# HP SOA Systinet Assertion Editor

Software Version: 2.50

## User Guide

Document Release Date: May 2007
Software Release Date: May 2007

# Legal Notices

*Warranty*

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

*Restricted Rights Legend*

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

*Third-Party Web Sites*

Mercury provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. Mercury makes no representations or warranties whatsoever as to site content or availability.

*Copyright Notices*



*Trademark Notices*

Java™ is a US trademark of Sun Microsystems, Inc. Microsoft®, Windows® and Windows XP® are U.S. registered trademarks of Microsoft Corporation. IBM®, AIX® and WebSphere® are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries. BEA® and WebLogic® are registered trademarks of BEA Systems, Inc.

# Contents

# Welcome to This Guide

Welcome to HP SOA Systinet Assertion Editor, a tool for use with the HP SOA Systinet Policy Manager component of SOA Systinet. Assertion Editor enables you to create, edit and delete assertions on any number of HP SOA Systinet Policy Manager servers. In addition, you can use Assertion Editor to test an assertion, running a local validation of the assertion against a source document.

## Concepts

Assertions are the atomic level of policy. In HP SOA Systinet Policy Manager, one or more assertions are collected together to form a WS-Policy document called a *technical policy*. The technical policy is a set of assertions that fulfils a management requirement. Technical policies in turn are associated with specific artifacts or artifact types to form a WS-PolicyAttachments document called a *business policy*. This is the top level of policy, embodying specific business requirements. SOA Systinet provides tools for testing whether sources comply with the relevant business policies.

To meet management requirements, a technical policy sometimes needs a new assertion. Changing requirements can also result in existing assertions becoming out-of-date. Assertion Editor is a tool built on the widely used Eclipse IDE to ease assertion creation and editing.

## How This Book Is Organized

This book is organized into the following chapters:

Introduction to the User Interface on page 12. A tour of the UI.

Getting Started on page 11. Getting started with Assertion Editor, including creating a project and following workflow.

Manipulating Assertions on page 25. Creating, downloading, editing and publishing assertions.

Testing Assertions on page 47. Running local validations.

Creating Custom Assertions on page 51. Creating and modifying custom source types and validation handlers.

## Document Conventions

The typographic conventions used in this document are:

| | |
|---|---|
| **run.bat make** | Script name or other executable command plus mandatory arguments. |
| [--help] | A command-line option. |
| either \| or | A choice of arguments. |
| *replace_value* | A command-line argument that should be replaced with an actual value. |
| {arg1 \| arg2} | A choice between two command-line arguments where one or the other is mandatory. |
| `rmdir /S /Q System32` | Operating system commands and other user input that you can type on the command line and press **Enter** to invoke. Items in *italics* should be replaced by actual values. |
| `C:\System.ini` | Filenames, directory names, paths and package names. |
| `a.append(b);` | Program source code. |
| `server.Version` | An inline Java or C++ class name. |
| `getVersion()` | An inline Java method name. |
| **Shift-N** | A combination of keystrokes. |
| **Service View** | A label, word or phrase in a GUI window, often clickable. |
| **New->Service** | Menu choice. |

## Documentation Updates

This manual's title page contains the following identifying information:

- Software version number
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

**http://ovweb.external.hp.com/lpe/doc_serv/**

# Support

## Mercury Product Support

You can obtain support information for products formerly produced by Mercury as follows:

- If you work with an HP Software Services Integrator (SVI) partner (www.hp.-com/managementsoftware/svi_partner_list), contact your SVI agent.
- If you have an active HP Software support contract, visit the HP Software Support Web site and use the Self-Solve Knowledge Search to find answers to technical questions.
- For the latest information about support processes and tools available for products formerly produced by Mercury, we encourage you to visit the Mercury Customer Support Web site at: http://support.-mercury.com.
- For the latest information about support processes and tools available for products formerly produced by Systinet, we encourage you to visit the Systinet Online Support Web site at: **http://www.systinet.-com/support/index**.
- If you have additional questions, contact your HP Sales Representative.

## HP Software Support

You can visit the HP Software Support Web site at:

www.hp.com/managementsoftware/services

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts

- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To find more information about access levels, go to: www.hp.-com/managementsoftware/access_level

To register for an HP Passport ID, go to: www.managementsoftware.hp.com/passport-registration.html

# 1 Getting Started

This chapter shows what you need to do before working with assertions in HP SOA Systinet Assertion Editor. It contains the following sections:

Launching Assertion Editor on page 11. The first time Assertion Editor is started, the data directory needs to be set. Also you should be in the Assertion Editor perspective.

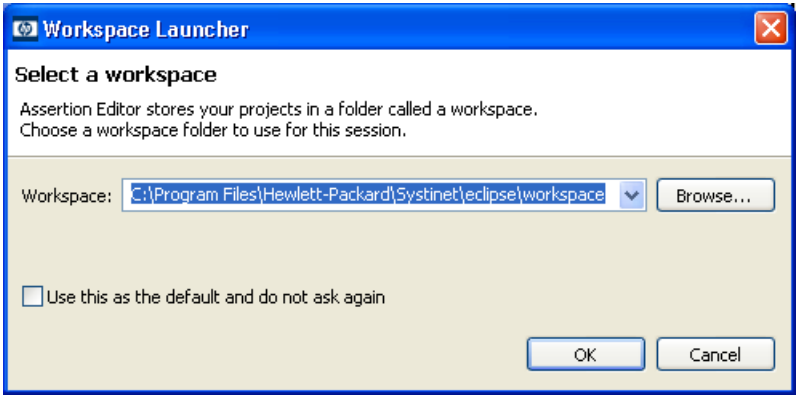
Creating an Assertion Project on page 20. Set up an Assertion Project.

## Launching Assertion Editor

HP SOA Systinet Assertion Editor is launched from the `start.exe` file in the `assertion-editor` installation directory or from a shortcut to this file.

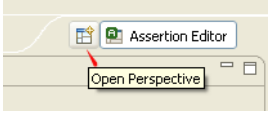
When you start Eclipse, select the directory in which to save your work. The default directory is `../assertion-editor/workspace`, as shown in Figure 1. If you want all future work to be stored in the same location, select **Use this as default and do not ask again**.

### Figure 1. Workspace Launcher



Assertion Editor automatically opens in the Assertion Editor perspective. The open perspective is indicated in a field just above the top right corner of the main pane, as in Figure 2. If Assertion Editor is not in the Assertion Editor perspective, click the **Open Perspective** icon next to the field and select **Assertion Editor**. The Assertion Editor perspective may also be selected from the **Window**+**Open Perspective**->**Other** menu.

