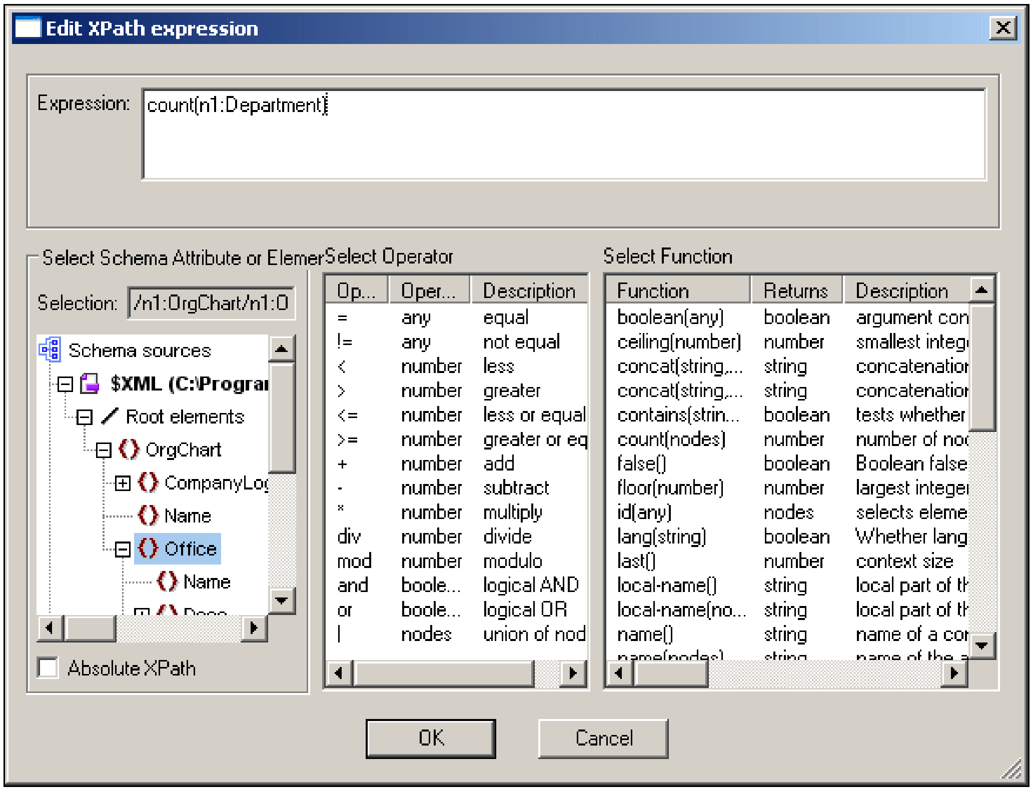
The link for the selected component can be removed by clicking the Remove Link button in the Insert Hyperlink dialog.

Condition, Output-Based Condition

The **Condition** command enables you to insert a condition at the cursor point or around the selection. A condition consists of one or more branches, with each branch containing a specific set of processing rules. In this way, different sets of processing rules can be specified for different branches. For example, if the content of a node is the string `Stop`, the branch can test this, and specify that the contents of the node be colored red; a second branch can test whether the contents of the node is the string `Go`, and, if yes, color the contents of the node green; a third branch can specify that if the contents of the node is neither the string `Stop` nor the string `Go`, the contents of the node should be colored black.

To insert a condition, do the following:

1. Place the cursor at the desired location in the design or select the component around which the condition is to be inserted.
2. Select the menu command **Insert | Condition** or right-click and select the context menu command **Insert | Condition**.
3. In the Edit XPath Expression dialog that pops up (*screenshot below*), enter the XPath expression.



Expression: count(n1:Department)

Selection: /n1:OrgChart/n1:O

☐ Absolute XPath

Op...	Oper...	Description
=	any	equal
!=	any	not equal
<	number	less
>	number	greater
<=	number	less or equal
>=	number	greater or eq
+	number	add
-	number	subtract
*	number	multiply
div	number	divide
mod	number	modulo
and	boole...	logical AND
or	boole...	logical OR
|	nodes	union of nod

Function	Returns	Description
boolean(any)	boolean	argument con
ceiling(number)	number	smallest integ
concat(string,...	string	concatenatior
concat(string,...	string	concatenatior
contains(strin...	boolean	tests whether
count(nodes)	number	number of noc
false()	boolean	Boolean false
floor(number)	number	largest integer
id(any)	nodes	selects eleme
lang(string)	boolean	Whether lang
last()	number	context size
local-name()	string	local part of th
local-name(no...	string	local part of th
name()	string	name of a cor

The context node for the expression being built is highlighted in the schema tree in the pane at extreme left. You can enter the XPath expression directly in the text box, or you can double click an item (in any of the three panes) to insert that item. Nodes inserted from the schema tree in the left-hand pane are inserted relative to the context node (if the Absolute XPath check box is unchecked) or as an absolute expression starting from the document node (if the Absolute XPath check box is checked).

4. Click **OK** to finish inserting the condition. The condition is created with one branch, the test for which is the XPath expression you entered.

Insert Output-Based Condition (Enterprise and Professional editions)

This command inserts an output based-condition at the cursor location or around the selected component. Each branch of the condition represents a single output (Authentic View or HTML). To determine which branch represents which output, mouseover the branch tag or check the XPath expression of the selected branch (in the Properties entry helper, in the Condition Branch entry, click the Edit button). If the output-based condition was created at a cursor insertion point, all branches will be empty and content will have to be inserted for each branch. If the output-based condition was created around a component, each branch will contain that component. For more details about output-based conditions, see Output-Based Conditions. You can edit, move, and delete output-based conditions in the same way you would with a standard condition.

Editing the XPath expressions of branches
To edit the XPath expression of a branch, select the branch in Design View. Then, in the Properties entry helper, select `condition branch | when`. Click the **Edit** button ... for the XPath item. This pops up the Edit XPath Expression dialog (*screenshot above*), in which you can edit the expression. Click `OK` when done.

Adding branches, changing the order of branches, and deleting branches
To add new branches, change the order of branches, and delete branches, right-click the required branch and select the relevant item from the context menu.

Table of Contents

Mousing over the **Table of Contents** command rolls out a submenu containing commands to insert various commands relating to the creation of a Table of Contents (TOC) template, TOC bookmarks, and a design document structure for the TOC.

The list of commands is as follows. For the details of how to use them click on the respective links, which will take you to the section on how to use the TOC.

- Insert Table of Contents
- TOC Bookmark
- TOC Bookmark (Wizard)
- TOC Reference
- TOC Reference | Entry Text / Leader / Page Reference
- Hierarchical Numbering
- Sequential Numbering
- Level
- Level Reference
- Template Serves as Level

Note: These commands are also available as commands in a context menu, depending on where you right click in the design.

Design Fragment

Mousing over the Design Fragment command rolls out a submenu containing all the Design Fragments currently in the design. Clicking a Design Fragment in the submenu inserts it at the cursor insertion point.

Page

With the **Page** command you can insert, for paged media output, a page break. Such insertions are possible only at cursor insertion points.

Page Break

Click **Page | Break** to define a page break at the cursor insertion point. The page break is displayed as a dashed line across the whole of the Design window. In HTML output, while the page break has no effect in the browser view, a page break will be inserted when the browser view of the HTML file is printed out.

Deleting Page Breaks

To delete page breaks, select the placeholder and click **Delete**.

DB Control

Mousing over the **DB Controls** command rolls out a submenu containing commands to insert controls in Authentic View that enable the Authentic View user to navigate the display of records in Authentic View and to query the DB. These control can be inserted in the design and will appear in the Authentic View document at the corresponding locations.

The list of commands is as follows. For the details of how to use them click on the respective links.

- Navigation
- Navigation + Goto
- Query Button

For details about how these controls are created and what they do, see the section SPS Design Features for DB.

11.7 Table Menu

The **Table** menu provides commands enabling you to insert a static table and to change the structure and properties of both static and dynamic tables. You can edit table structure by appending, inserting, deleting, joining, and splitting rows and columns. Properties of the table as well as of individual columns, rows, and cells are defined using CSS styles and HTML properties for tables and its sub-components.

The Table commands are available in the Table menu (*see list below*) and as icons in the Table toolbar. The availability of various table commands depends on the current cursor position. A static table can be inserted at any location in the SPS by clicking the Insert Table command. A dynamic table is inserted by creating an element as, or changing an element to, a table. To edit the table structure, place the cursor in the appropriate cell, column, or row, and select the required editing command. To edit a formatting property, place the cursor in the appropriate cell, column, row, or table, and, in the Styles entry helper and/or Properties entry helper, define the required property for that table component.

The following commands are available in the Table menu:

- Insert Table, Delete Table
- Append/Insert Row/Column
- Delete Row, Column
- Join Cell Left, Right, Below, Above
- Split Cell Horizontally, Vertically
- View Cell Bounds
- Vertical Alignment of Cell Content

Headers and footers
When you create a dynamic table, you can specify whether you wish to include headers and/or footers. (Footers are allowed only when the table grows top–down.) You can create a header and footer in a static table by manually inserting a top and bottom row, respectively. The structures of headers and footers in both static and dynamic tables can be modified by splitting and joining cells.

Navigating in tables
Use the Tab and arrow keys to navigate the table cells.

Adding cell content
Any type of SPS component can be inserted as the content of a cell. The component should be formatted using the standard formatting tools.

Insert Table, Delete Table

The **Insert Table** command inserts an empty static table into the design tab. Selecting this command opens a dialog box which allows you to define the size of the table (in terms of its rows and columns). You can change this structure subsequently by appending, inserting, and deleting rows and/or columns.

The **Delete Table** command deletes the static or dynamic table in which the cursor is.

To insert a dynamic table**,** see Creating dynamic tables.

Append/Insert Row/Column

The **Append Row** command appends a row to the static or dynamic table in which the cursor is.

The **Insert Row** command inserts a row above the row in which the cursor is. This command applies to both static and dynamic tables.

The **Append Column** command appends a column to the static or dynamic table in which the cursor is.

The **Insert Column** command inserts a column to the left of the column in which the cursor is. This command applies to both static and dynamic tables.

Delete Row, Column

The **Delete Row** command deletes the row in which the cursor is. This command applies to both static and dynamic tables.

The **Delete Column** command deletes the column in which the cursor is. This command applies to both static and dynamic tables.

Join Cell Left, Right, Below, Above

The **Join Cell Left** command joins the cell in which the cursor is to the adjacent cell on the left. The contents of both cells are concatenated in the new cell. All property values of the cell to the left are passed to the new cell. This command applies to both static and dynamic tables.

The **Join Cell Right** command joins the cell in which the cursor is to the cell on the right. The contents of both cells are concatenated in the new cell. All property values of the cell to the left are passed to the new cell. This command applies to both static and dynamic tables.

The **Join Cell Below** command joins the cell in which the cursor is to the cell below. The contents of both cells are concatenated in the new cell. All property values of the cell on the top are passed to the new cell. This command applies to both static and dynamic tables.

The **Join Cell Above** command joins the cell in which the cursor is to the cell above. The contents of both cells are concatenated in the new cell. All property values of the cell on top are passed to the new cell. This command applies to both static and dynamic tables.

Split Cell Horizontally, Vertically

The **Split Cell Horizontally** command creates a new cell to the right of the cell in which the cursor is. The contents of the original cell stay in the original cell. All properties of the original cell are passed to the new cell. This command applies to both static and dynamic tables.

The **Split Cell Vertically** command creates a new cell below the cell in which the cursor is. The contents of the original cell remain in the upper cell. All properties of the original cell are passed to the new cell. This command applies to both static and dynamic tables.

View Cell Bounds

The **View Cell Bounds** command toggles the display of table boundaries (borders) on and off for tables that have a table border value of 0. This command applies to both static and dynamic tables.

Vertical Alignment of Cell Content

Commands to set the vertical alignment of cell content are available as icons in the toolbar. Place the cursor anywhere in the cell, and click the required icon.

Vertically Align Top vertically aligns cell content with the top of the cell.

Vertically Align Middle vertically aligns cell content with the middle of the cell.

Vertically Align Bottom vertically aligns cell content with the bottom of the cell.

11.8 Authentic Menu

The **Authentic** menu contains commands that enable you to:

- Customize aspects of the Authentic View of an XML document that will be displayed using the SPS.
- Edit documents in the Authentic View preview of StyleVision.

The commands in the Authentic menu are listed below:

- Text State Icons
- CALS/HTML Tables
- Auto-Add Date Picker
- Auto-Add DB Controls
- Table, Markup
- Define DTD Entities
- Validate XML

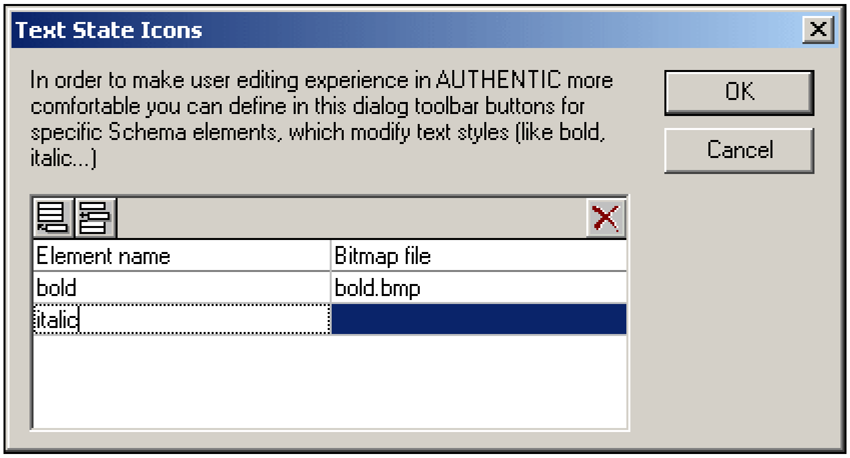
Each of these commands is described in detail in the sub-sections of this section.

