Rational Integration Tester

# Reference Guide for HTTP & Web Services

*Version 8.0.1*

**Note**

Before using this information and the product it supports, read the information in "Notices" on page 61.

This edition applies to version 8.0.1 of Rational Integration Tester and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# About this Publication

**Contents**

This guide describes how to configure and run IBM® Rational® Integration Tester using the HTTP transport and SOAP message formatters, which provide support for HTTP-based communications and the ability to communicate with Web Services.

## Intended Audience

This document intended to be read by those with a fair understanding and exposure to the concepts involved in both testing and development and in enterprise integration.

## Scope

This document discusses the configuration and use of HTTP and Web Services technologies in IBM Rational Integration Tester. Information about creating and deploying Web Services is beyond the scope of this document.

If you wish to familiarize yourself with these technologies, please refer to the relevant documentation from the appropriate providers or individuals.

## Typographical Conventions

The following typographical conventions are observed throughout this document:

| Type | Usage |
|---|---|
| `Constant Width` | Program output, listings of code examples, file names, commands, options, configuration file parameters, and literal programming elements in running text. |
| *Italic* | Document title names in statements that refer you to other documents. Also used to highlight concepts when first introduced. |
| Bold | Menu items in graphical user interface windows (such as Microsoft Windows-based or UNIX X Window applications) from which you select options or execute macros and functions.<br>Submenus and options of a menu item are indicated with a "greater than" sign, such as **Menu > Submenu** or **Menu > Option**. |

## Contacting IBM Support

To contact IBM Support, see: www.ibm.com/contact/us/en/

# Understanding Web Services

Contents

This chapter provides an overview of Web Services and how the technology is supported by Rational Integration Tester.

## 1.1 Overview

The Web services Architecture group at the W3C has defined a Web service as follows:

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards

Sending a request to a Web Service requires a message to be constructed and sent to where the Web Service resides, this message must conform to the Web Services expected structure as defined in a WSDL document that should be made available to all users of the service.

When a Web Service receives a request from a client it processes the required function of the service and prepares a reply. The reply is sent back to the client in the pre-defined method which is generally as a SOAP formatted message and includes the information requested.

A simple example of a Web Service is a stock quote system where a client uses a Web Service to provide real-time stock value data. In this case the client would send a request to the Web Service listing the name of the stock/company and would receive a reply stating its current value.

## 1.2 Applicable Technologies

Rational Integration Tester provides access to the following standards / technologies to provide support for Web Services:

- HTTP
- HTTPS
- WSDL
- SOAP

### 1.2.1 HTTP

Hyper Text Transfer Protocol (HTTP) is an application level text based protocol that provides the means for clients to interact with Web Servers using a series of requests and responses. It defines several operations that allow clients to request web content and submit information to the server for processing.

### 1.2.2 HTTPS

HTTPS is syntactically identical as HTTP. Using an https: URL indicates that HTTP is to be used, but with a different default port (443) and an additional encryption/ authentication layer (SSL) which sits between HTTP and TCP.

### 1.2.3 WSDL

Web Services Description Language (WSDL) is an XML-based specification that describes a Web Service. A WSDL document describes Web Service operations, input and output parameters, and how a client application connects to the Web Service, that is, the transport to be used.

### 1.2.4 SOAP

SOAP (Simple Object Access Protocol) is a lightweight XML-based protocol used to exchange information in a decentralized, distributed environment. The protocol consists of:

- An envelope that describes the SOAP message. The envelope contains the body of the message, identifies who should process it, and describes how to process it.
- A set of encoding rules for expressing instances of application-specific data types.
- A convention for representing remote procedure calls and responses.