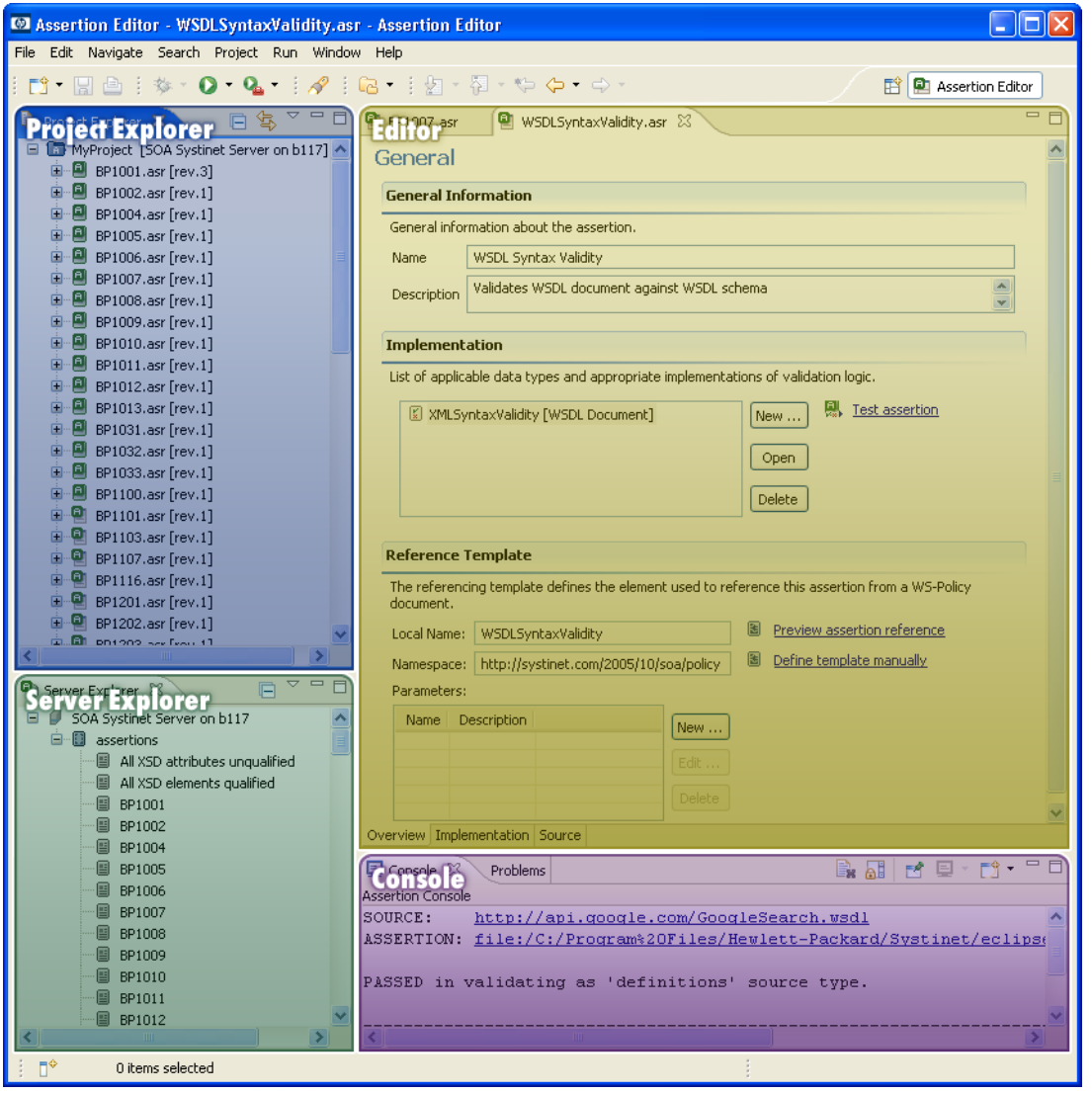
### Figure 2. Perspective




## Introduction to the User Interface

The user interface is split into four panes with menu options across the top, as shown in Figure 3. The upper left pane is the **Project Explorer**, a tree view of assertion projects and assertions. The pane beneath it, the **Server Explorer**, gives a tree view of the SOA Systinet servers on which the assertions reside. The main pane, the **Editor**, contains the details of the item highlighted on the left. Multiple tabs can be open in the **Editor** and the functionality varies depending on the tab. The bottom-right pane is the **Console** view, showing the results of assertion tests.

**Figure 3. Assertion Editor Dashboard Showing Panes**



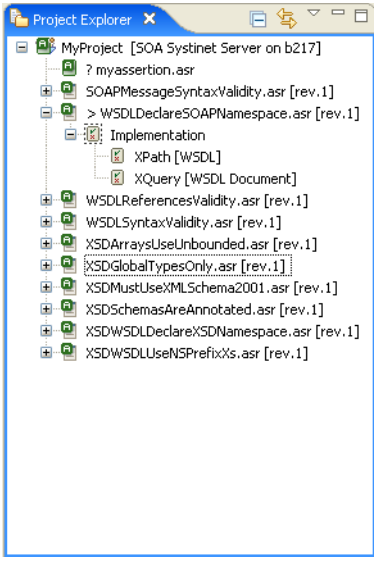
The user interface panes are described in the following sections:

- Project Explorer on page 14
- Server Explorer on page 16
- Editor on page 18
- Console on page 18

## Project Explorer

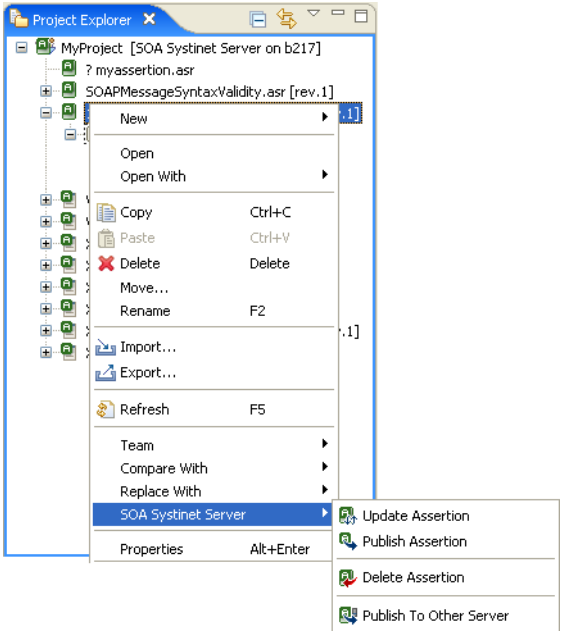
The **Project Explorer** is the upper left pane of the Assertion Editor dashboard (see Figure 3). This is a hierarchical list of projects, the assertions in each project and the validation definitions in each assertion. Project titles are followed by their host server names in brackets. Assertion names are followed by revision number in brackets. Assertions that have not been published are indicated by a question mark, **?**. Assertions that have been changed locally since they were last synchronized with the version on the server are indicated by a right arrow, >. Figure 4 shows examples of unpublished and unsynchronized assertions.

### Figure 4. Project Explorer View



Right-clicking a project, assertion or implementation opens its context menu, shown in Figure 5.

## Figure 5. Project Explorer Context Menu




The context menus of these items are broadly similar. They include, where relevant, standard **Open**, **Copy**, **Paste**, **Delete**, **Move** and **Rename** options. They also include the advanced options detailed in Table 1 on page 16.

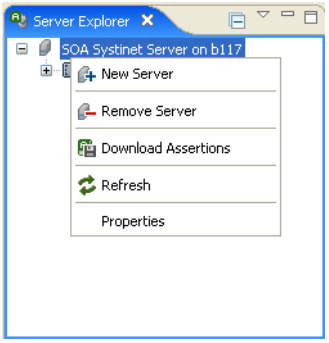
**Table 1. Project Explorer Context Menu Options**

| Option | Description |
|---|---|
| New | Open new project, file or folder. Or choose **Other** and start a wizard for creating a new assertion. |
| Open With | Instead of opening the assertion or implementation in the UI editor, you can open it in the SOA Systinet XML editor or a number of other editors |
| Import and Export | Import a project from or export a project to a file or Team Project Set. |
| Refresh | Reflect changes made outside the workspace. The local version does not change. This is the standard Eclipse **Refresh** action. |
| Team | You can Apply a Patch from a team project, or share your project with a team. |
| Compare with... | Compare the current version of the assertion with the history of its local changes. |
| Replace with... | Replace the content with local history. |
| SOA Systinet server | Update the assertion with the version on the server; publish the assertion to an SOA Systinet server; unpublish the assertion from the server. Please see Publishing Assertions on page 43. |
| Properties | General properties of the project, assertion or implementation. Here you can set 'Read-Only' status. |

## Server Explorer

The **Server Explorer** is the bottom left pane of the Assertion Editor dashboard (see Figure 3). It lists all the servers associated with all the Assertion Editor projects, as shown in Figure 6.

## Figure 6. Server Explorer View




Expand a server node to open a list of all the assertions published to that server. Right-click a server or assertion in the **Server Explorer** to open a similar context menu, with the options described in Table 2 on page 17.

## Table 2. Server Explorer Context Menu Options

| Option | Function |
|---|---|
| New Server | Add a server for downloading assertions. |
| Remove Server | Delete server from **Server Explorer**. |
| Download Assertions | Download assertions from a server. For more information, see Downloading Assertions on page 28. |
| Delete Assertion | Purge the assertion from the server (Assertion context menu only). For more information, see Deleting an Assertion From the Server on page 44. |
| Refresh | Refresh server's list of published assertions. |
| Properties | View and edit server name, URL, username and password (Server context menu only). |

## Editor

The **Editor** is the upper right pane of the Assertion Editor dashboard (see Figure 3). It has three tabs, described in the following sections:


- Overview Tab on page 18
- Implementation Tab on page 18
- Source Tab on page 18


### Overview Tab

The **Overview** tab shows the components of the assertion. It is divided into the following areas:

- **General Information.** The name of the assertion and its description.
- **Implementation.** This is a list of implementations of validation logic and the artifact types to which they apply. Implementations can be added or deleted here; please see Adding and Deleting Implementations on page 33 for procedures. Validation definition details are shown in the **Validations** tab.
- **Reference Template.** The referencing template defines the element used to reference this assertion from a WS-Policy document. It may include parameters, which represent requirements whose specific values might vary.

### Implementation Tab

The **Implementation** tab includes a list of implementations. Highlighting a implementation opens the appropriate editor in the window beneath.

### Source Tab

The **Source** tab is an XML editor for editing the assertion as a raw source document.

## Console

The **Console** is the lower right pane of the Assertion Editor dashboard (see Figure 3). It displays all console messages, most importantly the results of local validations.

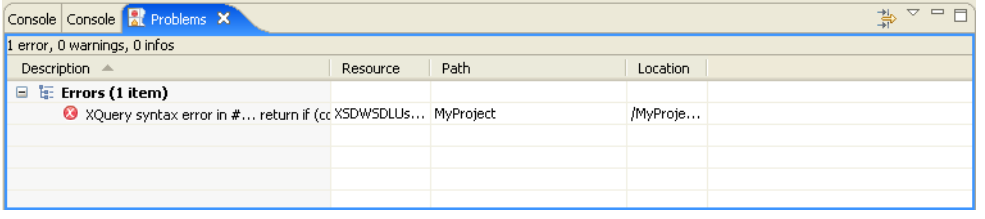
Across the top of the console pane are a number of useful display buttons, described in Table 3 on page 19.