Text State Icons

The **Text State Icons** command enables you to define an icon for a global element. Once defined the icon can be displayed in the toolbar of Authentic View (i.e. in Altova's XMLSpy, Authentic Desktop, and Authentic View products), thus allowing the Authentic View user to insert an element around selected text by clicking the icon. This feature is intended for elements that provide inline formatting, such as bold and italic fonts.

All global elements in the schema appear in the All Global Elements list in the Schema Sources entry helper. To create a Text State icon, do the following:

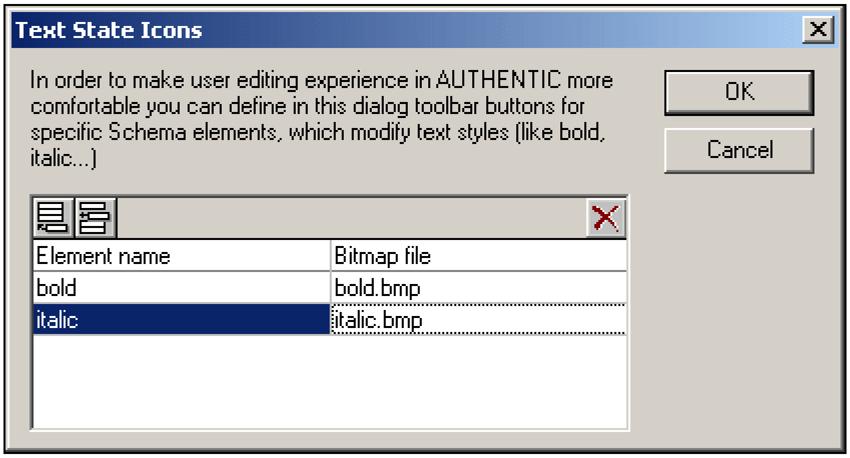
1. Define a global template for the global element that is to get a Text State icon. The formatting that you define for the global template will be applied to text when the Authentic View user clicks the Text State icon. (If the element you require is not available as a global element, you must make it a global element if you wish to use this feature.)
2. Select the menu option **Authentic | Text State Icons**. This opens the Text State Icons dialog box.
3. Click the Append button to add a new Text State icon.
4. In the **Element name** field, enter the name of the element for which you wish to create the Text State icon, and click OK.



In order to make user editing experience in AUTHENTIC more comfortable you can define in this dialog toolbar buttons for specific Schema elements, which modify text styles (like bold, italic...)

Element name	Bitmap file
bold	bold.bmp
italic	

5. Press **Tab** or double click in the **Bitmap file** column, and enter the name of the icon image that you wish to have displayed in the Authentic View toolbar.



In order to make user editing experience in AUTHENTIC more comfortable you can define in this dialog toolbar buttons for specific Schema elements, which modify text styles (like bold, italic...)

Element name	Bitmap file
bold	bold.bmp
italic	italic.bmp

6. Click **OK** to confirm.

7. Place the icon image file (`.bmp` file) in the `\\sps\Picts` folder of your application folder.

When you edit an XML document in Authentic View (in Altova's XMLSpy, Authentic Desktop, and Authentic View products) using this StyleVision Power Stylesheet, the icon image appears in the toolbar as a Text State icon. If the Authentic View user selects text and clicks the icon, the element represented by that icon is created around the selected text and the formatting you defined in the global template for that element is applied to the selected text.

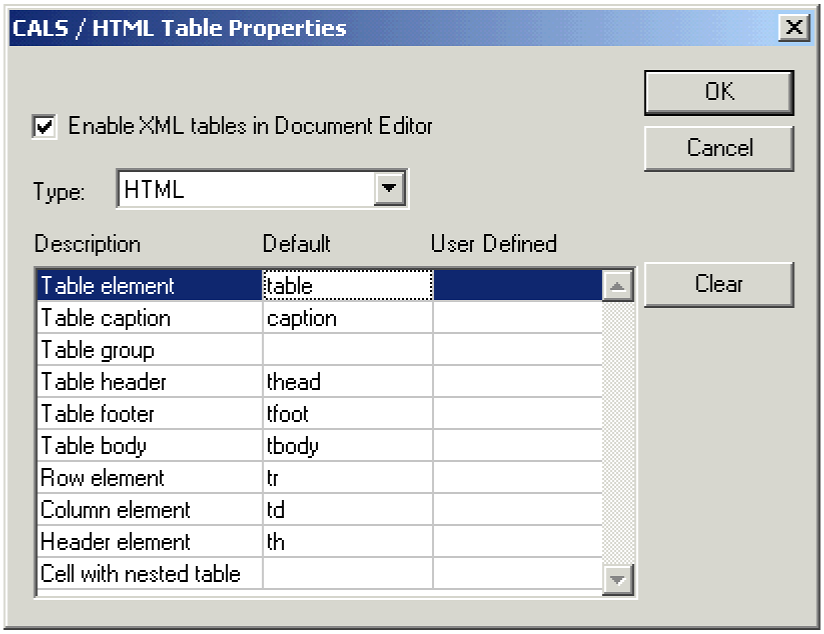
Note: Text State icons are not available in Authentic Preview of StyleVision.

CALS/HTML Tables

The **CALS / HTML Tables...** command pops up a dialog in which you specify:

- whether the Authentic View user may insert XML tables (following either the CALS or the HTML table model) in the XML document or not, and
- the XML elements that will constitute the XML table structure if XML tables are allowed.

When you click **Authentic | CALS / HTML Tables...**, the following dialog appears:



CALS / HTML Table Properties

☑ Enable XML tables in Document Editor

Type: HTML

Description	Default	User Defined
Table element	table	
Table caption	caption	
Table group		
Table header	thead	
Table footer	tfoot	
Table body	tbody	
Row element	tr	
Column element	td	
Header element	th	
Cell with nested table		

Enabling XML Tables in Authentic View
To enable XML tables in Authentic View, check the box marked Enable XML tables in Document Editor.

Enabling XML tables allows the Authentic View user to insert tables that can be structured and formatted as required by the user. This is as opposed to static and dynamic SPS tables, which are structured and formatted by you, the designer of the StyleVision Power Stylesheet. When an XML table is inserted in Authentic View, a set of XML elements conforming to either the CALS or HTML table model is inserted in the XML document. Cell content in Authentic View is entered as content of the corresponding element in the XML document. When the table is formatted in Authentic View, the appropriate attributes and their values are added to the relevant elements in the XML document.

Defining schema elements for the selected table model
In the combo box, select the table model (CALS or HTML) with which you want your schema table structure to correspond. This combo box is enabled only if the Enable XML tables check box has been checked. Note the following points:

- The **table elements in the schema** must correspond exactly with the structure of the selected table model (either CALS or HTML) to ensure the correct functioning of this feature; however, they do not need to have the same names as those of the table model elements.
- If a table element in the schema is named differently from the corresponding element in the selected table model, then the schema element must be mapped to the corresponding table model element by entering its name in the User Defined column of the CALS / HTML Table Properties dialog.

- The names of all **attributes** of schema table elements must correspond exactly with the relevant attribute names in the table model. These attributes must therefore be defined in the schema. Attribute names must correspond exactly because when the Authentic View user modifies the table structure or enters (for HTML tables) table properties, Authentic View enters attributes and enters/modifies attribute values for the appropriate element in the XML file. If these attributes are not present in the schema or are named (or spelled) differently in the schema, then the XML file will be invalid.

After selecting the appropriate model, all elements in the selected model which do not exist in the table structure of the schema, or which do not have a corresponding schema element specified for it, are displayed in red. If a schema element exists that corresponds to a CALS/HTML element, then you must map this element to the corresponding CALS/HTML element.

Note: When all the elements in the CALS / HTML Table Properties dialog are displayed in black, this means that the schema contains elements that correspond to the elements of the selected table model. If an element name in the Default column is displayed in red, this indicates that either no corresponding schema element exists or that a corresponding schema element exists but has not been mapped to the table model element; in the latter case you must enter the name of the corresponding schema element into the User Defined column.

Caution: If all the element names are displayed in black, be aware that this does not necessarily mean that the table structure of the schema corresponds exactly with the selected table model. There may still be additional elements in the table structure of the schema that could cause errors. You should carefully check your schema for this. Also check the attributes of the schema elements for exact correspondence and naming. Absent or wrongly named attributes will cause validation errors if the Authentic View user uses table formatting properties that use these attributes.

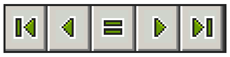
Auto-Add Date Picker

This is a toggle command that switches the Auto-Add Date Picker feature on and off. When the Auto-Add Date Picker feature is ON, any `xs:date` or `xs:dateTime` datatype element that is created as contents or as an input field will have the Date Picker automatically inserted within the element tags and after the `contents` placeholder or input field.

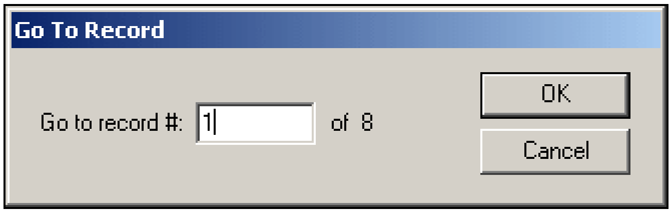
Auto-Add DB Controls

This is a toggle command that switches the Auto-Add DB Controls feature on and off.

When the Auto-Add DB Controls is on, then, whenever a DB table element is dropped into the design, the DB Controls panel (*shown below*) is inserted immediately before the `Row` child element of that DB table element.



The DB Controls panel enables the Authentic View user to navigate the rows of the DB table in Authentic View. The first (leftmost) button navigates to the first record; the second button navigates to the previous record; the third button is the Goto button; it pops up a dialog (*screenshot below*) that prompts you for the number of the record to which you wish to go; the fourth button navigates to the next record; and the fifth button navigates to the last record.



Go To Record

Go to record #: 1 of 8

When the Auto-Add DB Controls toggle is turned off, the DB Controls panel is **not** inserted when a DB table is dropped into the Design document.

Note: You can manually insert navigation buttons by placing the cursor anywhere between the start and end tags of the DB table and selecting the required option from the **Insert | DB Controls** submenu. Note that in this submenu the DB Controls panel can be inserted as the four navigation buttons or as the four navigation buttons plus the button that calls the Goto Record dialog.

Table, Markup

The **Table** command is enabled in Authentic View preview and allows you to manipulate the rows of a **dynamic table**. When you place the cursor over the command, a submenu pops out which contains more specific commands. You can append, insert, duplicate, and delete rows, and you can move the selected row up and down relative to the other rows of the table. A row is selected by placing the cursor inside it. Note that each row of a dynamic table represents an occurrence of a repeatable element within the set of all those repeatable elements.

With the **Markup** command, you can select between the Hide Markup and the various Show Markup modes. When you place the cursor over the command, a submenu which contains the specific commands. In Hide Markup mode, node tags are not displayed. In Show Full Markup mode, opening and closing tags, with their node names, are displayed.

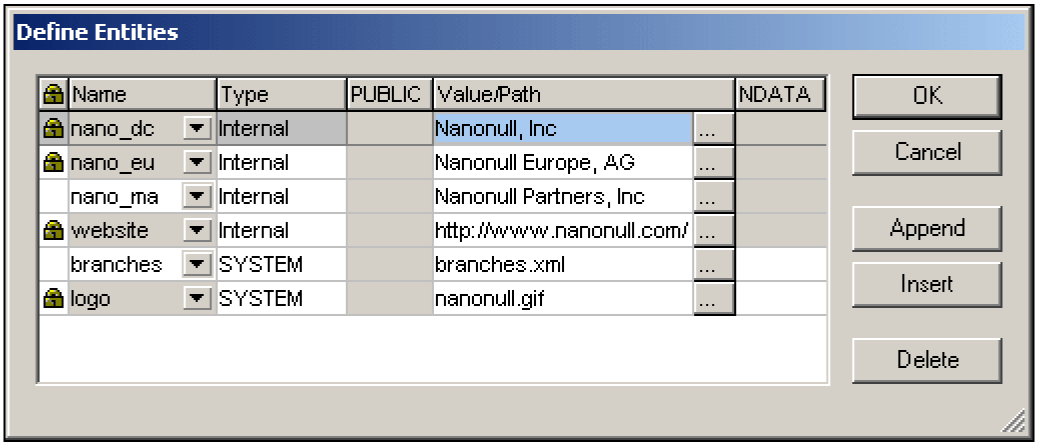
Define DTD Entities

The **Define Entities** command is available only in Authentic View. With the **Define Entities** command in Authentic View, you can define entities that you want to add to your **XML document**. After an entity has been defined, it can be inserted in the XML document by right-clicking at the location where you wish to insert the entity, and, from the context menu that pops up, selecting **Insert Entity**, and then the name of the entity to be inserted.