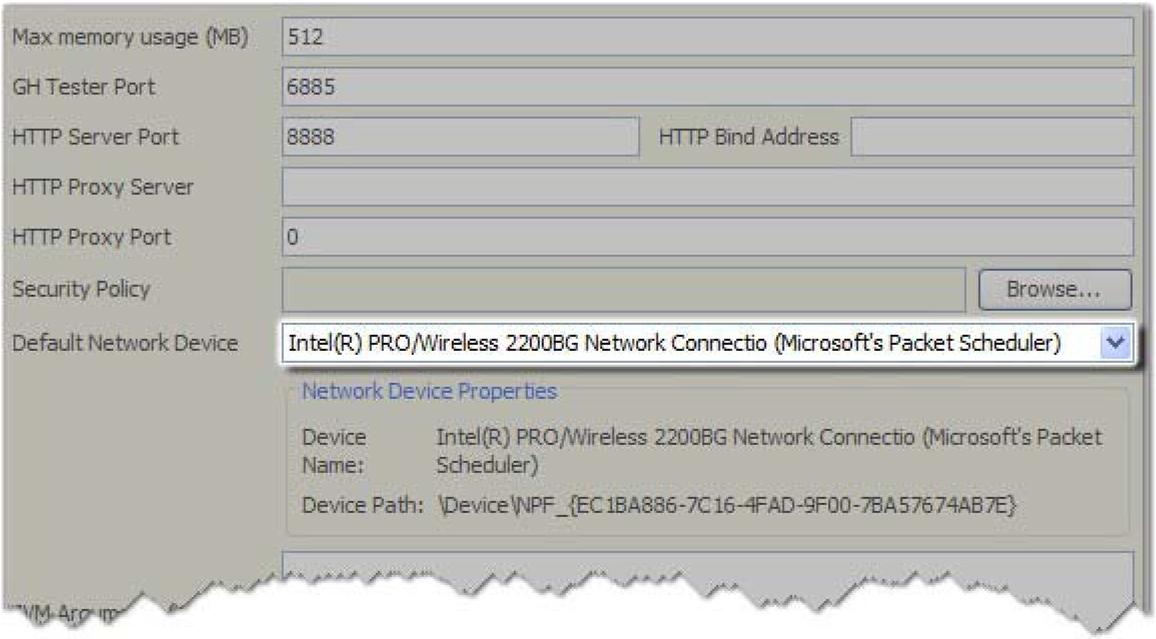
This information is embedded in a Multipurpose Internet Mail Extensions (MIME)-encoded package that can be transmitted over HTTP or other protocols. MIME is a specification for formatting non-ASCII messages so that they can be sent over the Internet.

## 1.3 Packet Capture Libraries

To use an HTTP Subscriber in “watch” mode, you must have the appropriate HTTP packet capture libraries installed on your machine.

- On UNIX machines, **libpcap** is normally installed already. If necessary, the latest package can be downloaded from http://www.tcpdump.org/.
- On Windows machines, **WinPCAP** will be installed as part of the main Rational Integration Tester installation.

The libraries enable Rational Integration Tester to access network devices on the installed machine so that it can monitor HTTP traffic. To do this, the network device to be monitored must be specified under **Application** in Rational Integration Tester’s Library Manager utility, as shown below.



| | | | |
|---|---|---|---|
| Max memory usage (MB) | 512 | | |
| GH Tester Port | 6885 | | |
| HTTP Server Port | 8888 | HTTP Bind Address | |
| HTTP Proxy Server | | | |
| HTTP Proxy Port | 0 | | |
| Security Policy | | Browse... | |
| Default Network Device | Intel(R) PRO/Wireless 2200BG Network Connectio (Microsoft's Packet Scheduler) | | |

Network Device Properties

| | |
|---|---|
| Device Name: | Intel(R) PRO/Wireless 2200BG Network Connectio (Microsoft's Packet Scheduler) |
| Device Path: | \Device\NPF_{EC1BA886-7C16-4FAD-9F00-7BA57674AB7E} |

The **Default Network Device** field provides a list of network interfaces that can be monitored. The properties of the selected device are displayed below it to help you choose the correct device.

After selecting the desired device, click **OK** to close Library Manager and save your changes.

# HTTP Transport

**Contents**

Rational Integration Tester provides access to HTTP- and HTTPS-based communications through an extension of its transport mechanism. The implementation provides both Client and Server facilities.

Standard HTTP and HTTPS operations (GET, POST, PUT, DELETE, HEAD, OPTIONS) are supported by Rational Integration Tester.