### Table 3. Console Display Buttons

| Icon | Use |
|---|---|
| Clear Console  | Erases currently open console. |
| Scroll lock  | Allows long messages to be read in parts. |
| Pin console  | Does not allow new consoles to replace one currently being viewed. |
| Display selected console | This icon is inherited from Eclipse and is not used in Assertion Editor. |
| New console  | Creates a new console window. |
| Minimize/Restore | Reduce the console to a small strip showing only the tabs for each console and the icons, and restore it to full size. The Maximize icon only restores the console size in Assertion Editor |
| Maximize | |

A **Problems** tab can also open in the **Console**, as shown in Figure 7. This tab lists all errors with missing servers, XQuery syntax, etc. To open the **Problems** tab, open the **Window** Eclipse menu and select **Window->Show View->Problems**. The **Problems** tab does not have the console display buttons, although it does have the standard Eclipse filter button.

### Figure 7. Problems Tab

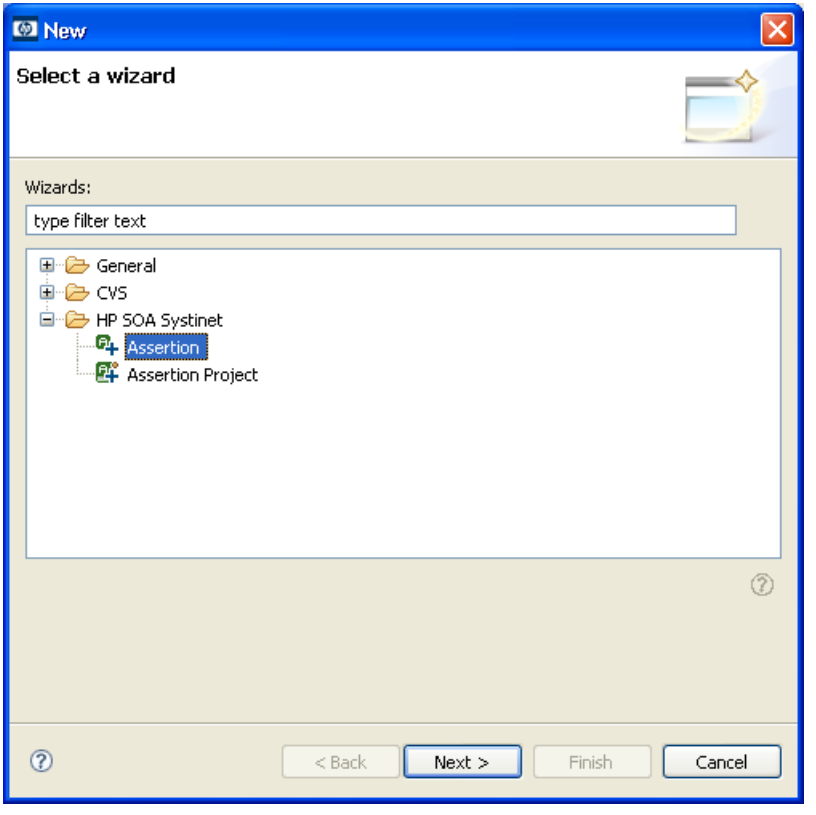


## Creating an Assertion Project

To work with assertions, you need an Assertion Project. You can create any number of Assertion Projects to help organize your work. To create an Assertion Project:

1 Open the **New Assertion Project** wizard in one of these ways:

- Click the **New** icon. The **New: Select a Wizard** window opens. Select **HP SOA Systinet->Assertion Project**.




- Alternatively, from the **File** menu, select **File+New-> Assertion Project**.

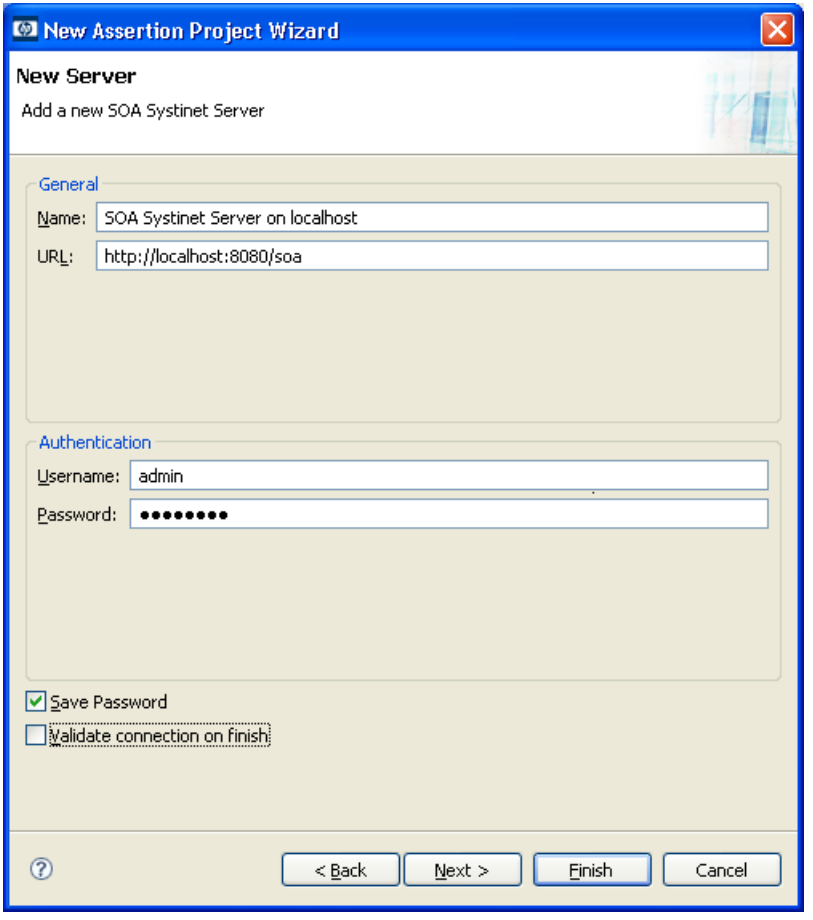
- Alternatively, press **Alt**-**Shift**-**N** and then press **R**. The **New Project: Select a Wizard** window opens. Select **HP SOA Systinet**->**Assertion Project**.

2 Type in a name and location for the project. The default location is the workspace you chose when you launched Assertion Editor (see Launching Assertion Editor on page 11).

3 Click **Next**. If no server is defined, the **New Server** window opens and you go to Step 5. Otherwise the **Choose Server** window opens.

4 Select whether to add a new server to the project or use a server you have already defined. This step does not appear if there are no defined servers. If you select an existing server, go to Step 6.

5 Type in the name and URL of a SOA Systinet server. Assertions are downloaded from and published to this server. You can include additional servers later. Enter the username and password you use for this server. You can save the password and validate your connection to the server.




6 Select assertions to download from the server. You can filter them by text or sort them according to technical policy or source type. Additional assertions can be downloaded later.

# Workflow

The following sections provide overviews of the workflow of the main Assertion Editor tasks:



## Creating and Publishing a New Assertion

To create and publish a new assertion:

1. Create a draft assertion, as described in Creating New Assertions on page 26.
2. Add implementations to the assertion, as described in Editing Assertions on page 31.
3. Test the new assertion as described in Testing Assertions on page 47.
4. Edit the assertion, as described in Editing Assertions on page 31, if it did not perform as required.
5. Repeat Step 3 and Step 4 until the assertion performs as required.
6. Publish the assertion to a server as described in Publishing Assertions on page 43.

## Editing Assertions

To edit an existing assertion:

1. If you do not have a local copy of the assertion, download one as described in Downloading Assertions on page 28.
2. Update the local copy of the assertion then open it in the **Assertion Editor**, as described in Editing Assertions on page 31.
3. You can edit general information (see Editing General Properties on page 33), implementations including validation handlers (see Adding and Deleting Implementations on page 33, and/or reference templates including parameters (see Editing Reference Templates on page 39).
4. Test the new assertion as described in Testing Assertions on page 47.