An entity that you define with this command can be any of three types:

- Internal parsed entity. The value of the entity is a text string that usually occurs frequently in the document. Using an entity ensures that all occurrences are expanded to the value defined here.
- External parsed entity. This is an external XML file that will replace each occurrence of the entity. The value of the entity is the URI of the external XML file.
- External unparsed entity. This is an external resource that will be called when the entity is processed. The value of the entity is the URI of the external resource.

Clicking the command, pops up the Define Entities dialog (*screenshot below*).



🔒	Name	Type	PUBLIC	Value/Path	NDATA
🔒	nano_dc	Internal		Nanonull, Inc	
🔒	nano_eu	Internal		Nanonull Europe, AG	
	nano_ma	Internal		Nanonull Partners, Inc	
🔒	website	Internal		http://www.nanonull.com/	
	branches	SYSTEM		branches.xml	
🔒	logo	SYSTEM		nanonull.gif	

For a description of how to use this dialog, see Define Entities in the Authentic View documentation.

Validate XML

The **Validate XML (F8)** command checks the validity of the XML file against the associated schema. Any additional validation requirement that you have entered for individual nodes (**Authentic | Node Settings**) is also checked. The result of the validation check is displayed in a pop-up message box.

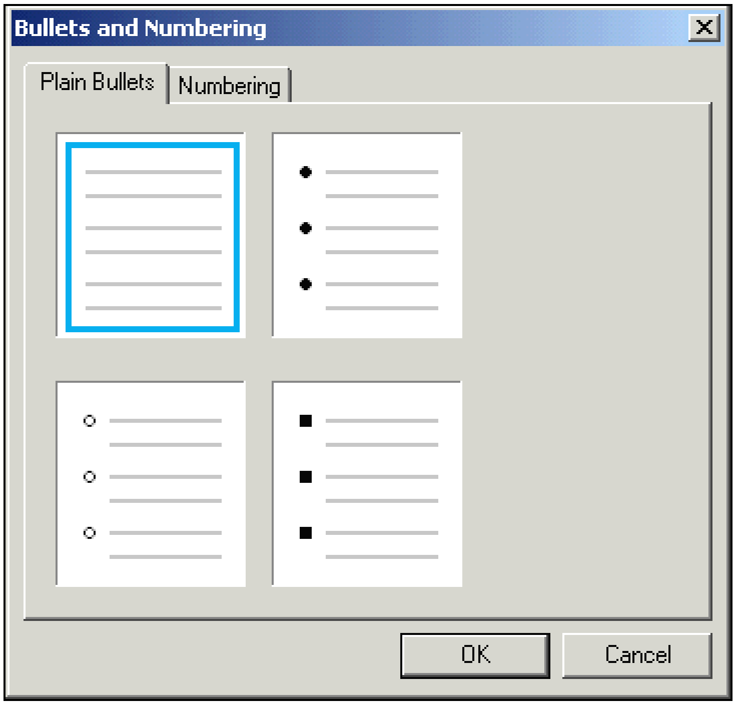
11.9 Properties Menu

The **Properties** menu contains commands that enable you to insert lists and define datatype formats for the input formatting feature. The description of the commands is organized into the following sub-sections:

- Bullets and Numbering command, to insert lists.
- Predefined Format Strings command, to define numeric datatype formats for a given SPS.

Bullets and Numbering

The **Bullets and Numbering** command enables you to insert a list at the cursor location. Clicking the command pops up the Bullets and Numbering dialog (*screenshot below*), in which you can select the list style; in the case of a numbered list, the initial number can also be specified.



Predefined Input Formatting Strings

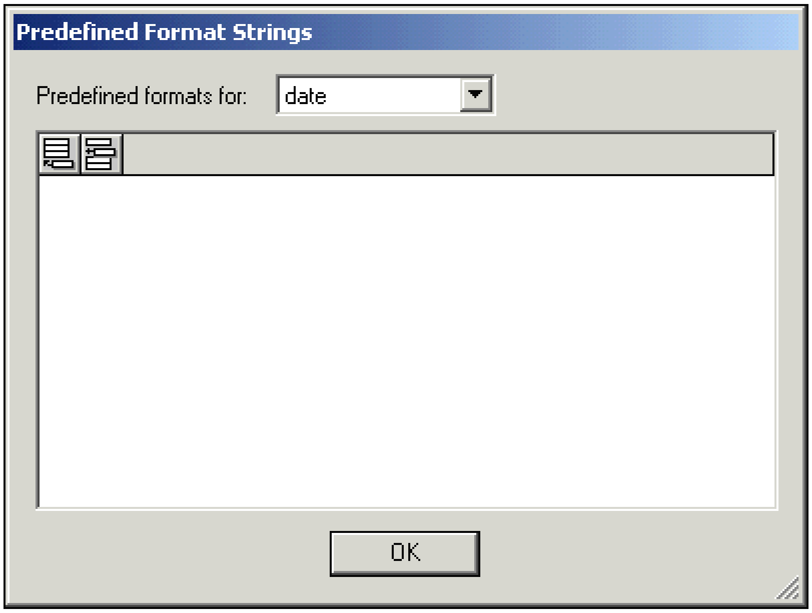
Any `(content)` placeholder, input field, or Auto-Calculation which is of a `numeric`, `date`, `time`, `dateTime` or `duration` datatype can be assigned a custom format with the Input Formatting dialog. In the Input Formatting dialog, you can either create a format directly or select from a drop-down list of predefined formats. This list consists of two types of predefined formats:

- supplied predefined formats (delivered with StyleVision), and
- customized predefined formats that you define with the **Predefined Format Strings** command. These customized formats are created for the currently open SPS file—and not for the entire application. After the customized formats have been defined, the SPS File must be saved in order for the formats to be available when the file is next opened.

Creating a predefined format string

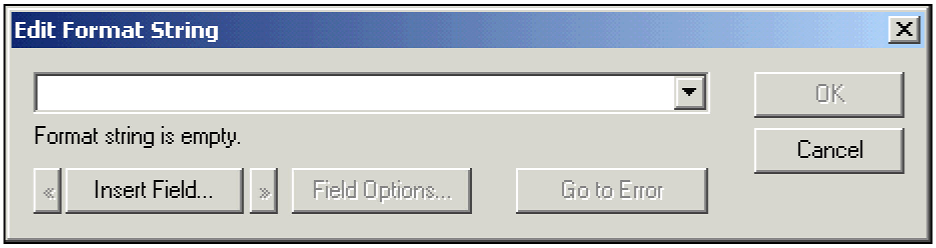
A predefined format string is specific to a datatype. To create a predefined format string, do the following:

1. Click **Properties | Predefined Format Strings...**. The following dialog appears:



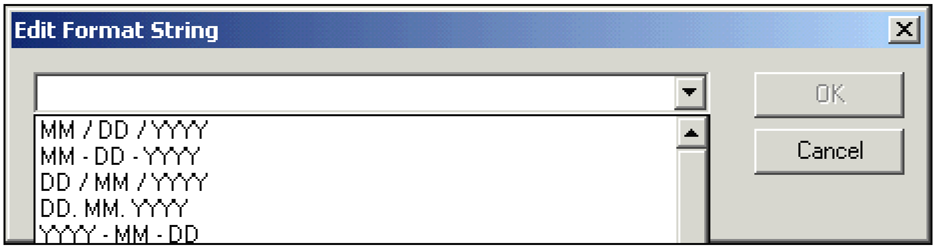
Predefined formats for: date

2. Select a datatype from the drop-down list in the combo box, and then click the Append or Insert icon as required. This pops up the Edit Format String dialog:



Format string is empty.

If you click the down arrow of the combo box, a drop-down list with the supplied predefined formats for that datatype is displayed (shown in the screenshot below).



MM / DD / YYYY
MM - DD - YYYY
DD / MM / YYYY
DD. MM. YYYY
YYYY - MM - DD

You can either select a format from the list and modify it, or you can enter a format directly into the input field. The syntax for defining a format is explained in Input Formatting. If you need help with the syntax, use the **Insert Field** and **Field Options** buttons.

3. After you have defined a format, click **OK**. The format string is added to the list of predefined formats for that datatype, and it will appear as an option in the Input Formatting dialog (of the current SPS file) when the selected element is of the corresponding datatype.

Note:

- You can add as many custom format strings for different datatypes as you want.
- The sequential order of format strings in the Predefined Format Strings dialog determines the order in which these format strings appear in the Input Formatting dialog. The customized format strings appear above the supplied predefined formats.
- To edit a custom format string, double-click the entry in the Predefined Format Strings dialog.
- To delete a custom format string, select it, and click the **Delete** button in the dialog.

11.10 Tools Menu

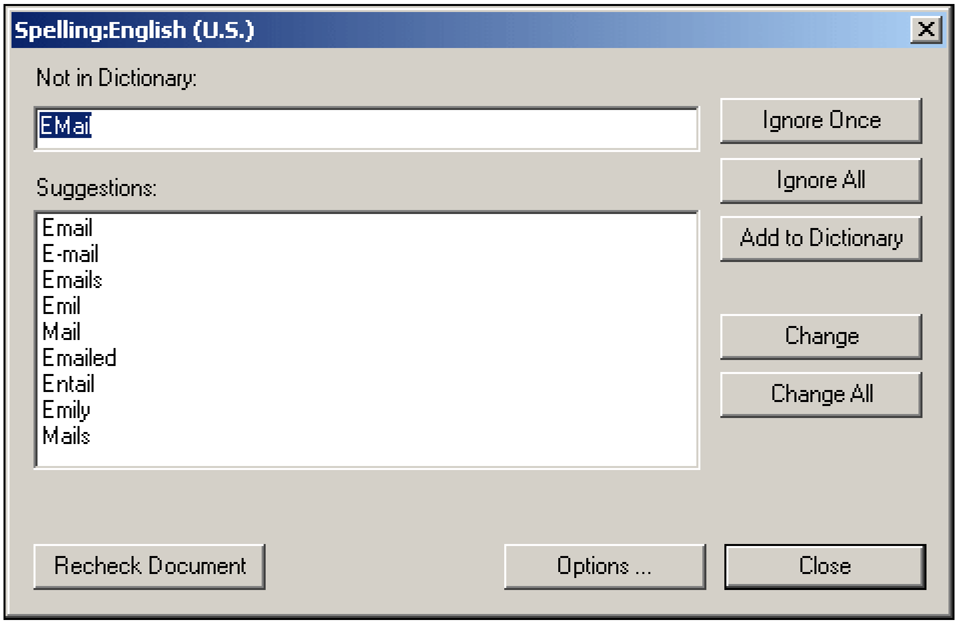
The **Tools** menu contains the spell-check command and commands that enable you to customize StyleVision.

The description of the Tools menu commands is organized into the following sub-sections:

- Spelling
- Spelling Options
- Customize
- Options

Spelling

The **Spelling (Shift+F7)** command runs a spelling check on the SPS (in Design View) or the document in Authentic View, depending on which is active. On clicking this command, the dialog shown below appears. Words that are not present in the selected dictionary are displayed, in document order and one at a time, in the Not in Dictionary field of the dialog and highlighted in the Design Document.


Spelling:English (U.S.)

Not in Dictionary: EMail

Suggestions:

Email
E-mail
Emails
Emil
Mail
Emailed
Entail
Emily
Mails

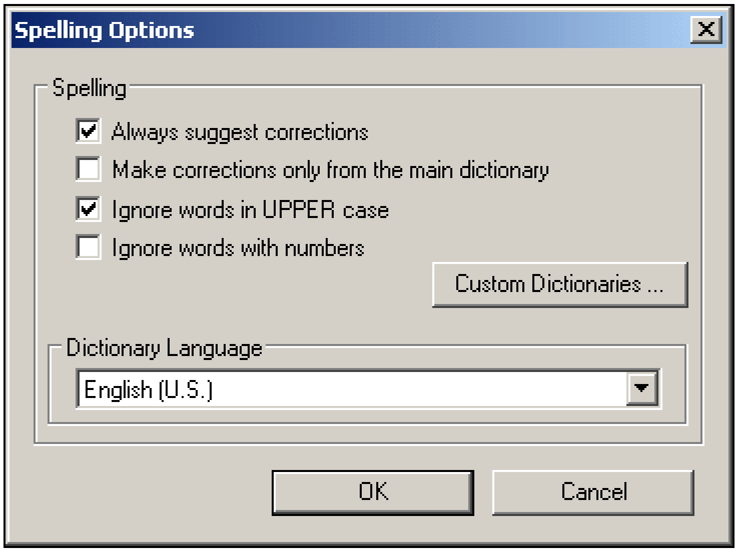
You can then select an entry from the list in the Suggestions pane and click **Change** or **Change All** to change the highlighted instance of this spelling or all its instances, respectively. (Double-clicking a word in the Suggestions list causes it to replace the unknown word.) Alternatively, you can ignore *this instance* of the unknown word (**Ignore Once**); or ignore *all instances* of this unknown word (**Ignore All**); or add this unknown word to the (default user) dictionary (**Add to Dictionary**). Adding the unknown word to the dictionary causes the spell-checker to treat the word as correct and to pass on to the next word not found in the dictionary.

After all the words not found in the dictionary have been displayed in turn, and an action taken for each, the spell-checker displays the message: "The spelling check is complete." You can then recheck the document from the beginning (**Recheck Document**) or close the dialog (**Close**).

The **Options** button opens the Spelling Options dialog, in which you can specify options for the spelling check.