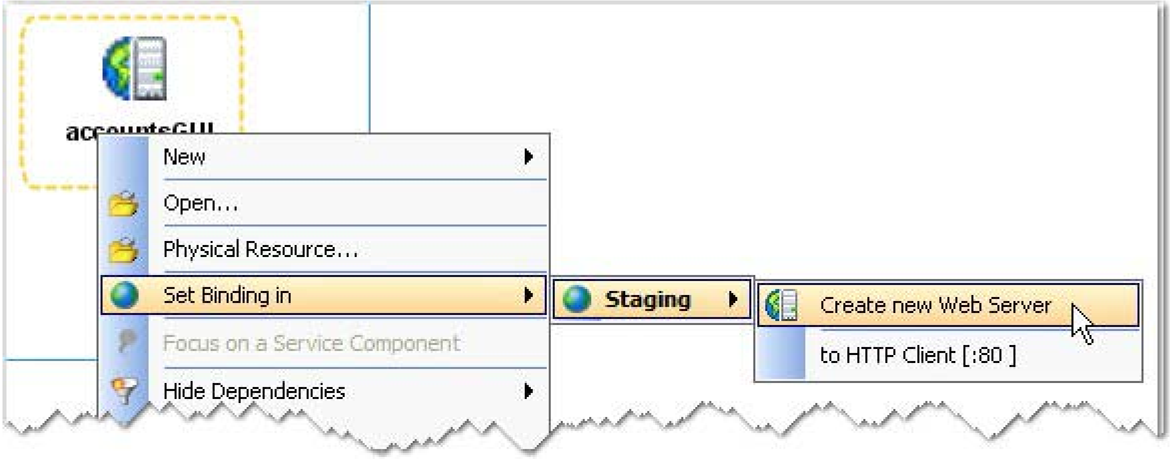
Further information about HTTP methods can be found on the W3C website, and full details are provided in RFC-1945 and RFC-2616.

## 2.1 Creating an HTTP Transport

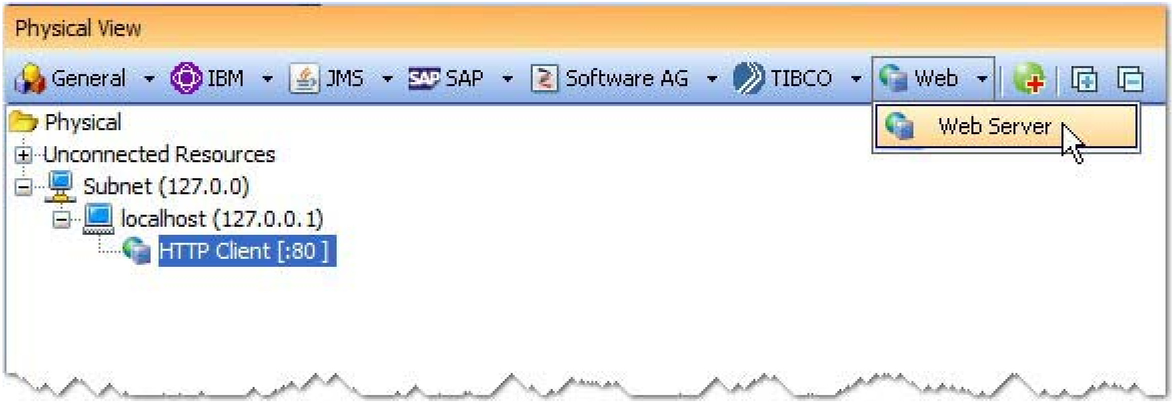
An HTTP transport is created when you create a physical Web Server resource in Rational Integration Tester's Architecture School.

In Architecture School, you can create a new resource in two ways:

- In the Logical View, right-click on an HTTP connection and select the **Set Binding in > <environment> > Create a new Web Server** option.



- In the Physical View, select the **Web > Web Server** option.