5 Edit the assertion, as described in Editing Assertions on page 31, if it did not perform as required.

6 Repeat Step 4 and Step 5 the assertion performs as required.

7 Publish the assertion to a server as described in Publishing Assertions on page 43.

# 2 Manipulating Assertions

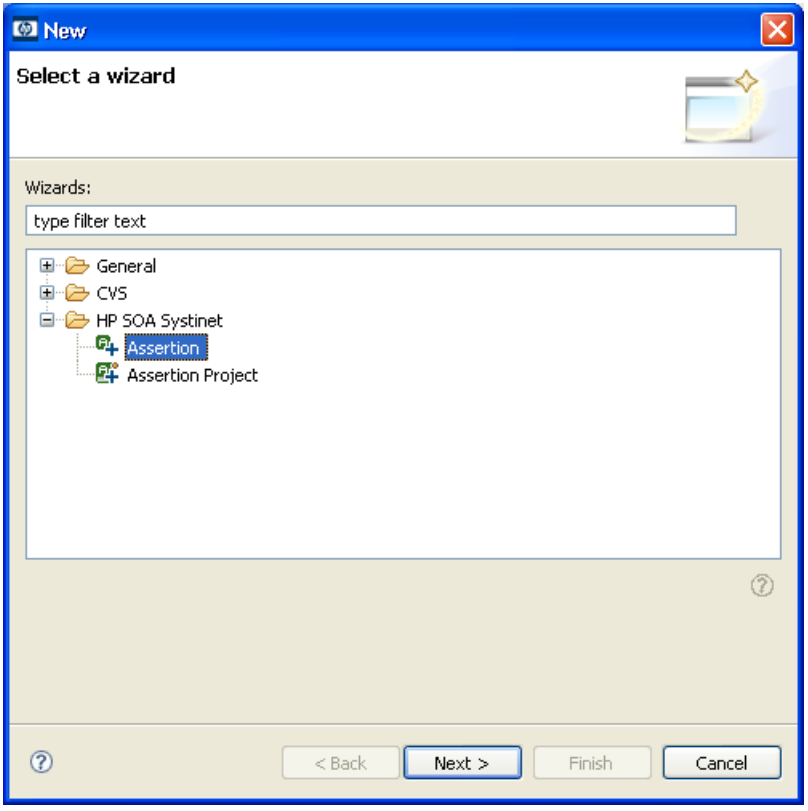
This chapter describes how to work with assertions, as detailed in the following sections:

## Creating New Assertions

To create a new assertion:

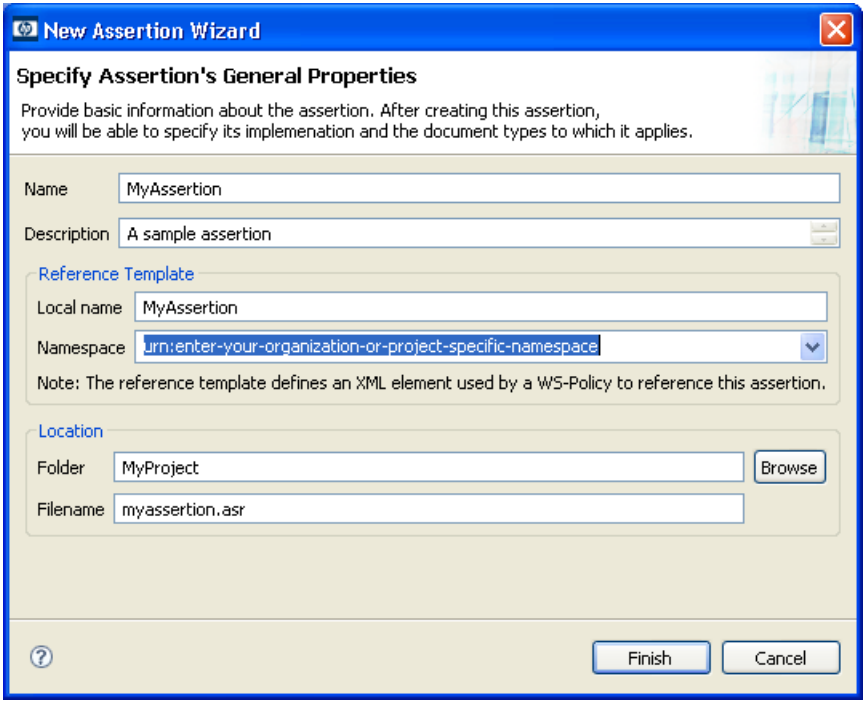
1

- Click the **New** icon. The **New: Select a Wizard** window opens. Select **HP SOA Systinet->Assertion**.




- Alternatively, from the **File** menu, select **File+New->Assertion**.
- Alternatively, press **Alt-Shift-N**. A context menu opens. Select **Assertion**.

The **New Assertion** wizard opens.

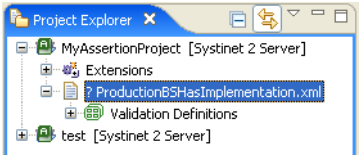



2 Fill in the fields as follows, then click **Finish**:

| Field | Value to enter |
|---|---|
| Name | Arbitrary name of assertion. Should be informative. Any characters can be used, but see Reference Template: Local name and Location: Filename. |
| Description | A text description of what the assertion is for. |
| Reference Template: Local name | The referencing template defines the element used to reference this assertion from a WS-Policy document. Arbitrary name of the XML reference template element. By default this is the same as the assertion name above, but only alphanumeric characters are allowed. Disallowed characters in the assertion name will be rendered as underscores _ by default. |

| Field | Value to enter |
|---|---|
| Reference Template: Namespace | Enter the policy namespace specific to your organization's coding guidelines, such as `http://example.com/soa/policy`. |
| Location: Folder | The Assertion Project folder in which you wish to save the assertion. |
| Location: Filename | Arbitrary name of the XML file of the assertion. Default is the same as the name of the assertion, with prohibited characters replaced by underscores. |

3 Find the newly created assertion in the **Project Explorer**. It will be listed under the project folder in which you saved it. Double-click the assertion to open it in the **Editor** pane.




4 Add an implementation as described in Adding and Deleting Implementations on page 33.

5 Run local validations to test the assertion as described in Testing Assertions on page 47.

6 Publish the assertion as described in Publishing Assertions on page 43.

## Downloading Assertions

Using Assertion Editor you can download assertions from a HP SOA Systinet Policy Manager server, then edit and test them. Assertions can be downloaded when you create a project, as described in Creating an Assertion Project on page 20. This section describes how to download them into an existing project.

If you need to download assertions from a server that is not in the project, add it to the project. To add a server:

1 Navigate to the **Server Explorer** view.

2 Right-click any item in the **Server Explorer**. A context menu appears.

3 Select **New Server**. The **New Server** wizard opens.

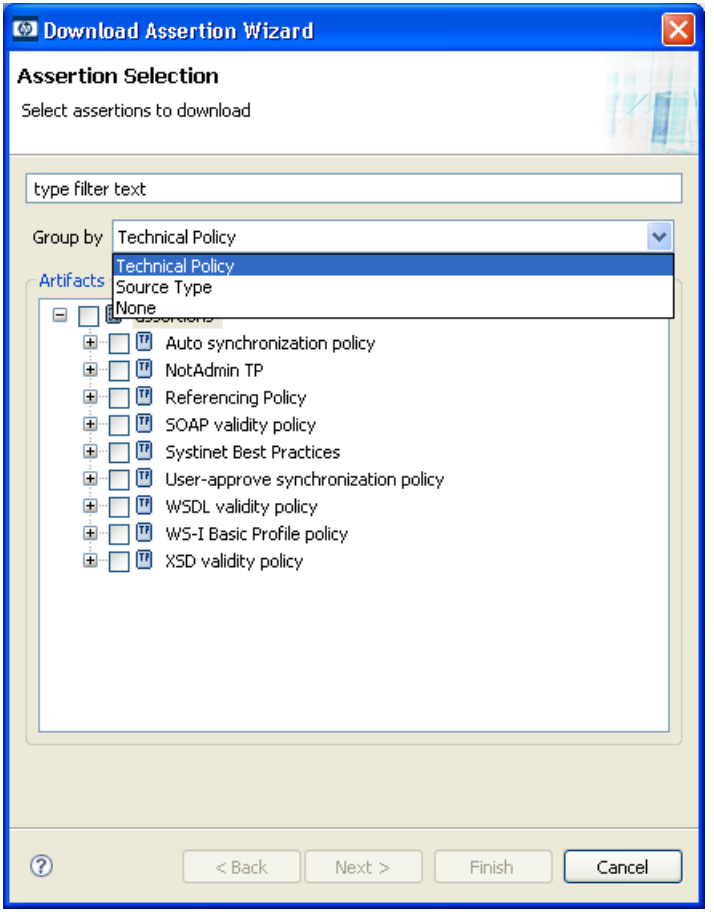
4 Type in the name and URL of a SOA Systinet server. Enter the username and password you use for this server. You can save the password and validate your connection to the server.

5 Click **Finish** to add the server.

To download assertions:

1 Right-click the server from which you want to download assertions. The context menu opens.

2 Click **Download assertions**. The **Download Assertions** wizard opens. It contains a tree view of assertions. You can display a tree organized by Technical Policy or Source type, or choose **None** to have a flat list of assertions in alphabetical order. You can also filter the assertions by text string.



3 Select assertions to download from the server. You can select individual assertions or groups of assertions according to source type or their inclusion in technical policies.