Spelling Options

The **Spelling options** command opens a dialog box (shown below) in which you specify options for the spell check.



☑ Always suggest corrections
☐ Make corrections only from the main dictionary
☑ Ignore words in UPPER case
☐ Ignore words with numbers

Dictionary Language: English (U.S.)

Always suggest corrections
Selecting this option causes suggestions from the current dictionary (main dictionary plus listed custom dictionaries) to be displayed in the Suggestions list box. Otherwise no suggestions will be shown.

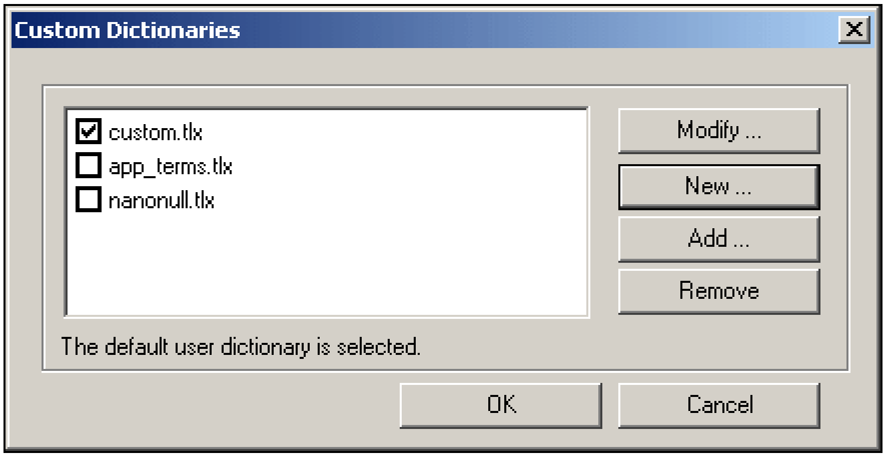
Make corrections only from main dictionary:
Selecting this option causes only the main dictionary to be used; none of the custom dictionaries is used. Additionally, the Custom Dictionaries... button is disabled, which prevents editing of the custom dictionaries.

Ignore words in UPPER case:
Selecting this option causes all upper case words to be ignored.

Ignore words with numbers:
Selecting this option causes all words containing numbers to be ignored.

Dictionaries
Each spell-checking round uses the current dictionary. The current dictionary consists of one uneditable main dictionary and the listed custom dictionaries. The number of available main dictionaries is fixed. You select a main dictionary from the drop-down menu in the Dictionary Language combo box. To edit the list of custom dictionaries used in a spell-check, or to edit the contents of a custom dictionary, click the Custom Dictionaries... button and select the required custom dictionary from the list of custom dictionaries.

When you click the **Custom Dictionaries** button, the following dialog appears:



Custom Dictionaries

☑ custom.tlx
□ app_terms.tlx
□ nanonull.tlx

The default user dictionary is selected.

Editing the Custom Dictionaries list

The listed custom dictionaries are part of the current dictionary.

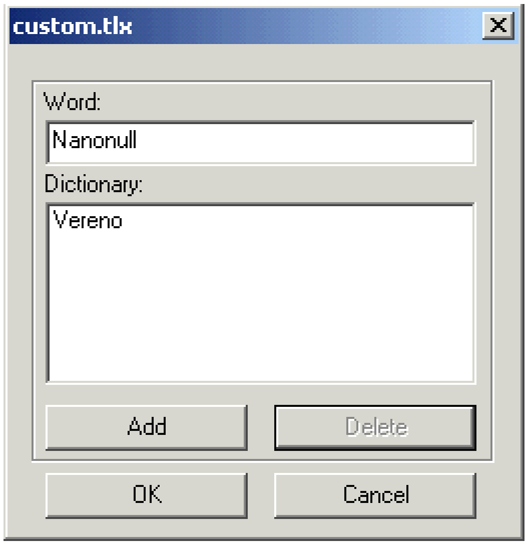
- To add an existing custom dictionary to the list, click the **Add** button; then browse for the required dictionary, and select it.
- To remove a custom dictionary from the list (and, therefore, from the current dictionary), select the dictionary to be removed and click the **Remove** button. This causes the dictionary to be removed from the list. It is, however, not deleted, and can be added to the list subsequently.
- To create a new custom dictionary and add it to the list, click the **New** button, open the folder in which the new dictionary is to be created, and give the new dictionary a name. This file must have a `.tlx` suffix.

When you start a spell check, all dictionaries listed in the Custom Dictionaries list box are searched. If you want to limit the search to specific dictionaries, use the Remove command to remove form the list those dictionaries you do not want searched.

The **default user dictionary** is the custom dictionary to which unknown words encountered in a spell-check are added when you click the Add to Dictionary command (during the spell-check). Select the default user dictionary by clicking the check box next to the dictionary you wish to make the default user dictionary.

Modifying the contents of a custom dictionary

To modify the content of a custom dictionary, click the custom dictionary to be modified, and click **Modify**. This opens the dictionary editor (shown below for the dictionary `custom.tlx`).



Word: Nanonull

Dictionary: Vereno

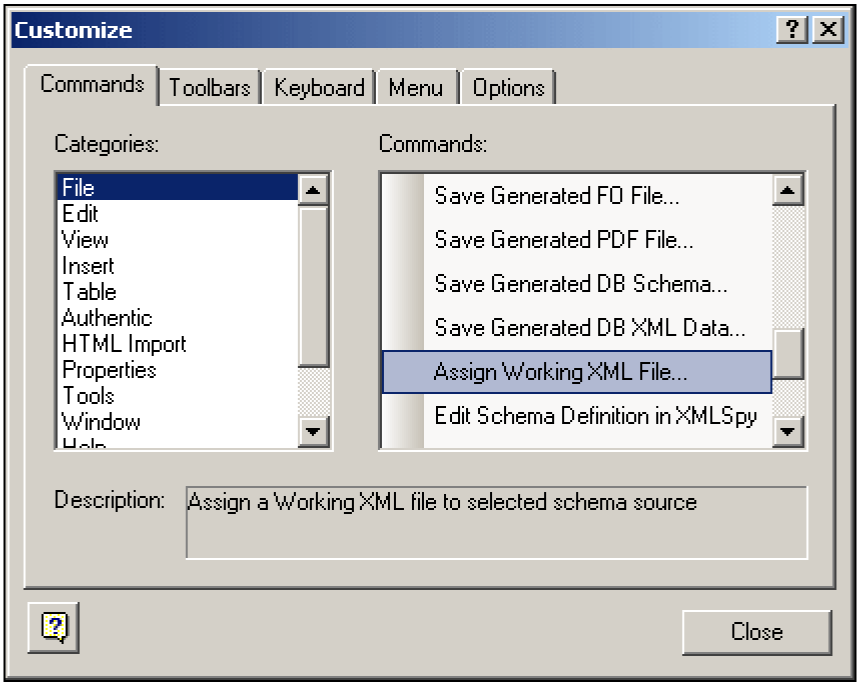
You can now add words to the dictionary and delete words. To add a word, place the cursor in the Word input field, enter the word, and click Add. To delete a word, select the word in the Dictionary pane, and click Delete.

Customize

The customize command lets you customize StyleVision to suit your personal needs.

Commands

The **Commands** tab of the Customize dialog allows you to place individual commands in the menu bar and the toolbar.


Customize
Commands
Toolbars
Keyboard
Menu
Options
Categories:
File
Edit
View
Insert
Table
Authentic
HTML Import
Properties
Tools
Window
Commands:
Save Generated FO File...
Save Generated PDF File...
Save Generated DB Schema...
Save Generated DB XML Data...
Assign Working XML File...
Edit Schema Definition in XMLSpy
Description:
Close

Assign a Working XML file to selected schema source

To add a command to the menu bar or toolbar, select the command in the Commands pane of the Commands tab, and drag it to the menu bar or toolbar. When the cursor is placed over a valid position an I-beam appears, and the command can be dropped at this location. If the location is invalid, a check mark appears. When you drop the command it is created as an icon if the command already has an associated icon; otherwise the command is created as text. After adding a command to the menu bar or toolbar, you can edit its appearance by right-clicking it and then selecting the required action.

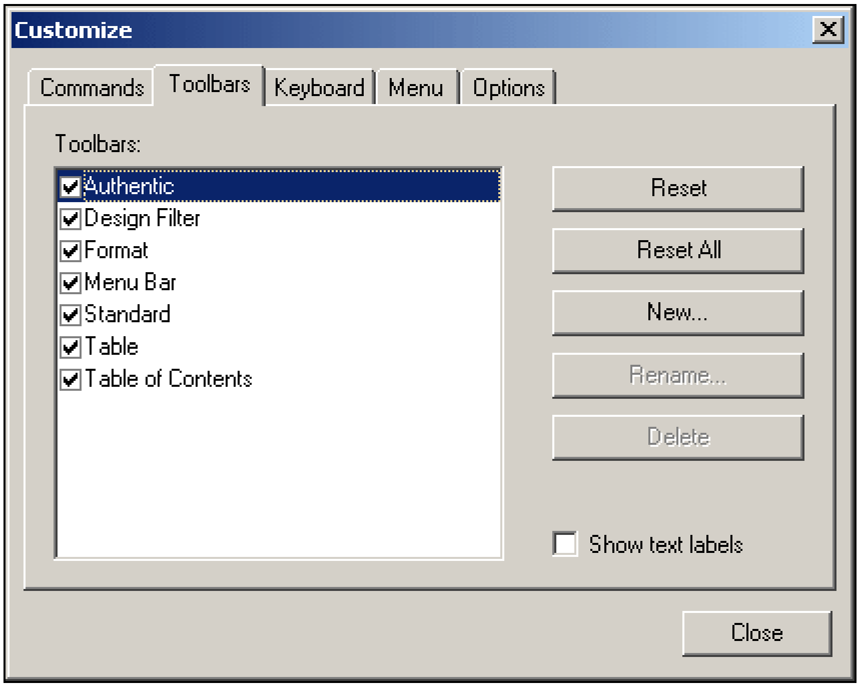
To delete a menu bar or toolbar item, with the Customize dialog open, right-click the item to be deleted, and select Delete.

Note:

- The customization described above applies to the application, and applies whether a document is open in StyleVision or not.
- To reset menus and toolbars to the state they were in when StyleVision was installed, go to the Toolbars tab and click the appropriate Reset button.

Toolbars

The **Toolbars** tab allows you to activate or deactivate specific toolbars, to show text labels for toolbar items, and to reset the menu bar and toolbars to their installation state.



Toolbars:

☑ Authentic
☑ Design Filter
☑ Format
☑ Menu Bar
☑ Standard
☑ Table
☑ Table of Contents

□ Show text labels

The StyleVision interface displays a fixed menu bar and several optional toolbars (Authentic, Design Filter, Format, Standard, Table, and Table of Contents).

Each toolbar can be divided into groups of commands. Commands can be added to a toolbar via the Commands tab. A toolbar can be dragged from its docked position to any location on the screen. Double-clicking a toolbar's (maximized or minimized) title bar docks and undocks the toolbar.

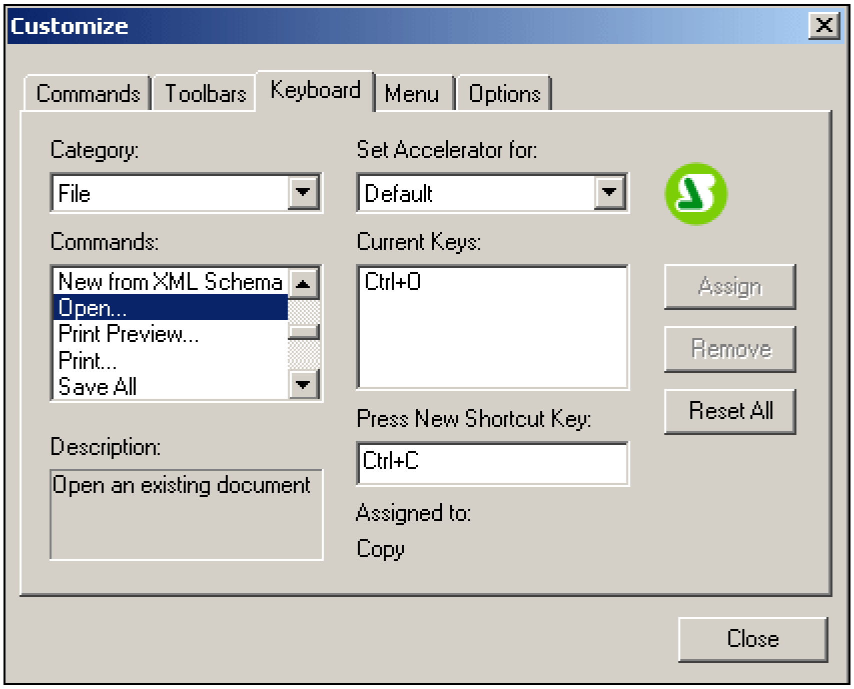
In the Toolbars tab of the Customize dialog, you can toggle a toolbar on and off by clicking in its checkbox. When a toolbar is selected (in the Toolbars tab), you can cause the text labels of that toolbar's items to be displayed by clicking the **Show text labels** check box. You can also reset a selected toolbar to the state it was in when StyleVision was installed by clicking the **Reset** button. You can reset all toolbars and the menu bar by clicking the **Reset All** button.