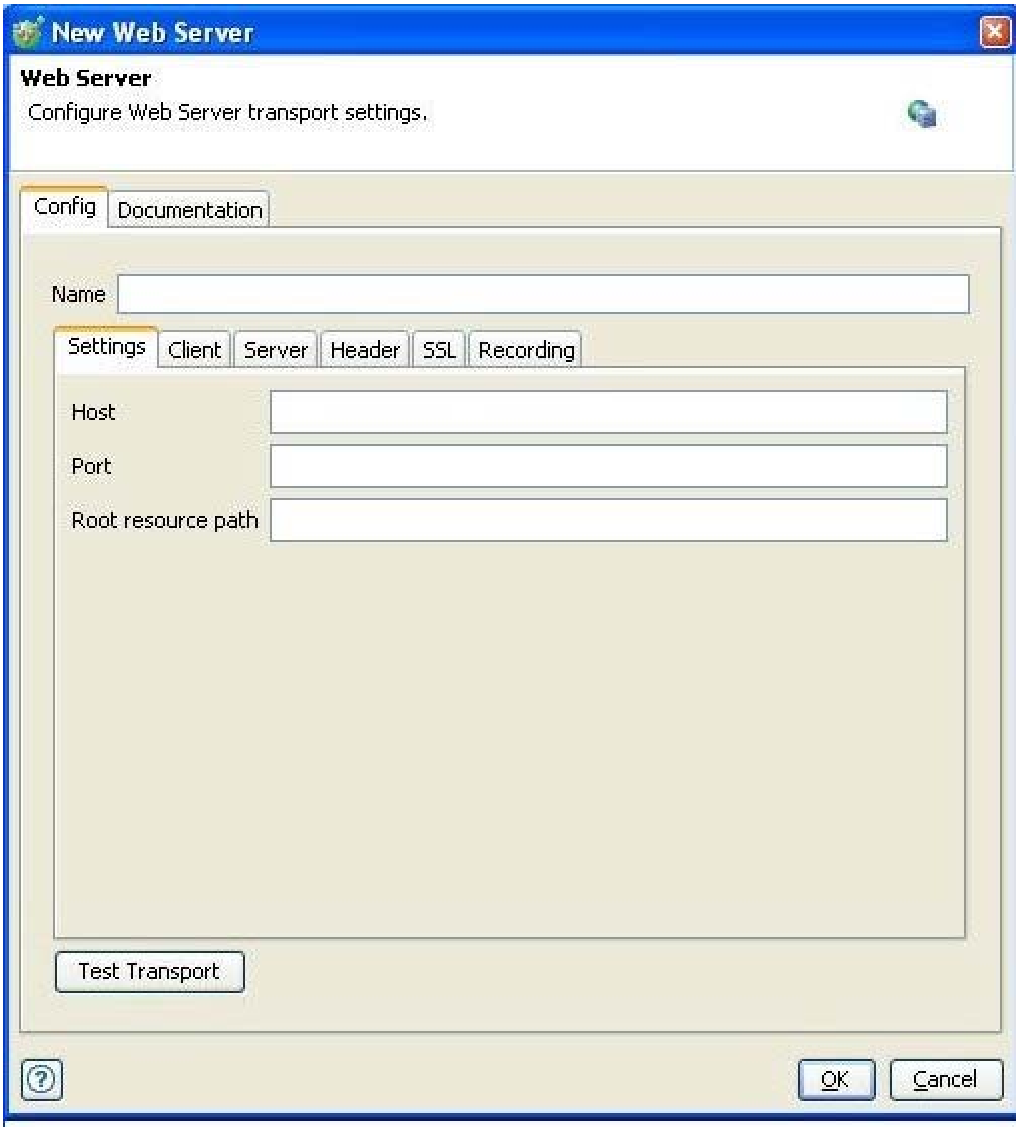


Each physical resource will represent an HTTP transport that can be selected and configured later on.

## 2.2 Configuring an HTTP Transport

To configure an HTTP transport, double-click the appropriate Web Server resource in Architecture School's Physical View.

The settings for the transport are separated into six tabs: **Settings**, **Client**, **Server**, **Headers**, **SSL**, and **Recording**.



Under the **Settings** tab, you can create single HTTP client/server transports.

If desired, enter a name for the transport in the **Name** field (to help identify it when multiple web servers are available).

The available settings are described in the following table:

| | |
|---|---|
| Host | The host name or IP address of the web server (using a host name requires proper DNS configuration on the local machine). |
| Port | The TCP port on the server to which the connection should be made. |
| Root resource path | The root URL path that should be appended to a resource that is requested in a Publish or Send request action that uses this HTTP transport. |