4 Click **Next**. The **Choose Location** window opens.

5 Choose the project to which you will download the assertions and click **Finish**.

## Editing Assertions

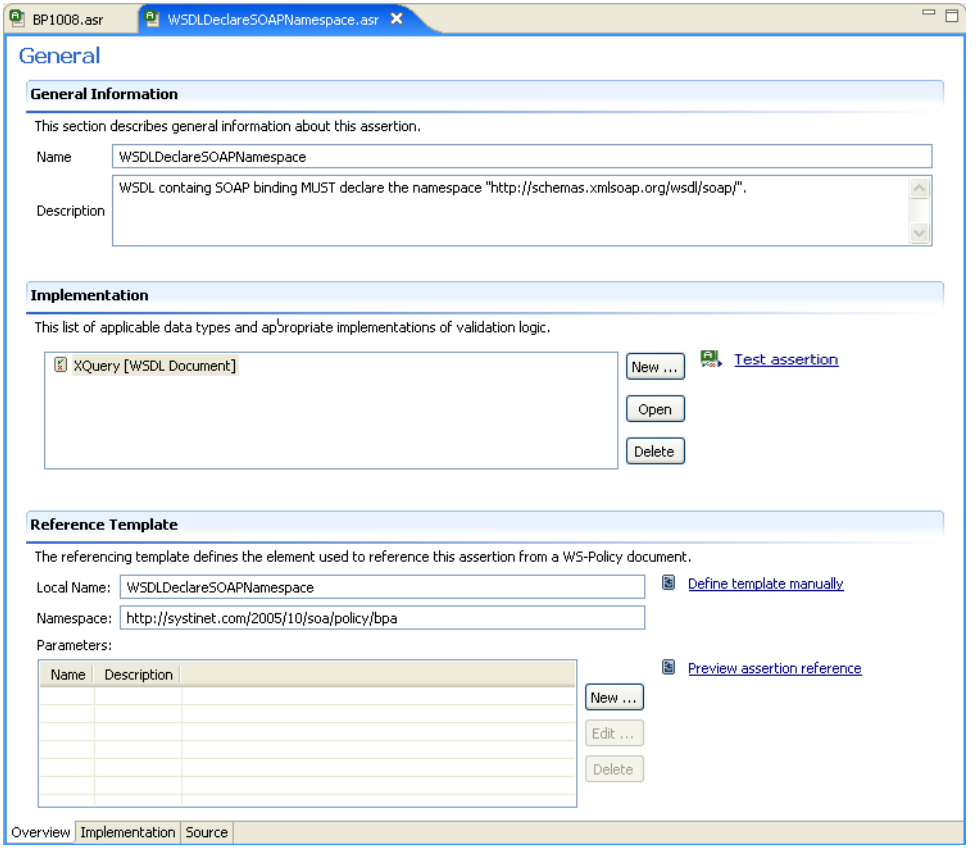
As its name suggests, the heart of Assertion Editor's functionality is the ability to edit assertions. To edit an assertion, you must have a local copy of it, which you created (see Creating New Assertions on page 26) or downloaded from a server (see Downloading Assertions on page 28).

Avoid revision conflicts. If you are editing an assertion that also exists on a server, update your local copy before editing it. To update an assertion from the server, right-click on the assertion in **Project Explorer** and select **SOA Systinet Server**->**Update Assertion** from the context menu.

If you change a local assertion before updating it from the server, there can be a revision conflict. Assertion Editor warns you if this is the case. Please see Resolving Conflicts on page 43.

To edit an assertion, open it in the **Editor**. Open the assertion by double-clicking it in the **Project Explorer**. The **Editor** opens in the **Overview** tab, as in Figure 1.

## Figure 1. Overview Tab of Editor




You can perform the following tasks in the **Editor**:

- Edit **General Information** about the assertion. (See Editing General Properties on page 33.)
- Add or delete **Implementations** and edit their XQuery or XPath definitions. (See Adding and Deleting Implementations on page 33.)

- Edit the **Reference Template**, for example by adding or deleting parameters. (See Editing Reference Templates on page 39.)

## Editing General Properties

General properties refer to the arbitrary name and text description of the assertion. Changing the name in the editor does not change the filename or reference template local name. To edit these properties:

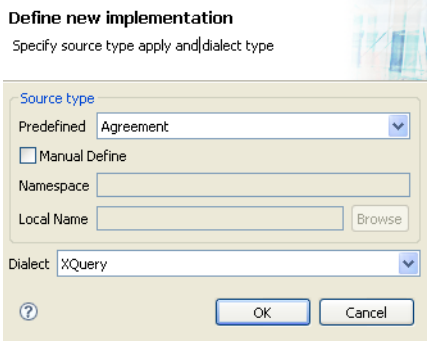
1. Locate the **General Properties** section in the **Overview** tab of the **Editor**. (See Figure 1.)
2. Change the **Name** and **Description**.
3. Click the **Save** icon or enter **Ctrl-S**.

## Adding and Deleting Implementations

A implementation contains a resource type and the code used to validate that resource type. An assertion must contain one or more implementations. See Assertion Schema in the SOA Systinet Reference Guide for more information.

To add or remove implementations, follow these steps:

1. Locate the **Implementations** section in the **Overview** tab of the **Editor** . (See Figure 1.)
2. To delete an implementation, find it in the list of implementations, highlight it and press **Delete**.
3. To add an implementation, press **New...**. The **Define New Implementation** wizard opens.



4 Fill in the fields as follows and then click **OK**. The wizard then closes and the **Implementation** tab of the **Assertion Editor** opens.

| Field | Value |
|---|---|
| Source Type: Predefined | Select from a drop-down menu of predefined artifacts to which the assertion will apply. Most source types are **Artifacts**, meaning they are abstract resources in the SOA Systinet repository consisting of SOA metadata. If a source type is labeled as a **Document**, it is the data itself to which the artifact refers and is external to the repository. |
| Source Type: Manual define | Select this if the assertion will apply to a different type of artifact than those predefined. You then provide the assertion's name and namespace. |
| Source Type: Namespace | If you are manually defining the artifact type, type in the artifact namespace. |
| Source Type: Local name | If you are manually defining the artifact type, type in the artifact name. |
| Dialect | Choose one of XQuery, XPath or Manual dialects for the validation handler. |

5 In the **Implementation** tab of the **Assertion Editor**, edit the XQuery or XPath definition so that it performs the desired task. Knowledge of XPath and XQuery dialects is needed. For instructions, see Writing XPath Definitions on page 34 or Writing XQuery Definitions on page 38.

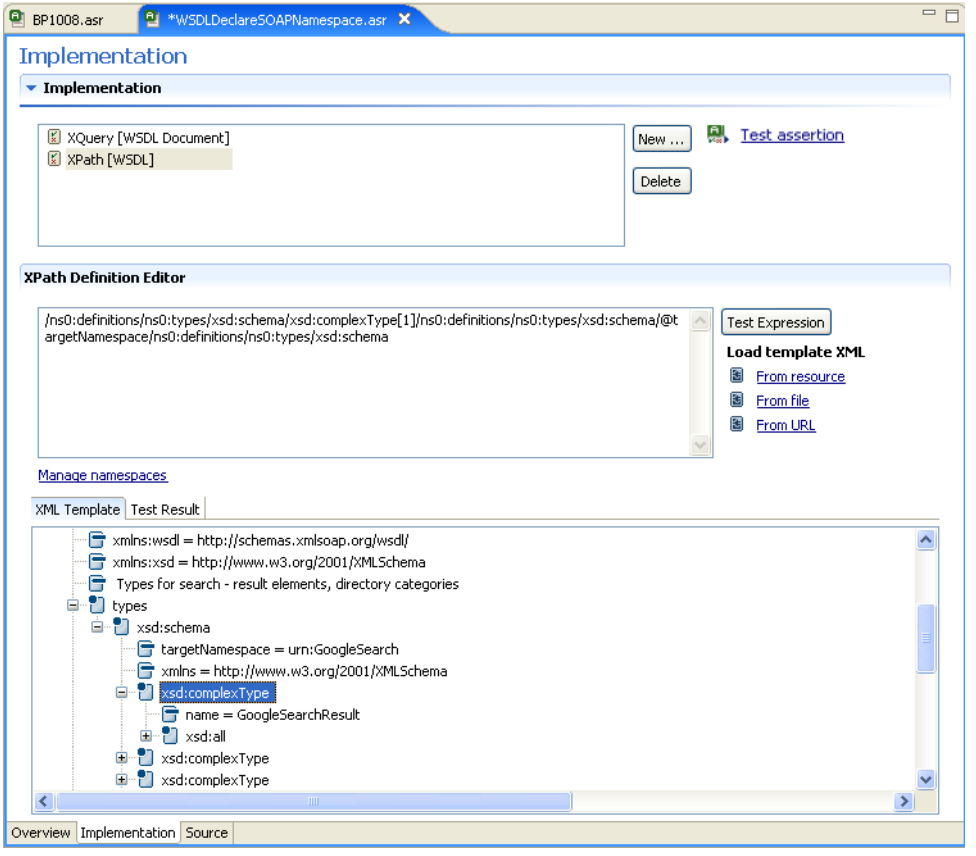
If you are editing a manual implementation, write a clear text that the end user can use to evaluate whether an artifact complies with the assertion. By default the assertion description is used as the manual implementation text.

6 To add additional implementations, repeat Step 3-Step 5.

## Writing XPath Definitions

After creating an implementation that uses an XPath validation handler, as described in Adding and Deleting Implementations on page 33, you need to write the XPath definition. Write this definition in the **Implementation** tab of the **Editor**, shown in Figure 2.