Menu Bar

Commands can be added to, and items deleted from, the menu bar: see Commands above. To reset the menu bar to the state it was in when StyleVision was installed, select Menu Bar in the Toolbars tab of the Customize dialog, and click the **Reset** button. (Clicking the **Reset All** button will reset the toolbars as well.)

Keyboard

The **Keyboard** tab allows you to define (or change) keyboard shortcuts for any StyleVision command.



Category: File

Set Accelerator for: Default

Commands:
New from XML Schema
Open...
Print Preview...
Print...
Save All

Current Keys: Ctrl+O

Description: Open an existing document

Press New Shortcut Key: Ctrl+C

Assigned to: Copy

To assign a shortcut to a command

1. Select the category in which the command is by using the Category combo box.
2. Select the command you want to assign a shortcut to in the Commands list box.
3. Click in the Press New Shortcut Key input field, and press the shortcut keys that are to activate the command. The shortcut immediately appears in the Press New Shortcut Key input field. If this shortcut has already been assigned to a command, then that command is displayed below the input field. (For example, in the screenshot above, Ctrl+C has already been assigned to the Copy command and cannot be assigned to the Open File command.) To clear the New Shortcut Key input field, press any of the control keys, Ctrl, Alt, or Shift.
4. Click the **Assign** button to permanently assign the shortcut. The shortcut now appears in the Current Keys text box.

To de-assign (or delete) a shortcut

1. Select the command for which the shortcut is to be deleted.
2. Click the shortcut you want to delete in the Current Keys list box.
3. Click the **Remove** button (which has now become active).

To reset all keyboard assignments

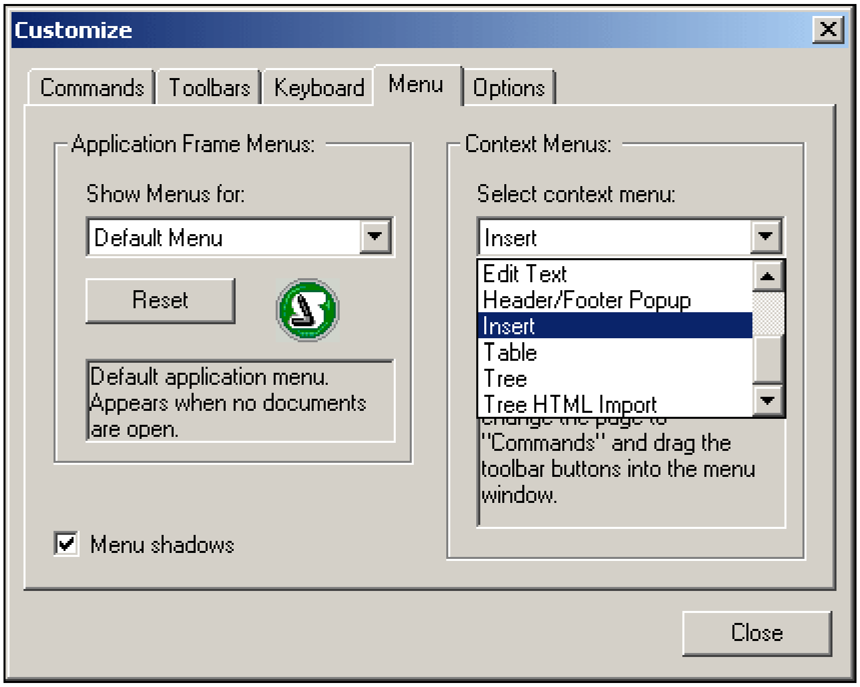
1. Click the **Reset All** button to go back to the original, installation-time shortcuts. A dialog box appears prompting you to confirm whether you want to reset all keyboard assignments.
2. Click Yes if you want to reset all keyboard assignments.

Set accelerator for

Currently no function is available.

Menu

The **Menu** tab allows you to customize the main menu bar as well as the context menus (right-click menus).



Show Menus for: Default Menu

Default application menu. Appears when no documents are open.

Select context menu: Insert

change the page to "Commands" and drag the toolbar buttons into the menu window.

☑ Menu shadows

To customize a menu

1. Select the menu bar you want to customize (Default Menu currently).
2. Click the **Commands** tab, and drag the commands to the menu bar of your choice.

To delete commands from a menu

1. Click right on the command or icon representing the command, and
2. Select the **Delete** option from the popup menu,

   or,

1. Select **Tools | Customize** to open the Customize dialog box, and
2. Drag the command away from the menu and drop it as soon as the check mark icon appears below the mouse pointer.

To reset either of the menu bars

1. Select the Default Menu entry in the combo box)
2. Click the **Reset** button just below the menu name. A prompt appears asking if you are sure you want to reset the menu bar.

To customize a context menu (a right-click menu)

1. Select the context menu from the combo box.
2. Click the **Commands** tab and drag the commands to the context menu that is now open.

To delete commands from a context menu

1. Click right on the command or icon representing the command, and
2. Select the **Delete** option from the popup menu

   or

1. Select **Tools | Customize** to open the Customize dialog box, and
2. Drag the command away from the context menu and drop it as soon as the check mark icon appears below the mouse pointer.

To reset a context menu

1. Select the context menu from the combo box, and
2. Click the **Reset** button just below the context menu name. A prompt appears asking if

you are sure you want to reset the context menu.

To close a context menu window

- Click on the **Close icon** at the top right of the title bar, or
- Click the Close button of the Customize dialog box.

Menu animations

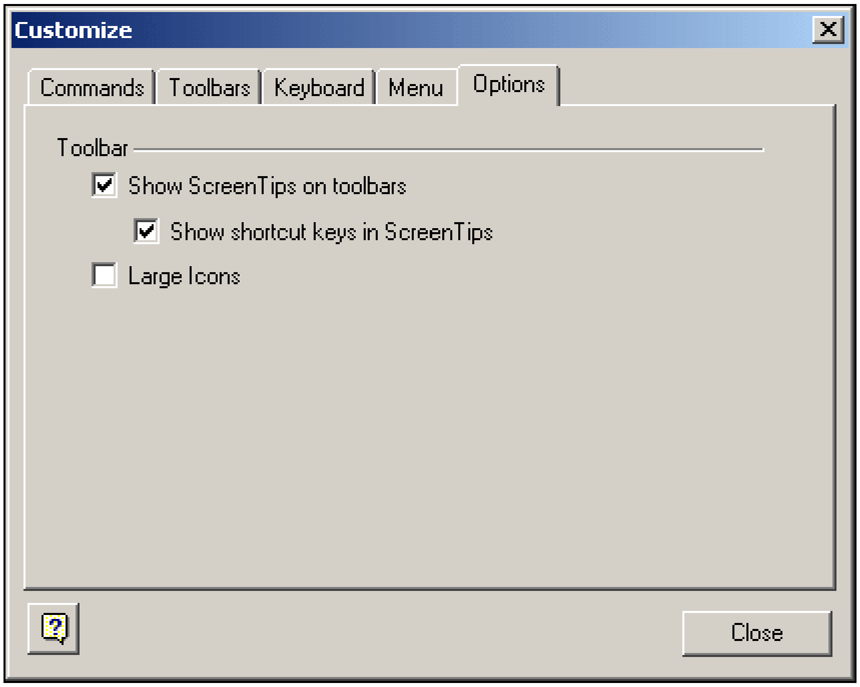
The menu animation option specifies the way a menu is displayed when a menu is clicked. Select an option from the drop-down list of menu animations.

Menu shadows

If you wish to have menus displayed with a shadow around it, select this option. All menus will then have a shadow.

Options

The **Options** tab allows you to customize additional features of the toolbar.



Toolbar

☑ Show ScreenTips on toolbars

☑ Show shortcut keys in ScreenTips

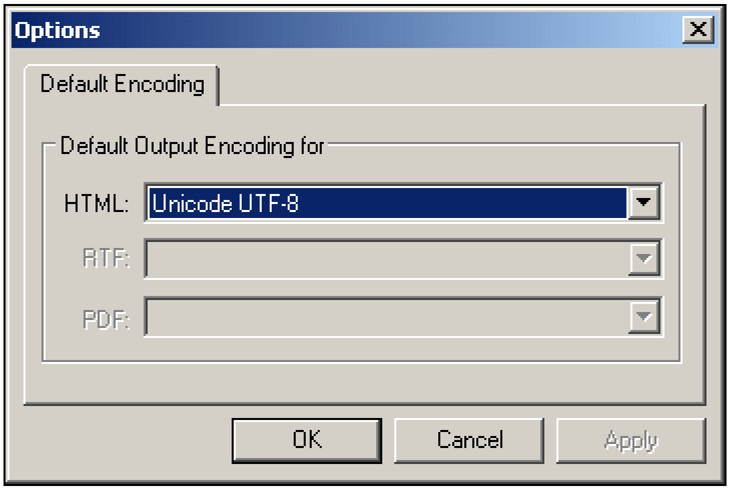
□ Large Icons

Screen Tips for toolbar items will be displayed if the Show Screen Tips option is checked. The Screen Tips option has a sub-option for whether shortcuts (where available) are displayed in the Screen Tips or not.

Toolbar items can also be displayed as large icons. To do this, check the Large Icons option.

Options

The **Options** command opens a dialog in which you can specify the encoding of the HTML output file.



To set the encoding of the output HTML file, open the dropdown menu of the combo box and select the desired option from the list of encoding options, and click OK. Every new SPS you create from this point on, will set the HTML output encoding as defined in this tab.

In the XSLT-for-HTML, the output encoding information is registered at the following locations:

- In the `encoding` attribute of the stylesheet's `xsl:output` element:
  `<xsl:output version="1.0" **encoding="UTF-8"** indent="no" omit-xml-declaration="no" media-type="text/html" />`
- In the `charset` attribute of the `content-type meta` element in the HTML header:
  `<meta http-equiv="Content-Type" content="text/html; **charset=UTF-8"** />`

Note: This setting is the default encoding for HTML output and will be used for new SPSs. You cannot change the encoding of the currently open SPS using this dialog. To change the encoding of the currently open SPS, use the File | Properties command.

11.11 Window Menu

The **Window menu** has commands to specify how StyleVision windows should be displayed in the GUI (cascaded, tiled, or maximized). To maximize a window, click the maximize button of that window.

Additionally, all currently open document windows are listed in this menu by document name, with the active window being checked. To make another window active, click the name of the window you wish to make active.

Windows dialog
At the bottom of the list of open windows is an entry for the Windows dialog. Clicking this entry opens the Windows dialog, which displays a list of all open windows and provides commands that can be applied to the selected window/s. (A window is selected by clicking on its name.)

Warning: To exit the Windows dialog, click OK; do **not** click the Close Window(s) button. The Close Window(s) button closes the window/s currently selected in the Windows dialog.

11.12 Help Menu

The **Help** menu contains commands to access the onscreen help manual for StyleVision, commands to provide information about StyleVision, and links to support pages on the Altova web site. The Help menu also contains the Registration dialog, which lets you enter your license key-code once you have purchased the product.

The description of the Help menu commands is organized into the following sub-sections:

- Table of Contents, Index, Search
- Registration, Order Form
- Other Commands

Table of Contents, Index, Search

The **Table of Contents** command opens the onscreen help manual for StyleVision with the Table of Contents displayed in the left-hand-side pane of the Help window. The Table of Contents provides a good overview of the entire Help document. Clicking an entry in the Table of Contents takes you to that topic.

The **Index** command opens the onscreen help manual for StyleVision with the Keyword Index displayed in the left-hand-side pane of the Help window. The index lists keywords and lets you navigate to a topic by double-clicking the keyword. If a keyword is linked to more than one topic, you are presented with a list of the topics to choose from.

The **Search** command opens the onscreen help manual for StyleVision with the Search dialog displayed in the left-hand-side pane of the Help window. To search for a term, enter the term in the input field, and press Return. The Help system performs a full-text search on the entire Help documentation and returns a list of hits. Double-click any item to display that item.

Activation, Order Form, Registration, Updates

Software Activation

After you download your Altova product software, you can activate it using either a free evaluation key or a purchased permanent license key.

- ***Free evaluation key.*** When you first start the software after downloading and installing it, the Software Activation dialog will pop up. In it is a button to request a free evaluation key-code. Enter your name, company, and e-mail address in the dialog that appears, and click Request Now! The evaluation key is sent to the e-mail address you entered and should reach you in a few minutes. Now enter the key in the key-code field of the Software Activation dialog box and click **OK** to start working with your Altova product. The software will be unlocked for a period of 30 days.
- ***Permanent license key.*** The Software Activation dialog contains a button to purchase a permanent license key. Clicking this button takes you to Altova's online shop, where you can purchase a permanent license key for your product. There are two types of permanent license: single-user and multi-user. Both will be sent to you by e-mail. A *single-user license* contains your license-data and includes your name, company, e-mail, and key-code.A *multi-user license* contains your license-data and includes your company name and key-code. Note that your license agreement does not allow you to install more than the licensed number of copies of your Altova software on the computers in your organization (per-seat license). Please make sure that you enter the data required in the registration dialog exactly as given in your license e-mail.

Note: When you enter your license information in the Software Activation dialog, ensure that you enter the data exactly as given in your license e-mail. For multi-user licenses, each user should enter his or her own name in the Name field.

The Software Activation dialog can be accessed at any time by clicking the **Help | Software Activation** command.