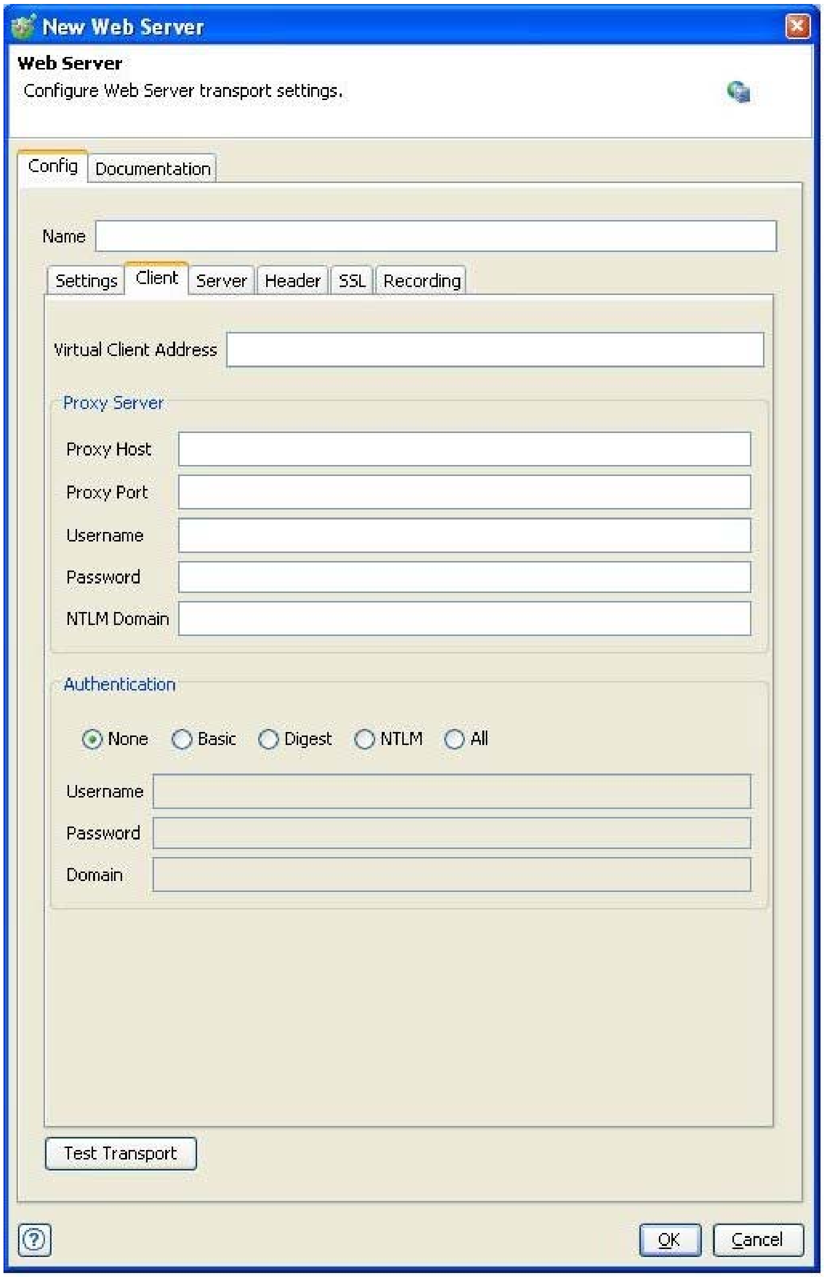
The **Client** and **Server** tabs enable you to enter additional client/server settings.

Additional configuration options are described in HTTP Header Settings and Content-Type and Content-Encoding.

**NOTE:** Rational Integration Tester 8.0.1 (or later) provides a sift-and-pass-through capability for any message-based stubs that use HTTP, IBM WebSphere MQ, or webMethods Integration Server transports. If you want to specify the default pass-through action for all HTTP transports, use the **Pass Through** window on the **Virtualization** page of the Preferences dialog box. The default pass-through action can be modified for each stub. (For information about using the Virtualization page and configuring pass-through actions for message-based stubs, refer to *IBM Rational Test Virtualization Server Reference Guide.*)

## 2.2.1 Client Settings

To configure HTTP client settings, click the **Client** tab.



New Web Server

Web Server

Configure Web Server transport settings.

Name

Virtual Client Address

Proxy Server

Proxy Host

Proxy Port

Username

Password

NTLM Domain

Authentication

● None ○ Basic ○ Digest ○ NTLM ○ All

Username

Password

Domain

In the **Virtual Client Address** field, enter the IP address to use as the 'from' address when using Rational Integration Tester on virtual machines.

The configuration is broken into the following main areas:

- Proxy Settings
- Authentication

**NOTE:** The use of SSL is described in SSL Configuration.

### Proxy Settings

Proxy settings are configured in the **Proxy Server** section.