**Figure 2. Implementation Tab for XPath Definition in Editor**



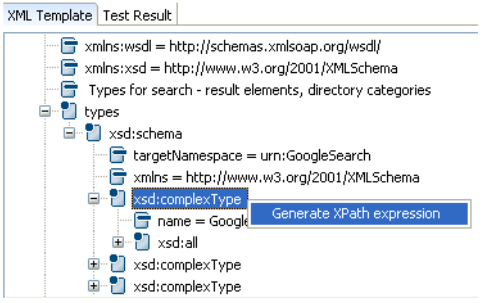
To write an XPath definition, follow these steps:

1 Import a sample XML document of the type to which the assertion applies. In the **XPath Definition Editor** (see Figure 2), there are three links under **Load XML Template**. Click one of them as follows:

- To load a sample XML document from your Assertion Editor project, click **From resource**.
- To load a sample XML document from your local file system, click **From file**.
- To load a sample XML document from the Web, click **From URL**.

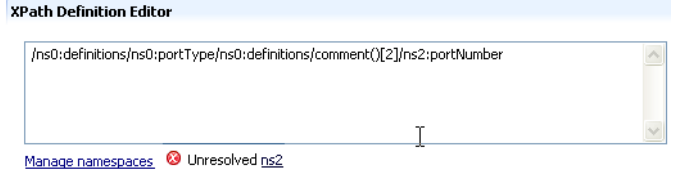
The XML document appears in the **XML Template** tab.

2 Add an XPath expression. Right-click the relevant line in the sample XML document and select **Generate XPath expression** from the context menu. The XPath expression appears in the **XPath Definition Editor** field. You can only have one XPath expression for each implementation. An artifact will pass validation if at least one XML node matches the XPath expression.



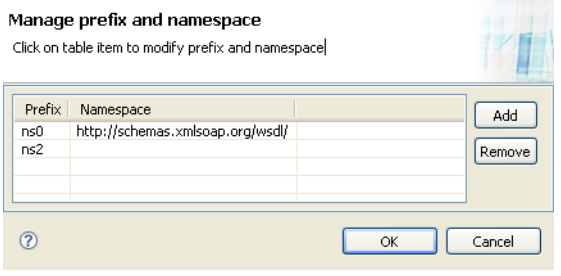

3 Modify the XPath expression in the **XPath Definition Editor**, if necessary.

4 If the XPath contains any unresolved namespace prefixes, an **Unresolved** warning appears.



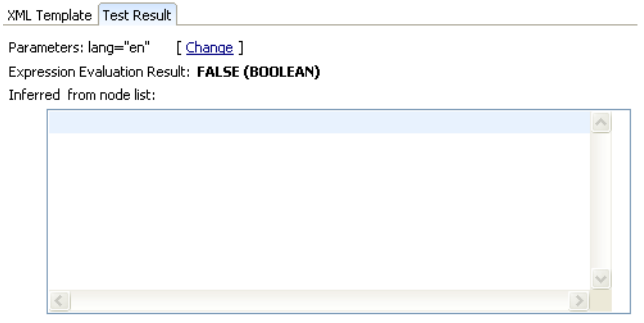
Click the unresolved prefix link next to the warning to open the **Manage prefix and namespace** pane, where you can define the prefix's namespace. See Step 5.

5 You can add, delete or change the namespaces that define the prefixes in the XPath. Click **Manage namespaces** and the **Manage prefix and namespace** wizard appears.




To add a namespace, click **Add** and type the prefix and namespace URL. To delete a namespace, select it and click **Remove**. Click **OK** when done.

6 Test the XPath expression by clicking **Test expression**. The results of the test appear in the **Test Results** tab of the **XPath Expression Editor**.



## Writing XQuery Definitions

The only native tool that Assertion Editor supplies for writing and editing XQueries is syntax highlighting, including error highlighting when saving an XQuery. However, Assertion Editor supports external XML editors for working with XQueries.

To use an external XQuery editor with Assertion Editor, add the following Saxon extension to the external editor: `pm-extension-functions.jar`, located in
`AE_HOME/plugins/com.systinet.tools.assertioneditor.lib_1.0.0/lib/saxon-extensions/`

For instructions on how to add the Saxon extension to the most popular XML editors, see these sections:


- Editing XQueries in oXygen on page 38
- Editing XQueries in Stylus Studio on page 39


### Editing XQueries in oXygen

To set up oXygen™ to edit XQueries:

1. Open or create the XQuery file in oXygen.
2. Click the **Configure Transformation Scenario** icon. The **Configure Transformation Scenario** wizard opens.
3. Select **Execute XQuery** and click **New**. The **Edit Scenario** pane opens.
4. In the **Transformer** field, select "Saxon 8B."
5. Click **Extension**. The **Extensions** pane opens.
6. Click **Add**. The **Add Extension** pane opens. Type in or browse for the path to `pm-extension-functions.jar`.
7. Click **OK** in all wizard panes to save the transformation scenario.

When you open any other XQuery files, you must always choose this transformation scenario and then edit the XQuery file to force oXygen to rebuild it.

This procedure was created for oXygen 8.1. Other versions can be used but some details may differ.

### Editing XQueries in Stylus Studio

Follow these steps to use Stylus Studio™ with Assertion Editor:

1. Select **Tools**->**Options**. The **Options** dialogue opens.
2. Navigate to **Module Setting**+**XQuery**->**Processor Settings**. In the **Processor** field, select Saxon 8.7.3 and select **Use as default processor**.
3. Navigate to **Project**->**Set Classpath** and add the path to `pm-extension-functions.jar`.

After Stylus Studio is set up, you can open an XQuery in Stylus Studio from Assertion Editor by right-clicking the XQuery in **Project Explorer** and choosing **Open With**->**System Editor** from the context menu.

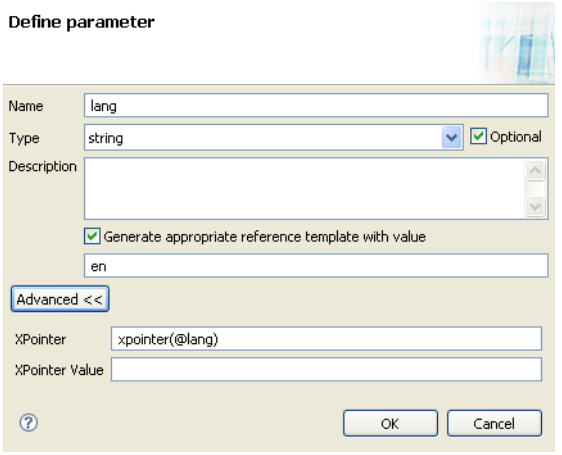
## Editing Reference Templates

The referencing template defines the element used to reference this assertion from a WS-Policy document. It can include parameters, which represent requirements whose specific values might vary. For more information see the Assertion Schema chapter in the SOA Systinet Reference Guide.

To add a custom field that Assertion Editor does not support into the reference template, click **Define template manually**. An editor opens with syntax highlighting.

To edit an assertion's reference template, including its parameters:

1. Locate the **Reference Template** section in the **Overview** tab of the Assertion Editor pane. (See Figure 1.)
2. Change the namespace if it is not correct. You can also change the local name of the reference template to any arbitrary value.
3. To add a parameter, click **New...** next to the table of parameters. To edit an existing assertion, highlight it in the table of assertions and press **Edit**. In both cases the **Define Parameter** wizard opens.




4 Fill out the wizard's fields as follows, then click **OK**:

| **Field** | **Value** |
|---|---|
| Name | Arbitrary parameter name. |
| Type | Choose the XML Schema data type that matches your parameter. Default is 'String.' |
| Default value | The value of the parameter if it is not otherwise set in a SOA Systinet technical policy. |
| Optional | Sets the parameter as optional. If an artifact violates an optional parameter during validation, an error is raised but the artifact still passes validation. If you do not select **Optional**, the parameter is mandatory. If an artifact violates a mandatory parameter, it fails during validation. |
| Description | Arbitrary text describing the parameter. |
| Generate appropriate reference template with value | Select this to set a default value for the parameter, which you type into the field beneath the check box. If you do not generate a reference template with a default value, the policy developer sets this value when including the assertion into a technical policy with HP SOA Systinet Policy Manager. For mandatory parameters, the value must be set either here or in HP SOA Systinet Policy Manager. |

| Advanced Options | |
|---|---|
| **Field** | **Value** |
| XPointer | Assertion Editor automatically generates an XPointer or XPointer/ValueXPointer pair to express the parameter. Click **Advanced** if you need to edit the XPointer and/or ValueXPointer. See Assertion Schema in the SOA Systinet Reference Guide for more details. |

5 To see what the reference template looks like when embedded into a technical policy, click **Preview assertion reference**. A dialog opens in which you set example parameter values.