Order Form

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

Registration

The first time you start your Altova software after having activated it, a dialog appears asking whether you would like to register your product. There are three buttons in this dialog:

- **OK:** Takes you to the Registration Form
- **Remind Me Later:** Pops up a dialog in which you can select when you wish to be next reminded.
- **Cancel:** Closes the dialog and suppresses it in future. If you wish to register at a later time, you can use the **Help | Registration** command.

Check for Updates

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

Other Commands

The **Support Center** command is a link to the Altova Support Center on the Internet. The Support Center provides FAQs, discussion forums where problems are discussed, and access to Altova's technical support staff.

The **FAQ on the Web** command is a link to Altova's FAQ database on the Internet. The FAQ database is constantly updated as Altova support staff encounter new issues raised by customers.

The **Components Download** command is a link to Altova's Component Download Center on the Internet. From here you will be able to download a variety of companion software to use with Altova products. Such software ranges from XSLT and XSL-FO processors to Application Server Platforms. The software available at the Component Download Center is typically free of charge.

The **StyleVision on the Internet** command is a link to the Altova website on the Internet. You can learn more about StyleVision and related technologies and products at the Altova website.

The **StyleVision Training** command is a link to the Online Training page at the Altova website. Here you can select from online courses conducted by Altova's expert trainers.

The **About StyleVision** command displays the splash window and version number of your product.

Chapter 12

Appendices

12 Appendices

These appendices contain (i) information about the XSLT Engines used in StyleVision; (ii) information about the conversion of DB datatypes to XML Schema datatypes; (iii) technical information about StyleVision; and (iv) licensing information for StyleVision. Each appendix contains the sub-sections listed below:

XSLT Engine Information
Provides implementation-specific information about the Altova XSLT Engines, which are used by StyleVision to generate output.

- Altova XSLT 1.0 Engine
- Altova XSLT 2.0 Engine
- XPath 2.0 and XQuery 1.0 Functions

DatatypesDB2XSD
When DB fields are converted to XML nodes, the DB datatypes are converted to XML Schema datatypes. This appendix lists the mappings for the following source DBs.

- MS Access
- MS SQL Server
- MySQL
- Oracle
- ODBC
- ADO
- Sybase

Technical Data
Provides technical information about StyleVision.

- OS and memory requirements
- Altova XML Parser
- Altova XSLT and XQuery Engines
- Unicode support
- Internet usage

License Information
Contains information about the way StyleVision is distributed and about its licensing.

- Electronic software distribution
- License metering
- Copyright
- End User License Agreement

12.1 XSLT Engine Information

This section contains information about implementation-specific features of the Altova XSLT 1.0 Engine and Altova XSLT 2.0 Engine.

XSLT 1.0 Engine: Implementation Information

The Altova XSLT 1.0 Engine is built into Altova's XMLSpy, StyleVision, Authentic, and MapForce XML products. It is also available in the free AltovaXML package. The Altova XSLT 1.0 Engine implements and conforms to the World Wide Web Consortium's XSLT 1.0 Recommendation of 16 November 1999 and XPath 1.0 Recommendation of 16 November 1999 . Limitations and implementation-specific behavior are listed below.

Limitations

- The **`xsl:preserve-space`** and **`xsl:strip-space`** elements are not supported.
- When the `method` attribute of `xsl:output` is set to HTML, or if HTML output is selected by default, then special characters in the XML or XSLT file are inserted in the HTML document directly as special characters; they are not inserted as HTML character references in the output. For instance, the character `&#160;` (the decimal character reference for a non-breaking space) is not inserted as `&nbsp;` in the HTML code, but directly as a non-breaking space.

Implementation's handling of whitespace-only nodes in source XML document
The XML data (and, consequently, the XML Infoset) that is passed to the Altova XSLT 1.0 Engine is stripped of boundary-whitespace-only text nodes. (A boundary-whitespace-only text node is a child whitespace-only text node that occurs between two elements within an element of mixed content.) This stripping may have an effect on the value returned by the `fn:position()`, `fn:last()`, and `fn:count()` functions.

For any node selection that selects text nodes also, boundary-whitespace-only text nodes would typically also be included in the selection. However, since the XML Infoset used by the Altova engines has boundary-whitespace-only text nodes stripped from it, these nodes are not present in the XML Infoset. As a result, the size of the selection and the numbering of nodes in the selection will be different than that for a selection which included these text nodes. The `fn:position()`, `fn:last()`, and `fn:count()` functions, therefore, could produce results that are different from those produced by some other processors.

A situation in which boundary-whitespace-only text nodes are evaluated as siblings of other elements arises most commonly when `xsl:apply-templates` is used to apply templates. When the `fn:position()`, `fn:last()`, and `fn:count()` functions are used in patterns with a name test (for example, `para[3]`, which is short for `para[position()=3]`), boundary-whitespace-only nodes are irrelevant since only the named elements (`para` in the above example) are selected. (Note, however, that boundary-whitespace-only nodes **are** relevant in patterns that use the wildcard, for example, `*[10]`.)

Note: If a boundary-whitespace-only text node is required in the output, then insert the required whitespace within one of the two adjoining child elements. For example, the XML fragment:

```
<para>This is <b>bold</b> <i>italic</>.</para>
```

when processed with the XSLT template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```

will produce:

```
This is bolditalic.
```

To get a space between `bold` and `italic` in the output, insert a space character within either the `<b>` or `<i>` elements in the XML source. For example:

```
<para>This is <b>bold</b> <i> italic</>.</para> or
<para>This is <b>bold&#x20;</b> <i>italic</>.</para> or
<para>This is <b>bold</b><i>&#x20;italic</>.</para>
```

When such an XML fragment is processed with the same XSLT template given above, it will produce:

```
This is bold italic.
```

XSLT 2.0 Engine: Implementation Information

The Altova XSLT 2.0 Engine is built into Altova's XMLSpy, StyleVision, Authentic, and MapForce XML products. It is also available in the free AltovaXML package. This section describes the engine's implementation-specific aspects of behavior. It starts with a section giving general information about the engine, and then goes on to list the implementation-specific behavior of XSLT 2.0 functions.

For information about implementation-specific behavior of XPath 2.0 functions, see the section, XPath 2.0 and XQuery 1.0 Functions.

General Information

The Altova XSLT 2.0 Engine conforms to the World Wide Web Consortium's (W3C's) XSLT 2.0 Recommendation of 23 January 2007. Note the following general information about the engine.

Backwards Compatibility

The Altova XSLT 2.0 Engine is backwards compatible. The only time the backwards compatibility of the XSLT 2.0 Engine comes into play is when using the XSLT 2.0 Engine of Altova XML to process an XSLT 1.0 stylesheet. Note that there could be differences in the outputs produced by the XSLT 1.0 Engine and the backwards-compatible XSLT 2.0 Engine.

In all other Altova products, the backwards-compatibility issue never arises. This is because these products automatically select the appropriate engine for the transformation. For example, consider that in XMLSpy you specify that a certain XML document be processed with an XSLT 1.0 stylesheet. When the transformation command is invoked, XMLSpy automatically selects the XSLT 1.0 Engine of XMLSpy to carry out the transformation.

Note: The stylesheet version is specified in the `version` attribute of the `stylesheet` or `transform` element of the stylesheet.

Namespaces

Your XSLT 2.0 stylesheet should declare the following namespaces in order for you to be able to use the type constructors and functions available in XSLT 2.0. The prefixes given below are conventionally used; you could use alternative prefixes if you wish.

Namespace Name	Prefix	Namespace URI
XML Schema types	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0 functions	`fn:`	`http://www.w3.org/2005/xpath-functions`

Typically, these namespaces will be declared on the `xsl:stylesheet` or `xsl:transform` element, as shown in the following listing:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   ...
</xsl:stylesheet>
```

The following points should be noted:

- The Altova XSLT 2.0 Engine uses the XPath 2.0 and XQuery 1.0 Functions namespace (listed in the table above) as its **default functions namespace**. So you can use XPath 2.0 and XSLT 2.0 functions in your stylesheet without any prefix. If you declare the XPath 2.0 Functions namespace in your stylesheet with a prefix, then you can additionally use the prefix assigned in the declaration.
- When using type constructors and types from the XML Schema namespace, the prefix used in the namespace declaration must be used when calling the type constructor (for example, `xs:date`).
- With the CRs of 23 January 2007, the `untypedAtomic` and duration datatypes (`dayTimeDuration` and `yearMonthDuration`), which were formerly in the XPath Datatypes namespace (typically prefixed `xdt:`) have been moved to the XML Schema namespace.

- Some XPath 2.0 functions have the same name as XML Schema datatypes. For example, for the XPath functions `fn:string` and `fn:boolean` there exist XML Schema datatypes with the same local names: `xs:string` and `xs:boolean`. So if you were to use the XPath expression `string('Hello')`, the expression evaluates as `fn:string('Hello')`—not as `xs:string('Hello')`.

Schema-awareness

The Altova XSLT 2.0 Engine is schema-aware.

Whitespace in XML document

By default, the Altova XSLT 2.0 Engine strips all boundary whitespace from boundary-whitespace-only nodes in the source XML document. The removal of this whitespace affects the values that the `fn:position()`, `fn:last()`, `fn:count()`, and `fn:deep-equal()` functions return. For more details, see Whitespace-only Nodes in XML Document in the XPath 2.0 and XQuery 1.0 Functions section.

Note: If a boundary-whitespace-only text node is required in the output, then insert the required whitespace within one of the two adjoining child elements. For example, the XML fragment:

```
<para>This is <b>bold</b> <i>italic</>.</para>
```

when processed with the XSLT template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```

will produce:

```
This is bolditalic.
```

To get a space between `bold` and `italic` in the output, insert a space character within either the `<b>` or `<i>` elements in the XML source. For example:

```
<para>This is <b>bold</b> <i> italic</>.</para> or
<para>This is <b>bold&#x20;</b> <i>italic</>.</para> or
<para>This is <b>bold</b><i>&#x20;italic</>.</para>
```

When such an XML fragment is processed with the same XSLT template given above, it will produce:

```
This is bold italic.
```

XSLT 2.0 elements and functions

Limitations and implementation-specific behavior of XSLT 2.0 elements and functions are listed in the section XSLT 2.0 Elements and Functions.

XPath 2.0 functions

Implementation-specific behavior of XPath 2.0 functions is listed in the section XPath 2.0 and XQuery 1.0 Functions.

XSLT 2.0 Elements and Functions

Limitations
The **`xsl: preserve-space`** and **`xsl: strip-space`** elements are not supported.

Implementation-specific behavior
Given below is a description of how the Altova XSLT 2.0 Engine handles implementation-specific aspects of the behavior of certain XSLT 2.0 functions.

`function-available`
The function tests for the availability of XSLT 2.0 functions, not for the availability of XPath 2.0 functions.

`unparsed-text`
The `href` attribute accepts (i) relative paths for files in the base-uri folder, and (ii) absolute paths with or without the `file: //` protocol.

XPath 2.0 and XQuery 1.0 Functions

XPath 2.0 and XQuery 1.0 functions are evaluated by:

- the **Altova XPath 2.0 Engine**, which (i) is a component of the Altova XSLT 2.0 Engine, and (ii) is used in the XPath Evaluator of Altova's XMLSpy product to evaluate XPath expressions with respect to the XML document that is active in the XMLSpy interface.
- the **Altova XQuery 1.0 Engine**.

This section describes how XPath 2.0 and XQuery 1.0 functions are handled by the Altova XPath 2.0 Engine and Altova XQuery 1.0 Engine. Only those functions are listed, for which the behavior is implementation-specific, or where the behavior of an individual function is different in any of the three environments in which these functions are used (that is, in XSLT 2.0, in XQuery 1.0, and in the XPath Evaluator of XMLSpy). Note that this section does not describe how to use these functions. For more information about the usage of functions, see the World Wide Web Consortium's (W3C's) XQuery 1.0 and XPath 2.0 Functions and Operators Recommendation of 23 January 2007.

General Information

Standards conformance

- The Altova XPath 2.0 Engine implements the World Wide Web Consortium's (W3C's) XPath 2.0 Recommendation of 23 January 2007. The Altova XQuery 1.0 Engine implements the World Wide Web Consortium's (W3C's) XQuery 1.0 Recommendation of 23 January 2007. The XPath 2.0 and XQuery 1.0 functions support in these two engines is compliant with the XQuery 1.0 and XPath 2.0 Functions and Operators Recommendation of 23 January 2007.
- The Altova XPath 2.0 Engine conforms to the rules of XML 1.0 (Fourth Edition) and XML Namespaces (1.0).

Default functions namespace

The default functions namespace has been set to comply with that specified in the standard. Functions can therefore be called without a prefix.

Boundary-whitespace-only nodes in source XML document

The XML data (and, consequently, the XML Infoset) that is passed to the Altova XPath 2.0 Engine and Altova XQuery 1.0 Engine is stripped of boundary-whitespace-only text nodes. (A boundary-whitespace-only text node is a child whitespace-only text node that occurs between two elements within an element of mixed content.) This stripping has an effect on the value returned by the `fn:position()`, `fn:last()`, `fn:count()`, and `fn:deep-equal()` functions.

For any node selection that selects text nodes also, boundary-whitespace-only text nodes would typically also be included in the selection. However, since the XML Infoset used by the Altova engines has boundary-whitespace-only text nodes stripped from it, these nodes are not present in the XML Infoset. As a result, the size of the selection and the numbering of nodes in the selection will be different than that for a selection which included these text nodes. The `fn:position()`, `fn:last()`, `fn:count()`, and `fn:deep-equal()` functions, therefore, could produce results that are different from those produced by some other processors.

A situation in which boundary-whitespace-only text nodes are evaluated as siblings of other elements arises most commonly when `xsl:apply-templates` is used to apply templates. When the `fn:position()`, `fn:last()`, and `fn:count()` functions are used in patterns with a name test (for example, `para[3]`, which is short for `para[position()=3]`), boundary-whitespace-only nodes are irrelevant since only the named elements (`para` in the above example) are selected. (Note, however, that boundary-whitespace-only nodes **are** relevant in patterns that use the wildcard, for example, `*[10]`.)

Numeric notation

On output, when an `xs:double` is converted to a string, scientific notation (for example, `1.0E12`) is used when the absolute value is less than 0.000001 or greater than 1,000,000. Otherwise decimal or integer notation is used.

Precision of `xs:decimal`

The precision refers to the number of digits in the number, and a minimum of 18 digits is required by the specification. For division operations that produce a result of type `xs:decimal`, the precision is 19 digits after the decimal point with no rounding.

Implicit timezone

When two `date`, `time`, or `dateTime` values need to be compared, the timezone of the values being compared need to be known. When the timezone is not explicitly given in such a value, the implicit timezone is used. The implicit timezone is taken from the system clock, and its value can be checked with the `fn:implicit-timezone()` function.

Collations

Only the Unicode codepoint collation is supported. No other collations can be used. String comparisons, including for the `fn:max` and `fn:min` functions, are based on this collation.

Namespace axis

The namespace axis is deprecated in XPath 2.0. Use of the namespace axis is, however, supported. To access namespace information with XPath 2.0 mechanisms, use the `fn:in-scope-prefixes()`, `fn:namespace-uri()` and `fn:namespace-uri-for-prefix()` functions.

Static typing extensions

The optional static type checking feature is not supported.

Functions Support

The table below lists (in alphabetical order) the implementation-specific behavior of certain functions. The following general points should be noted:

- In general, when a function expects a sequence of one item as an argument, and a sequence of more than one item is submitted, then an error is returned.
- All string comparisons are done using the Unicode codepoint collation.
- Results that are QNames are serialized in the form `[prefix:]localname`.

Function Name	Notes
`base-uri`	• If external entities are used in the source XML document and if a node in the external entity is specified as the input node argument of the `base-uri()` function, it is still the base URI of the including XML document that is used—not the base URI of the external entity. • The base URI of a node in the XML document can be modified using the `xml:base` attribute.
`collection`	• The `collection()` function is a mapping of form `(string, nodes)`, currently called `available collections` and left empty by the external environment. The function therefore returns either (i) the empty sequence (when called with no argument or with an empty sequence), or (ii) an error (when called with a non-empty argument).
`count`	• See note on whitespace in the General Information section.

contd./

Function Name	Notes
`current-date`, `current-dateTime`, `current-time`	• The current date and time is taken from the system clock. • The timezone is taken from the implicit timezone provided by the evaluation context; the implicit timezone is taken from the system clock. • The timezone is always specified in the result.
`deep-equal`	• See note on whitespace in the General Information section.
`doc`	• An error is raised only if no XML file is available at the specified location or if the file is not well-formed. The file is validated if a schema is available. If the file is not valid, the invalid file is loaded without schema information.

`id`	• In a well-formed but invalid document that contains two or more elements having the same ID value, the first element in document order is returned.
`in-scope-prefixes`	• Only default namespaces may be undeclared in the XML document. However, even when a default namespace is undeclared on an element node, the prefix for the default namespace, which is the zero-length string, is returned for that node.
`last`	• See note on whitespace in the General Information section.
`lower-case`	• The ASCII character set only is supported.
`normalize-unicode`	• Not supported.
`position`	• See note on whitespace in the General Information section.

contd. /

Function Name	Notes
`resolve-uri`	• If the second, optional argument is omitted, the URI to be resolved (the first argument) is resolved against the base URI from the static context, which is the URI of the XSLT stylesheet or the base URI given in the prolog of the XQuery document. • The relative URI (the first argument) is appended after the last `"/"` in the path notation of the base URI notation. • If the value of the first argument is the zero-length string, the base URI from the static context is returned, and this URI includes the file name of the document from which the base URI of the static context is derived (e.g. the XSLT or XML file).
`static-base-uri`	• The base URI from the static context is the base URI of the XSLT stylesheet or the base URI specified in the prolog of the XQuery document. • When using XPath Evaluator in the XMLSpy IDE, the base URI from the static context is the URI of the active XML document.
`upper-case`	• The ASCII character set only is supported.

12.2 Datatypes in DB-Generated XML Schemas

When an XML Schema is generated from a database (DB), the datatypes specific to that DB are converted to XML Schema datatypes. The mappings of DB datatypes to XML Schema datatypes for commonly used DBs are given in this Appendix. Select from the list below.

- MS Access
- MS SQL Server
- MySQL
- Oracle
- ODBC
- ADO
- Sybase

MS Access

When an XML Schema is generated from an MS Access database (DB), the MS Access DB datatypes are converted to XML Schema datatypes as listed in the table below.

MS Access Datatype	XML Schema Datatype
GUID	xs:ID
char	xs:string
varchar	xs:string
memo	xs:string
bit	xs:boolean
Number(single)	xs:float
Number(double)	xs:double
Decimal	xs:decimal
Currency	xs:decimal
Date/Time	xs:dateTime
Number(Long Integer)	xs:integer
Number(Integer)	xs:short
Number(Byte)	xs:byte
OLE Object	xs:base64Binary

MS SQL Server

When an XML Schema is generated from an MS SQL Server database (DB), the MS SQL Server DB datatypes are converted to XML Schema datatypes as listed in the table below.

MS SQL Server Datatype	XML Schema Datatype
uniqueidentifier	xs:ID
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
ntext	xs:string
sysname	xs:string
bit	xs:boolean
real	xs:float
float	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
binary	xs:base64Binary
varbinary	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

MySQL

When an XML Schema is generated from a MySQL database (DB), the MySQL DB datatypes are converted to XML Schema datatypes as listed in the table below.

My SQL Datatype	XML Schema Datatype
char	xs:string
varchar	xs:string
text	xs:string
tinytext	xs:string
mediumtext	xs:string
longtext	xs:string
tinyint(1)	xs:boolean
float	xs:float
double	xs:double
decimal	xs:decimal
datetime	xs:dateTime
blob	xs:base64Binary
tinyblob	xs:base64Binary
mediumblob	xs:base64Binary
longblob	xs:base64Binary
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

Oracle

When an XML Schema is generated from an Oracle database (DB), the Oracle DB datatypes are converted to XML Schema datatypes as listed in the table below.

Oracle Datatype	XML Schema Datatype
ROWID	xs:ID
CHAR	xs:string
NCHAR	xs:string
VARCHAR2	xs:string
NVARCHAR2	xs:string
CLOB	xs:string
NCLOB	xs:string
NUMBER (with check constraint applied)*	xs:boolean
NUMBER	xs:decimal
FLOAT	xs:double
DATE	xs:dateTime
INTERVAL YEAR TO MONTH	xs:gYearMonth
BLOB	xs:base64Binary

* If a check constraint is applied to a column of datatype NUMBER, and the check constraint checks for the values 0 or 1, then the NUMBER datatype for this column will be converted to an XML Schema datatype of xs:boolean. This mechanism is useful for generating an xs:boolean datatype in the generated XML Schema.

ODBC

When an XML Schema is generated from an ODBC database (DB), the ODBC DB datatypes are converted to XML Schema datatypes as listed in the table below.

ODBC Datatype	XML Schema Datatype
SQL_GUID	xs:ID
SQL_CHAR	xs:string
SQL_VARCHAR	xs:string
SQL_LONGVARCHAR	xs:string
SQL_BIT	xs:boolean
SQL_REAL	xs:float
SQL_DOUBLE	xs:double
SQL_DECIMAL	xs:decimal
SQL_TIMESTAMP	xs:dateTime
SQL_DATE	xs:date
SQL_BINARY	xs:base64Binary
SQL_VARBINARY	xs:base64Binary
SQL_LONGVARBINARY	xs:base64Binary
SQL_INTEGER	xs:integer
SQL_SMALLINT	xs:short
SQL_BIGINT	xs:long
SQL_TINYINT	xs:byte

ADO

When an XML Schema is generated from an ADO database (DB), the ADO DB datatypes are converted to XML Schema datatypes as listed in the table below.

ADO Datatype	XML Schema Datatype
adGUID	xs:ID
adChar	xs:string
adWChar	xs:string
adVarChar	xs:string
adWVarChar	xs:string
adLongVarChar	xs:string
adWLongVarChar	xs:string
adVarWChar	xs:string
adBoolean	xs:boolean
adSingle	xs:float
adDouble	xs:double
adNumeric	xs:decimal
adCurrency	xs:decimal
adDBTimeStamp	xs:dateTime
adDate	xs:date
adBinary	xs:base64Binary
adVarBinary	xs:base64Binary
adLongVarBinary	xs:base64Binary
adInteger	xs:Integer
adUnsignedInt	xs:unsignedInt
adSmallInt	xs:short
adUnsignedSmallInt	xs:unsignedShort
adBigInt	xs:long
adUnsignedBigInt	xs:unsignedLong
adTinyInt	xs:byte
adUnsignedTinyInt	xs:unsignedByte

Sybase

When an XML Schema is generated from a Sybase database (DB), the Sybase DB datatypes are converted to XML Schema datatypes as listed in the table below.

Sybase Datatype	XML Schema Datatype
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
sysname-varchar(30)	xs:string
bit	xs:boolean
real	xs:float
float	xs:float
double	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
timestamp	xs:dateTime
binary<=255	xs:base64Binary
varbinary<=255	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
tinyint	xs:byte

12.3 Technical Data

This section contains useful background information on the technical aspects of your software. It is organized into the following sections:

- OS and Memory Requirements
- Altova XML Parser
- Altova XSLT and XQuery Engines
- Unicode Support
- Internet Usage

OS and Memory Requirements

Operating System
This software application is a 32-bit Windows application that runs on Windows 2000 and Windows XP.

Memory
Since the software is written in C++ it does not require the overhead of a Java Runtime Environment and typically requires less memory than comparable Java-based applications. However, each document is loaded fully into memory so as to parse it completely and to improve viewing and editing speed. The memory requirement increases with the size of the document.

Memory requirements are also influenced by the unlimited Undo history. When repeatedly cutting and pasting large selections in large documents, available memory can rapidly be depleted.

Altova XML Parser

When opening any XML document, the application uses its built-in validating parser (the Altova XML Parser) to check for well-formedness, validate the document against a schema (if specified), and build trees and Infosets. The Altova XML Parser is also used to provide intelligent editing help while you edit documents and to dynamically display any validation error that may occur.

The built-in Altova XML Parser implements the Final Recommendation of the W3C's XML Schema specification. New developments recommended by the W3C's XML Schema Working Group are continuously being incorporated in the Altova Parser, so that Altova products give you a state-of-the-art development environment.

Altova XSLT and XQuery Engines

Altova products use the Altova XSLT 1.0 Engine, Altova XSLT 2.0 Engine, and Altova XQuery 1.0 Engines. Documentation about implementation-specific behavior for each engine is in the section Engine Information, in Appendix 1 of the product documentation, should that engine be used in the product.

These three engines are also available in the AltovaXML package, which can be downloaded from the Altova website free of charge. Documentation for using the engines is available with the AltovaXML package.

Unicode Support

Unicode is the new 16-bit character-set standard defined by the Unicode Consortium that provides a unique number for every character,

- no matter what the platform,
- no matter what the program,
- no matter what the language.

Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. No single encoding could contain enough characters: for example, the European Union alone requires several different encodings to cover all its languages. Even for a single language like English, no single encoding was adequate for all the letters, punctuation, and technical symbols in common use.

These encoding systems used to conflict with one another. That is, two encodings used the same number for two different characters, or different numbers for the same character. Any given computer (especially servers) needs to support many different encodings; yet whenever data is passed between different encodings or platforms, that data always runs the risk of corruption.

Unicode is changing all that!
Unicode provides a unique number for every character, no matter what the platform, no matter what the program, and no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Base and many others.

Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, etc., and is the official way to implement ISO/IEC 10646. It is supported in many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends.

Incorporating Unicode into client-server or multi-tiered applications and web sites offers significant cost savings over the use of legacy character sets. Unicode enables a single software product or a single web site to be targeted across multiple platforms, languages and countries without re-engineering. It allows data to be transported through many different systems without corruption.

Windows 2000 and Windows XP

Altova's XML products provide full Unicode support. To edit an XML document, you will also need a font that supports the Unicode characters being used by that document.

Please note that most fonts only contain a very specific subset of the entire Unicode range and are therefore typically targeted at the corresponding writing system. Consequently you may encounter XML documents that contain "unprintable" characters, because the font you have selected does not contain the required glyphs. Therefore it can sometimes be very useful to have a font that covers the entire Unicode range - especially when editing XML documents from all over the world.

The most universal font we have encountered is a typeface called Arial Unicode MS that has been created by Agfa Monotype for Microsoft. This font contains over 50,000 glyphs and covers the entire set of characters specified by the Unicode 2.1 standard. It needs 23MB and is included with Microsoft Office 2000.

We highly recommend that you install this font on your system and use it with the application if you are often editing documents in different writing systems. This font is not installed with the "Typical" setting of the Microsoft Office setup program, but you can choose the Custom Setup option to install this font.

In the `/Examples` folder in your application folder you will also find a new XHTML file called `Unicode-UTF8.html` that contains the sentence "When the world wants to talk, it speaks Unicode" in many different languages ("Wenn die Welt miteinander spricht, spricht sie Unicode") and writing-systems (世界的に話すなら、Unicode です。) - this line has been adopted from the 10th Unicode conference in 1997 and is a beautiful illustration of the importance of Unicode for the XML standard. Opening this file will give you a quick impression on what is possible with Unicode and what writing systems are supported by the fonts available on your PC installation.