The available settings are described in the following table:

| | |
|---|---|
| Proxy Host | The host name or IP address of the proxy server to use for HTTP connections. |
| Proxy Port | The TCP port on the proxy server to which the connection should be made. |
| Username/Password | Optional user name and password to use for connecting to the proxy server. |
| NTLM Domain | If running under the Windows platform, you can specify a domain name in which the user account should be authenticated. |

### Authentication

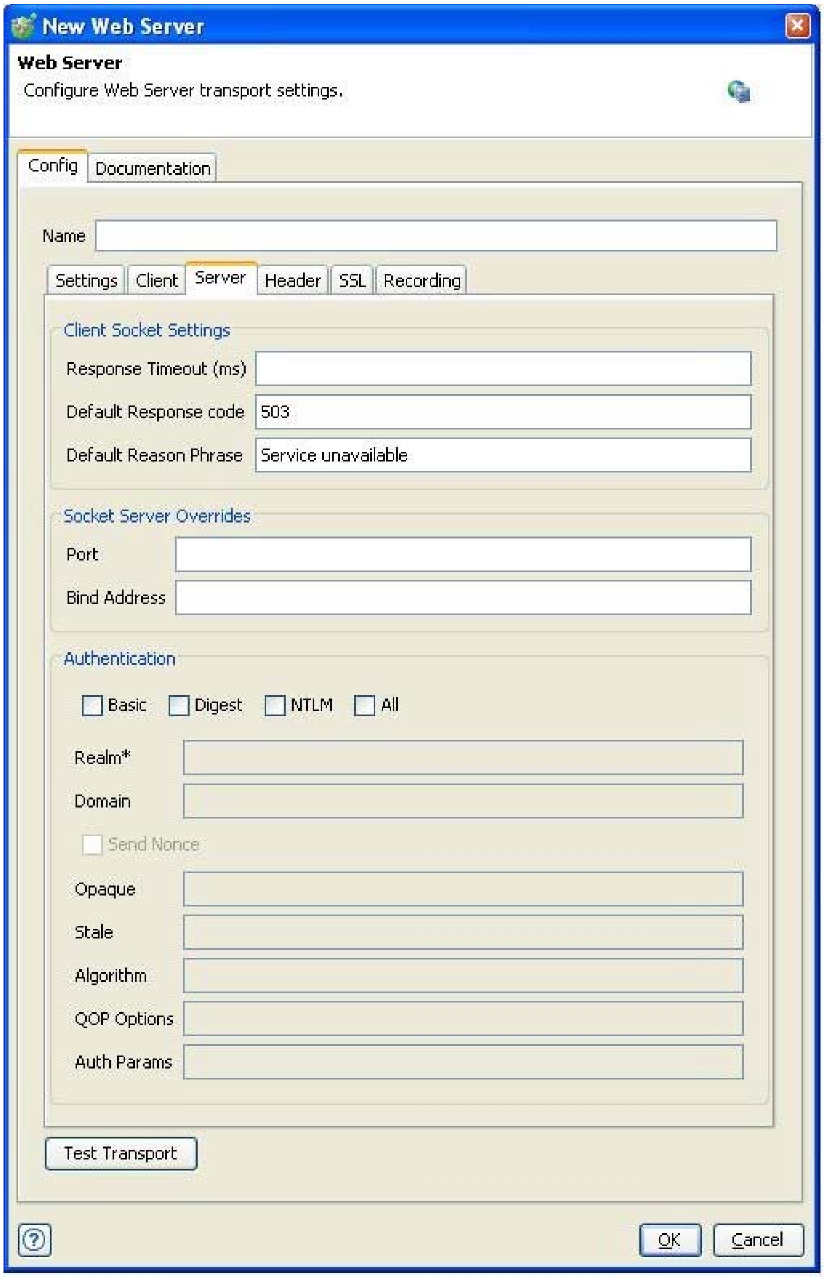
If authentication is required for connections to the specified web server, the method and its details can be configured under **Authentication**.

The available options are described in the following table:

| | |
|---|---|
| None | The default option, to be used when no authentication is required. |
| Basic | Basic authentication that transmits a non-encrypted user name and password. |
| Digest | Digest authentication that sends an encrypted user name and password. |
| NTLM | Windows authentication that sends an NT or Active Directory user name, password, and domain. |
| All | If the specific method of authentication is unknown, use this option to have the client attempt all options. |

### 2.2.2 Server Settings

To configure HTTP server settings, click the **Server** tab.