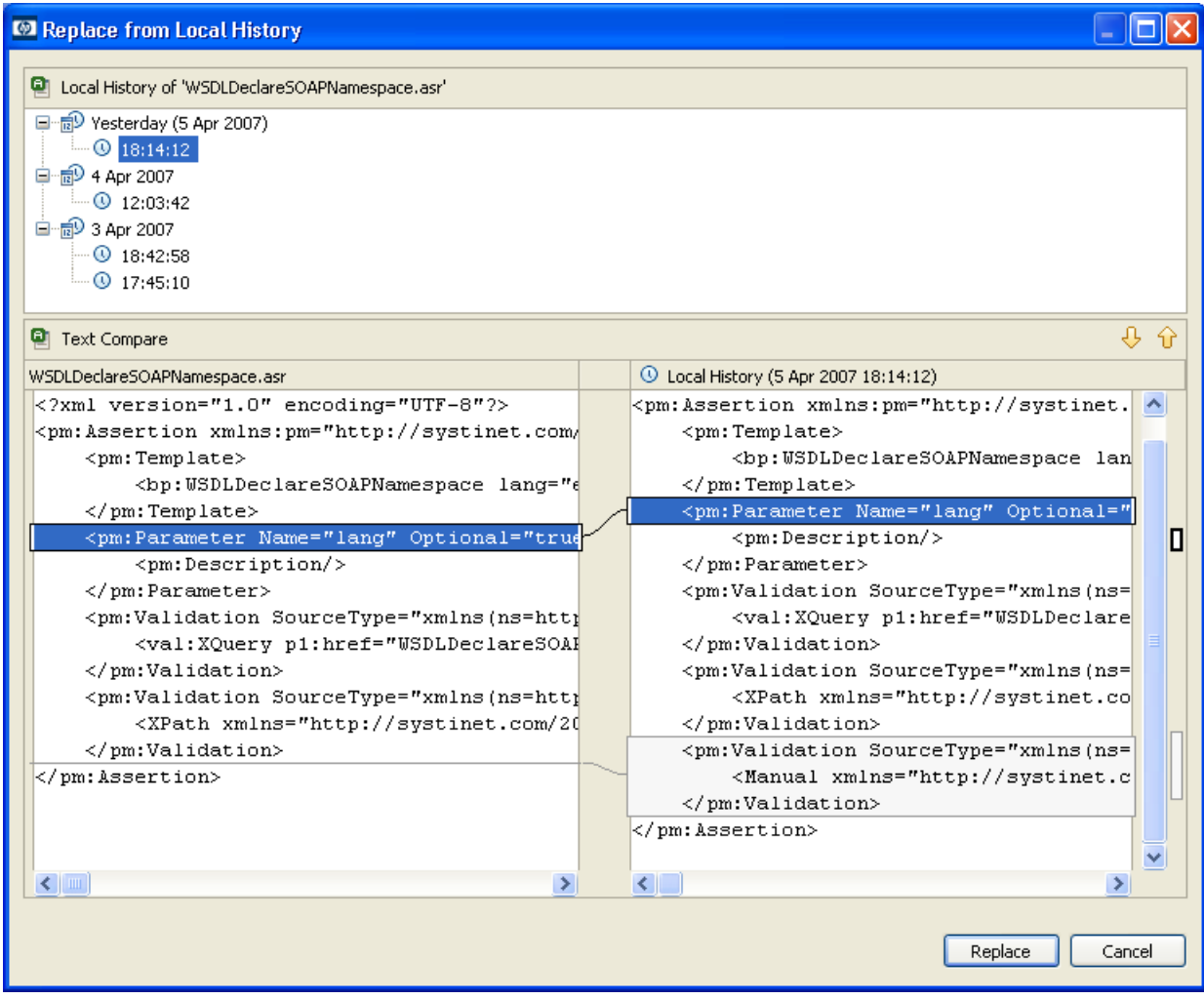
## Comparing Assertion Versions

Assertion Editor uses Eclipse's **Compare** function to track version numbers and let you roll back an assertion to a previous version. To compare versions of an assertion:

1 Find the assertion in the **Project Explorer. Note that the revision number is given in brackets after the name.**

2 Right-click the assertion. Its context menu opens.

3 Select **Replace With->Local History**. The **Replace with Local History** window opens.




Changes to XQuery implementations do not appear in this window. XQueries are held in separate, standalone files so they can be accessed by external XML editors. Please use your editor's revision control feature for XQueries.

4 Read through the differences between versions.

5 Click **Replace** to replace the current version with the one you are comparing it to, or click **Cancel**.

For more information about the **Compare With** and **Replace With** features, see the Eclipse help contents.

## Publishing Assertions

After writing, editing and testing an assertion, publish it to a SOA Systinet server. To publish an assertion:

1 Find the assertion in the **Project Explorer**. Assertions that have not been published are indicated by a question mark, **?**. Assertions that have been changed locally since they were last synchronized with the version on the server are indicated by a right arrow, >.

2 Right-click the assertion. Its context menu opens.

3 Select **SOA Systinet Server**->**Publish Assertion**. Assertion Editor publishes the assertion.

If the server to which you want to publish is not in the project, select **SOA Systinet Server**->**Publish to Other Server** in the assertion's context menu. The **New Server** wizard opens. This wizard is described in Downloading Assertions on page 28 .

If changes were made to the version on the server since you last synchronized, a conflict warning appears and asks you if you want to force publication. See Resolving Conflicts on page 43 for more information.

## Resolving Conflicts

Conflicts occur when different changes are made to your local copy and the server copy of an assertion and you then update or publish your local copy (see Editing Assertions on page 31 and Publishing Assertions on page 43). Assertion Editor warns you about the conflict and asks if you want to force the update or publication. Forcing an assertion to be updated overwrites any local changes you made. Forcing an assertion to be published overwrites any changes that were made to the version on the server. The safe way to deal with such conflicts is to cancel publication or update and then:

1 Copy your local version of the assertion to a different location in the **Project Explorer**.

2 Right-click one copy of the assertion in the **Project Explorer**. A context menu opens.

3 Select **SOA Systinet Server->Update**. A conflict warning appears.

4 Click **OK**. Assertion Editor then overwrites this local copy of the assertion with the version on the server.

5 Edit this updated version of the assertion, referring to the copy of the version you previously worked on.

# Deleting Assertions

If an assertion is no longer useful, you can do one of the following:

- Delete the local version. See Deleting a Local Assertion on page 44.
- Delete the server version. See Deleting an Assertion From the Server on page 44.

Instead of deleting the server version of an assertion, you may unpublish it.

## Deleting a Local Assertion

Deleting the local copy of the assertion does not affect the version on the server.

To delete a local copy of an assertion:

1 Find the assertion in the **Project Explorer**.

2 Right-click on the assertion.

3 A context menu appears. Select **Delete**.

4 A prompt appears. Confim you wish to delete the assertion.

## Deleting an Assertion From the Server

Deleting the version of an assertion that is on a server does not affect any local versions. You can then publish anew a local version to the server.

To delete an assertion on a server:

1. Find the assertion under its server in the **Server Explorer**.
2. Right-click on the assertion.
3. A context menu appears. Select **Delete assertion**.

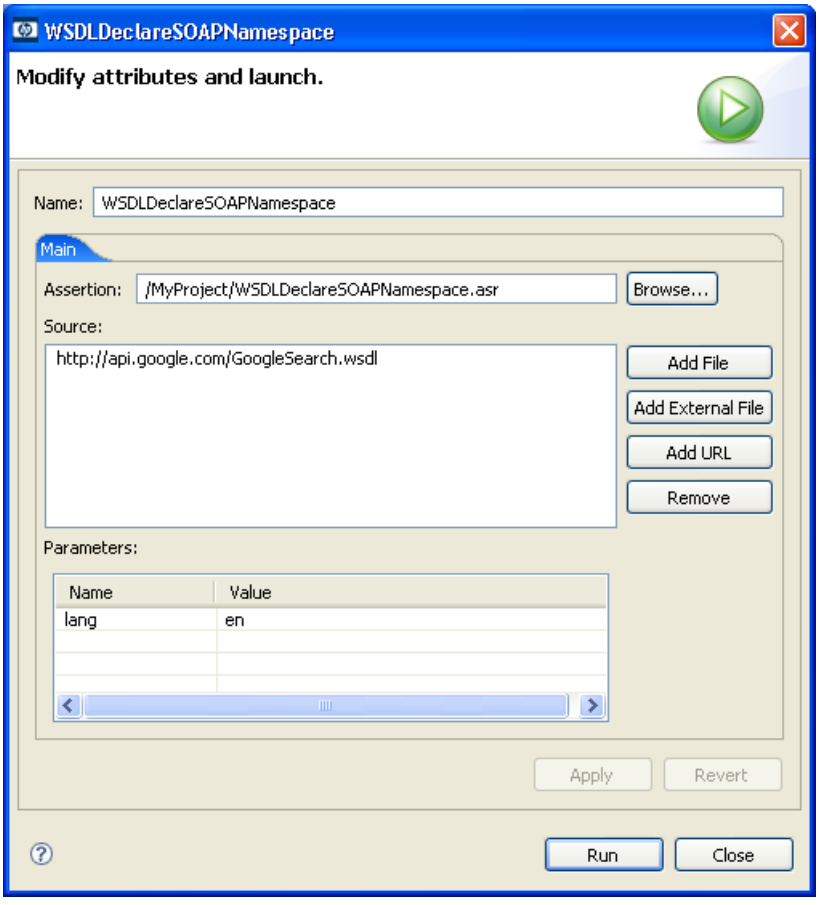
Alternatively, you can delete an assertion from the server from the **Project Explorer**. This gives you the option of deleting the local copy at the same time. To delete an assertion from the server and the local copy simultaneously:

1. Find the assertion in the **Project Explorer**.
2. Right-click on the assertion.
3. A context menu appears. Select **SOA Systinet Server->Delete Assertion**.
4. A prompt appears. Select **Also delete resources from local file system**. Alternatively, select **Do not delete resources on local file system** to delete only the server's copy of the assertion.