Right-to-Left Writing Systems

Please note that even under Windows NT 4.0 any text from a right-to-left writing-system (such as Hebrew or Arabic) is not rendered correctly except in those countries that actually use right-to-left writing-systems. This is due to the fact that only the Hebrew and Arabic versions of Windows NT contains support for rendering and editing right-to-left text on the operating system layer.

Internet Usage

Altova applications will initiate Internet connections on your behalf in the following situations:

- If you click the "Request evaluation key-code" in the Registration dialog (**Help | Software Activation**), the three fields in the registration dialog box are transferred to our web server by means of a regular http (port 80) connection and the free evaluation key-code is sent back to the customer via regular SMTP e-mail.
- If you use the Open URL... dialog box to open a document directly from a URL (**File | Open URL**), that document is retrieved through a http (port 80) connection. (*This functionality is available in XMLSpy and Authentic Desktop.*)
- If you open an XML document that refers to an XML Schema or DTD and the document is specified through a URL, it is also retrieved through a http (port 80) connection once you validate the XML document. This may also happen automatically upon opening a document if you have instructed the application to automatically validate files upon opening in the File tab of the Options dialog (**Tools | Options**). (*This functionality is available in XMLSpy and Authentic Desktop.*)
- If you are using the Send by Mail... command (**File | Send by Mail**) in XMLSpy, the current selection or file is sent by means of any MAPI-compliant mail program installed on the user's PC.

12.4 License Information

This section contains:

- Information about the distribution of this software product
- Information about the copyrights related to this software product
- The End User License Agreement governing the use of this software product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

Electronic Software Distribution

This product is available through electronic software distribution, a distribution method that provides the following unique benefits:

- You can evaluate the software free-of-charge before making a purchasing decision.
- Once you decide to buy the software, you can place your order online at the Altova website and immediately get a fully licensed product within minutes.
- When you place an online order, you always get the latest version of our software.
- The product package includes a comprehensive integrated onscreen help system. The latest version of the user manual is available at www.altova.com (i) in HTML format for online browsing, and (ii) in PDF format for download (and to print if you prefer to have the documentation on paper).

30-day evaluation period
After downloading this product, you can evaluate it for a period of up to 30 days free of charge. About 20 days into this evaluation period, the software will start to remind you that it has not yet been licensed. The reminder message will be displayed once each time you start the application. If you would like to continue using the program after the 30-day evaluation period, you have to purchase an End User License Agreement, which is delivered in the form of a key-code that you enter into the Software Activation dialog to unlock the product. You can purchase your license at the online shop at the Altova website.

Distributing the product
If you wish to share the product with others, please make sure that you distribute only the installation program, which is a convenient package that will install the application together with all sample files and the onscreen help. Any person that receives the product from you is also automatically entitled to a 30-day evaluation period. After the expiration of this period, any other user must also purchase a license in order to be able to continue using the product.

For further details, please refer to the End User License Agreement at the end of this section.

License Metering

Your Altova product has a built-in license metering module that helps you avoid any unintentional violation of the End User License Agreement. Your product is licensed either as a single-user or multi-user installation, and the license-metering module makes sure that no more than the licensed number of users use the application concurrently.

This license-metering technology uses your local area network (LAN) to communicate between instances of the application running on different computers.

Single license

When the application starts up, it sends a short broadcast datagram to find any other instance of the product running on another computer in the same network segment. If it doesn't get any response, it will open a port for listening to other instances of the application. Other than that, it will not attempt to communicate over a network. If you are not connected to a LAN, or are using dial-up connections to connect to the Internet, the application will not generate any network traffic at all.

Multi license

If more than one instance of the application is used within the same LAN, these instances will briefly communicate with each other on startup. These instances exchange key-codes in order to ensure that the number of concurrent licenses purchased is not accidentally violated. This is the same kind of license metering technology that is common in the Unix world and with a number of database development tools. It allows Altova customers to purchase reasonably-priced concurrent-use multi-user licenses.

Please note that your Altova product at no time attempts to send any information out of your LAN or over the Internet. We have also designed the applications so that they send few and small network packets so as to not put a burden on your network. The TCP/IP ports (2799) used by your Altova product are officially registered with the IANA (*see* http://www.isi.edu/in-notes/iana/assignments/port-numbers *for details*) and our license-metering module is tested and proven technology.

If you are using a firewall, you may notice communications on port 2799 between the computers that are running Altova products. You are, of course, free to block such traffic between different groups in your organization, as long as you can ensure by other means, that your license agreement is not violated.

You will also notice that, if you are online, your Altova product contains many useful functions; these are unrelated to the license-metering technology.

Copyright

All title and copyrights in this software product (including but not limited to images, photographs, animations, video, audio, music, text, and applets incorporated in the product), in the accompanying printed materials, and in any copies of these printed materials are owned by Altova GmbH or the respective supplier. This software product is protected by copyright laws and international treaty provisions.

- This software product ©1998-2007 Altova GmbH. All rights reserved.
- The Sentry Spelling-Checker Engine © 2000 Wintertree Software Inc.
- STLport © 1999, 2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, © 1996, 1997 Silicon Graphics Computer Systems, Inc, © 1997 Moscow Center for SPARC Technology.
- Scintilla © 1998–2002 Neil Hodgson `<neilh@scintilla. org>`.
- "ANTLR Copyright © 1989-2005 by Terence Parr (www.antlr.org)"

All other names or trademarks are the property of their respective owners.

Altova End User License Agreement

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 ("Software License 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 Software License 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, whether or not you decide to purchase the Software. 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. Please go to our Web site at http://www.altova.com/eula to download and print a copy of this Software License Agreement for your files and http://www.altova.com/privacy to review the privacy policy.

1. SOFTWARE LICENSE

(a) **License Grant.** Upon your acceptance of this Software License Agreement Altova grants you a non-exclusive, non-transferable (except as provided below), limited license to install and use a copy of the Software on your compatible computer, up to the Permitted Number of computers. The Permitted Number of computers shall be delineated at such time as you elect to purchase the Software. During the evaluation period, hereinafter defined, only a single user may install and use the software on one computer. If you have licensed the Software as part of a suite of Altova software products (collectively, the "Suite") and have not installed each product individually, then the Software License Agreement governs your use of all of the software included in the Suite. If you have licensed SchemaAgent, then the terms and conditions of this Software License Agreement apply to your use of the SchemaAgent server software ("SchemaAgent Server") included therein, as applicable and you are licensed to use SchemaAgent Server solely in connection with your use of Altova Software and solely for the purposes described in the accompanying documentation. In addition, if you have licensed XMLSpy Enterprise Edition or MapForce Enterprise Edition, or UModel, your license to install and use a copy of the Software as provided herein permits you to generate source code based on (i) Altova Library modules that are included in the Software (such generated code hereinafter referred to as the "Restricted Source Code") and (ii) schemas or mappings that you create or provide (such code as may be generated from your schema or mapping source materials hereinafter referred to as the "Unrestricted Source Code"). In addition to the rights granted herein, Altova grants you a non-exclusive, non-transferable, limited license to compile into executable form the complete generated code comprised of the combination of the Restricted Source Code and the Unrestricted Source Code, and to use, copy, distribute or license that executable. You may not distribute or redistribute, sublicense, sell, or transfer to a third party the Restricted Source Code, unless said third party already has a license to the Restricted Source Code through their separate license agreement with Altova or other agreement with Altova. Altova reserves all other rights in and to the Software. With respect to the feature(s) of

UModel that permit reverse-engineering of your own source code or other source code that you have lawfully obtained, such use by you does not constitute a violation of this Agreement. Except as otherwise permitted in Section 1(h) reverse engineering of the Software is strictly prohibited as further detailed therein.

(b) **Server Use.** You may install one copy of the Software on your computer file server for the purpose of downloading and installing the Software onto other computers within your internal network up to the Permitted Number of computers. If you have licensed SchemaAgent, then you may install SchemaAgent Server on any server computer or workstation and use it in connection with your Software. No other network use is permitted, including without limitation using the 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 the Software through a valid license from Altova. If you have purchased Concurrent User Licenses as defined in Section 1(c) you may install a copy of the Software on a terminal server 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 session from another computer on the network provided that the total number of user that access or use the Software on such network or terminal server does not exceed the Permitted Number. Altova makes no warranties or representations about the performance of Altova software in a terminal server environment and the foregoing are expressly excluded from the limited warranty in Section 5 hereof and technical support is not available with respect to issues arising from use in such an environment.

(c) **Concurrent Use**. If you have licensed a "Concurrent-User" version of the Software, you may install the Software on any compatible computers, 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. The Permitted Number of concurrent users shall be delineated at such time as you elect to purchase the Software licenses.

(d) **Backup and Archival Copies.** You may make one backup and one 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 under Section 3.

(e) **Home Use.** You, as the primary user of the computer on which the Software is installed, may also install the Software on one of your home computers for your use. However, the Software may not be used on your home computer at the same time as the Software is being used on the primary computer.

(f) **Key Codes, Upgrades and Updates.** 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 Software 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. If the Software that you have licensed is an upgrade or an update, then the update replaces all or part of the Software previously licensed. The update or upgrade and the associated license keys does not constitute the granting of a second license to the Software in that you may not use the upgrade or update in addition to the Software that it is replacing. You agree that use of the upgrade of update terminates your license to use the Software or portion thereof replaced.

(g) **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 granted to you in this Software License Agreement. As between you and Altova, documents, files, stylesheets, generated program code (including the Unrestricted Source Code) and schemas that are authored or created by you via your utilization of the Software, in accordance with its Documentation and the terms of this Software License Agreement, are your property.

(h) **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. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department.

(i) **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 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 Software License 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 Software License Agreement. 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 Software License Agreement and to ensure their compliance with these restrictions. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR THE ACCURACY AND ADEQUACY OF THE SOFTWARE FOR YOUR INTENDED USE AND YOU WILL INDEMNIFY AND HOLD HARMLESS ALTOVA FROM ANY 3RD PARTY SUIT TO THE EXTENT BASED UPON THE ACCURACY AND ADEQUACY OF THE SOFTWARE IN YOUR USE. WITHOUT LIMITATION, THE SOFTWARE IS NOT INTENDED FOR USE IN THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, COMMUNICATION SYSTEMS OR AIR TRAFFIC CONTROL EQUIPMENT, WHERE THE FAILURE OF THE SOFTWARE COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE.

2. INTELLECTUAL PROPERTY RIGHTS

Acknowledgement of Altova's 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. XMLSpy, Authentic, StyleVision, MapForce, Markup Your Mind, Axad, Nanonull, and Altova are trademarks of Altova GmbH (registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows 95, Windows 98, Windows NT, Windows 2000 and Windows XP 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 Software License 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 each of this Software License 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 Software License 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 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 Software License 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. 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 Software License 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 Software License 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 SOFTWARE LICENSE 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 Software License 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 with 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 Software License 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 infringements that would not be such, except for customer-supplied elements.

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 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. 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 Software License Agreement before installation. Updates of the operating system and application software not specifically covered by this Software License Agreement are your responsibility and will not be provided by Altova under this Software License 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 Software License 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**. Altova has a built-in license metering module that helps you to avoid any unintentional violation of this Software License Agreement. Altova may use your internal network for license metering between installed versions of the Software.

(b) **Software Activation**. **Altova's Software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration, use, or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova license server. You agree that Altova may use these measures and you agree to follow any applicable requirements.**

(c) **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.

(d) **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 Software License Agreement. By your acceptance of the terms of this Software License Agreement or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Software License Agreement and/or the Privacy Policy as revised from time to time. European users understand and consent to the processing of personal information in the United States for the purposes described herein. Altova has the right in its sole discretion to amend this provision of the Software License 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.

8. TERM AND TERMINATION

This Software License Agreement may be terminated (a) by your giving Altova written notice of termination; or (b) by Altova, at its option, giving you written notice of termination if you commit a breach of this Software License 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.. In addition the Software License Agreement governing your use of a previous version that you have upgraded or updated of the Software is terminated upon your acceptance of the terms and conditions of the Software License Agreement accompanying such upgrade or update. Upon any termination of the Software License Agreement, you must cease all use of the Software that it 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(g), (h), (i), 2, 5(b), (c), 9, 10 and 11 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 Software License 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. 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 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.

11. GENERAL PROVISIONS

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 Software License 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 Software License 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 Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts 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 Software License 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 Software License 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.

This Software License 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 Software License 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 Software License 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 Software License Agreement. This Software License 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 Software License 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 Software License Agreement. If, for any reason, any provision of this Software License Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Software License Agreement, and this Software License 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: 2006-09-05

Index