Name

Client Socket Settings

Response Timeout (ms)

Default Response code 503

Default Reason Phrase Service unavailable

Socket Server Overrides

Port

Bind Address

Authentication

□ Basic □ Digest □ NTLM □ All

Realm*

Domain

□ Send Nonce

Opaque

Stale

Algorithm

QOP Options

Auth Params

The configuration is broken into the following main areas:

- Client Socket Settings
- Socket Server Overrides
- Authentication

**NOTE:** The use of SSL is described in SSL Configuration.

### Client Socket Settings

Client Socket settings are configured in the **Client Socket Settings** section.

The available options are described in the following table:

| | |
|---|---|
| Response Timeout | The amount of time (in milliseconds) that the client socket should remain open. After the time-out expires, the socket will no longer accept a response from any outstanding communication. |
| Default Response code | The default response code to return to the client as part of the HTTP header field when an error occurs. Response codes can be used to determine the cause of failures (for example, code 404 is a standard response code for a file not found). |
| Default Reason Phrase | The default error message to be returned to the client when an error occurs. |

### Socket Server Overrides

Socket Server Override settings are configured in the **Client Socket Overrides** section.

The available options are described in the following table:

| | |
|---|---|
| Port | This is an optional field for specifying an alternate TCP port that the server should use to listen for HTTP requests. |
| Bind Address | This is an optional field for specifying an alternate host name or IP address to which the server should be bound. |

### Authentication

If authentication is required on the web server, the method and its details can be configured under **Authentication**.

The available authentication methods are described below:

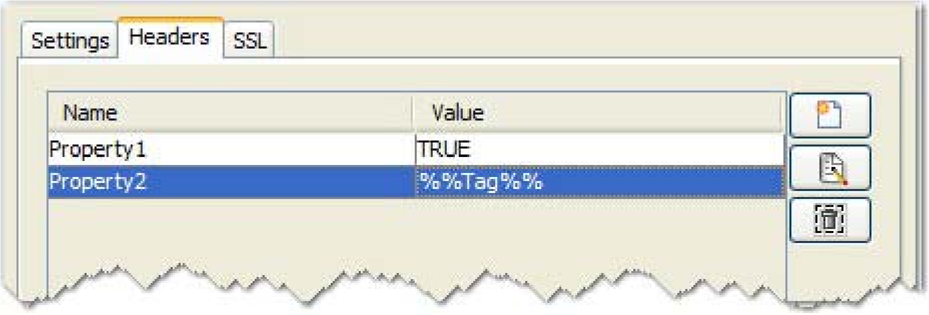
- **Basic** - basic authentication requiring a user name and password from the client (**Realm** is required when using basic authentication).
- **Digest** - digest authentication that expects an encrypted user name and password.
- **NTLM** - Windows authentication that expects an NT or Active Directory user name, password, and domain.
- **All** - use this option to enable all authentication types on the server.

Additional options, required depending on the authentication method(s) in use, are described in the following table:

| | |
|---|---|
| Realm | The realm is presented to the client and helps users understand which user name and password to enter. The realm should contain the name of the authenticating host and may indicate the group of users who might have access (for example, registered_users@abc.domain.com). |
| Domain | A quoted list of URIs that define the protected space of the web server. Multiple URIs should be separated by a space. |
| Send Nonce | Enable this option to send a nonce (a unique string of data generated with each server response). |
| Opaque | A string of data which should be returned unchanged by the client in the authorization header of all subsequent requests to the same domain. This string should be base64 or hexadecimal data. |
| Stale | A flag indicating that the previous client request was rejected because the nonce value was incorrect (stale). This value should be set to TRUE if the client has sent the correct user name and password. This value should be left blank or set to false when an invalid user name and password are received. |
| Algorithm | A pair of algorithms that should be used to generate the digest and checksum. If left blank, MD5 is used. |
| QOP Options | An optional, quoted string that indicates the quality of protection values supported by the server. “auth” indicates authentication, and “auth-int” indicates authentication with integrity protection. |
| Auth Params | Not currently used. |

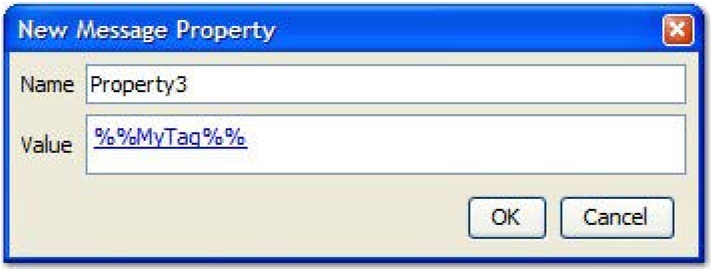
## 2.2.3 HTTP Header Settings

The **Header** tab enables users to define key-value pairs to be added to HTTP message header fields on all messages that are sent using the transport.



| Name | Value |
|---|---|
| Property1 | TRUE |
| Property2 | %%Tag%% |

To add a new pair, click 📄. To edit an existing pair, select it and click 📝. To delete an existing pair, select it and click 🗑.