# 3 Testing Assertions

Before publishing an assertion, test it. To test an assertion:

1. Open the assertion in the **Editor** by double-clicking it in the **Project Explorer**.
2. From either the **Overview** or **Validations** tab, click **Test assertion**. The **Modify attributes and launch** dialogue opens.




3 If you wish to test a different assertion, click **Browse**. The **Choose Assertion** window opens, displaying a tree view of the Assertion Editor projects. Select an assertion and click **OK**.

4 Select source files for testing the assertion. You can:

- Click **Add File** to browse the Assertion Editor projects for source files.
- Click **Add External Files** to browse the local file system for source files.

- Click **Add URL** and type in the URL of a source file.

5 Highlight any unwanted source files and click **Remove**.

6 In the **Parameters** table, type in the parameter values you want to test and click **Apply**. To use the default parameter value assigned when the reference template was created, click **Revert**. For more information, see Editing Reference Templates on page 39.

7 Click **Run**. The test results appear in the **Console** tab (see ae.tour.console).

If the test results are unsatisfactory, edit the test as described in Editing Assertions on page 31.

# 4 Creating Custom Assertions

Assertion Editor comes with predefined elements that suffice for almost all use cases. However, you can customize the following:

- Source types, described in Customizing Source Type on page 51
- Validation handlers, described in Adding PM Extensions on page 52

## Customizing Source Type

When you define the implementation of an assertion, you choose the source type to which the assertion applies (see Adding and Deleting Implementations on page 33). Assertion Editor provides a list of source types and allows you to manually define a source type when you create an implementation. You may also add a source type to the list of predefined source types. To add, edit or delete a source type:

1. From the toolbar, select **Windows**->**Preferences**. The **Preferences** wizard opens.
2. Select **Assertion Editor**->**Source Type**. A table opens displaying source type names, local names and namespaces.
3. To delete an existing source type, select it and click **Delete**.
4. To edit an existing source type, select it and click **Edit**. The **Edit Source Type** window opens. Type in a new name, local name and/or namespace and click **OK**.
5. To create a new source type, click **Add**. The **New Source Type** window opens. Type in the name, local name and namespace and click **OK**.

## Adding PM Extensions

Policy Manager can be extended with custom-written validation handlers, in addition to the XQuery and XPath handlers that are included in the distribution. To add such an extension to your project:

1. Right-click the project. A context menu opens.
2. Select **Properties** from the context menu. The **Properties** window opens, with a tree view.
3. Select **PM Extensions** from the **Properties** tree view. A list of PM extensions in the project appears.
4. Click **Add PM Extension** to import a PM extension from another project, or click **Add External PM Extension** to add an extension from your filesystem.

After adding a PM extension to your Assertion Editor project, apply it to all relevant HP SOA Systinet Policy Manager servers with the Setup tool. Please see the SOA Systinet Administration Guide for details.

# Glossary

| | |
|---|---|
| artifact | A resource represented by a repository document. Artifacts may be categorized using taxonomies. This may be achieved by defining artifact property types in terms of taxonomies. |
| category | A division or classification in a taxonomy. |
| client | An application (or computer system) that enables use of server(s). See also server. |
| conflict | A simultaneous change to the same data that cannot be resolved. For example, automatic import or update of data from a SOA Systinet server may result in conflicts. |
| context action | An action performed using an item on a context menu. See also context menu. |
| context menu | A menu that pops-up when you right-click an item or selected items to perform actions possible on them. |
| credentials | Username and password for logging into a system such as a SOA Systinet server. |
| data integrity | The conformance of data to a format or structure and sometimes also the accuracy of the data. In particular, data items may refer to each other by means of identifiers or keys (such as taxonomy IDs or category names) and a referenced data item must then exist. Also, the meaning of data may assume constraints such as a fixed set of categories in a taxonomy. Such assumptions may be informal or hidden, so breaking the constraints may result in inaccurate data, regardless of its conformance to a particular format. |
| download | Retrieve copies of data from a SOA Systinet server, including initial copies or updates. Conflicts may prevent updates being applied if changes have also been made to local copies of data. See also import, update, Taxonomy XML file, conflict. |

| | |
|---|---|
| folder | A container for documents or files, such as a file system directory or an Eclipse folder. An Eclipse folder is a file system directory. |
| IDE | Integrated Development Environment |
| validation | The effect of change on data integrity in a SOA Systinet repository. |
| Validation Handler | A pluggable piece of code whose task is to validate a single assertion (or a collection of assertions of the same type, in the case of WSI, XQuery, etc.) against an input document. Its input consists of the input document, assertion being validated, definition of the assertion and a set of credentials. Its output is a validation report for this single assertion. |
| import | Retrieve copies of data from a file. If a local copy with the same name already exists, you will be asked if you want to overwrite it.<br>See also download, Taxonomy XML file. |
| JAR File | A file compressed using the Java Archive (JAR) file format. |
| perspective | An Eclipse perspective. This is a view that presents certain features to the user. |
| project | An Eclipse project containing a hierarchy of folders. Organization by projects and folders is reflected on your file system where taxonomies are stored as XML files. Taxonomy explorer displays SOA Systinet servers in a single hierarchy, so for most users a single project is sufficient for the folders of each SOA Systinet server. Projects are mainly of interest to users familiar with the Eclipse IDE. |
| property | An attribute or characteristic of an object, entity or artifact etc. that is represented by a value. |
| publish | Save changes made to copies of data to a repository in which it is stored. |
| registry | A type of repository in which taxonomies may be stored. A registry is a central listing of the available services and resources in SOA.<br>See also repository. |

| | |
|---|---|
| repository | A repository is used to store resources such as SOA artifacts, including taxonomies. Taxonomies may be also serve to define the repository data structure.<br>See also server. |
| REST | Representational State Transfer (REST) is a style of software architecture for distributed systems. SOA Systinet uses a RESTful simple interface to transmit domain-specific data over HTTP without an additional messaging layer. |
| restore | Restore a deleted taxonomy using a copy on hard disk. Taxonomies can only be restored if they were previously published to a server or downloaded from the server. Once a deletion has been published to the server, the deleted taxonomy can no longer be restored. |
| RCP | A Rich Client Platform such as provided by Eclipse. RCPs are used to produce rich client applications with all the benefits of re-use.<br>See also rich client. |
| rich client | A client application that enables use of server(s) and the functionality they provide and provides additional features through client-side processing.<br>See also client. |
| server | A computer that provides a service, such as access to a repository through a particular port. To access the service requires knowledge of the computer name (its hostname), the port number and the protocol used for communication.<br>See also repository, client. |
| Service Oriented Architecture | Service Oriented Architecture is a simple software design approach in which all functions are modeled as services and are built to be consumed over a network using standards-based interoperability and policies. |
| SOA | See Service Oriented Architecture. |
| SOA Systinet server | A server running SOA Systinet. SOA Systinet servers have taxonomies stored in a repository.<br>See also server, SOA Systinet. |

| | |
|---|---|
| system taxonomy | These taxonomies are generated automatically upon starting the SOA Systinet platform and are not meant to be changed by the user.<br>See also SOA Systinet server, taxonomy. |
| SOA Systinet | HP's latest suite of products for SOA Governance and Lifecycle Management. It consists of a platform and a number of applications. The platform provides repository storage and a data model in which taxonomies play an important part. |
| taxonomy | A taxonomy defines a set of values that can be used for the classification of artifacts. |
| Taxonomy XML file | An XML file holding the *taxonomy* entity from a *registry*. The root element of this file is <taxonomy>.<br>See also import. |
| tModel | A structure that takes the form of keyed metadata (data about data). In a general sense, the purpose of a tModel within the UDDI specification is to provide a reference system based on abstraction. Among the roles that a tModel plays in UDDI is the ability to provide and to describe compliance with a specification or concept to a taxonomy, for example. |
| UDDI | See Universal Description, Discovery and Integration. |
| UDDI Registry | A UDDI Registry is an implementation of the UDDI specification that allows web service vendors to register information about the web services they offer so that others can find them.<br>See also registry. |
| Universal Description, Discovery and Integration | UDDI is a specification for distributed Web-based information registries of Web services. |
| URI | Uniform Resource Identifier – the generic term for all types of names and addresses that refer to objects on the World Wide Web. A URL is one kind of URI. |
| URL | Uniform Resource Locator – the global address of documents and other resources on the World Wide Web. The first part of the address indicates |

| | |
|---|---|
| | what protocol to use and the second part specifies the IP address or the domain name where the resource is located.<br>See also URI. |
| update | Updating local copies of data on a SOA Systinet server with respect to any changes made in the repository. Also obtaining local copies where there are none, but the term *download* more accurately describes both these cases.<br>See also download. |