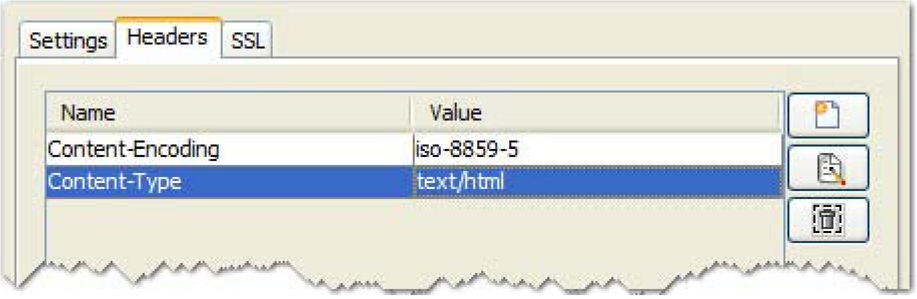
Name: Property3

Value: %%MyTag%%

Various type validation options are provided and values can be populated from tags.

### 2.2.3.1 Content-Type and Content-Encoding

Content-Type and Content-Encoding (that is, charset) properties for all messages that use the selected transport can be set with HTTP Headers (see HTTP Header Settings). An example of using these options is shown below.

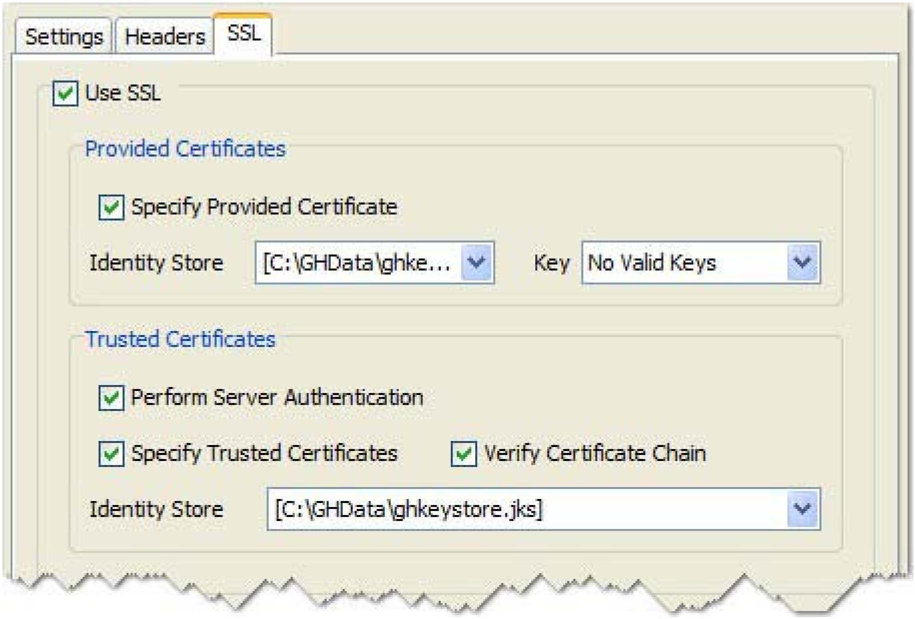


| Name | Value |
|---|---|
| Content-Encoding | iso-8859-5 |
| Content-Type | text/html |

If nothing is specified for Content-Encoding, ISO8559-1 is used. This option should be fine for all Latin-1 characters within the unicode range of u0000 through u00FF. If the message body contain characters outside this range, the encoding will be set automatically to UTF.

### 2.2.4 SSL Settings

The HTTP transport SSL settings are configured under the **SSL** tab, as shown below:



☑ Use SSL

Provided Certificates

☑ Specify Provided Certificate

Identity Store [C:\GHData\ghke... Key No Valid Keys

Trusted Certificates

☑ Perform Server Authentication

☑ Specify Trusted Certificates ☑ Verify Certificate Chain

Identity Store [C:\GHData\ghkeystore.jks]

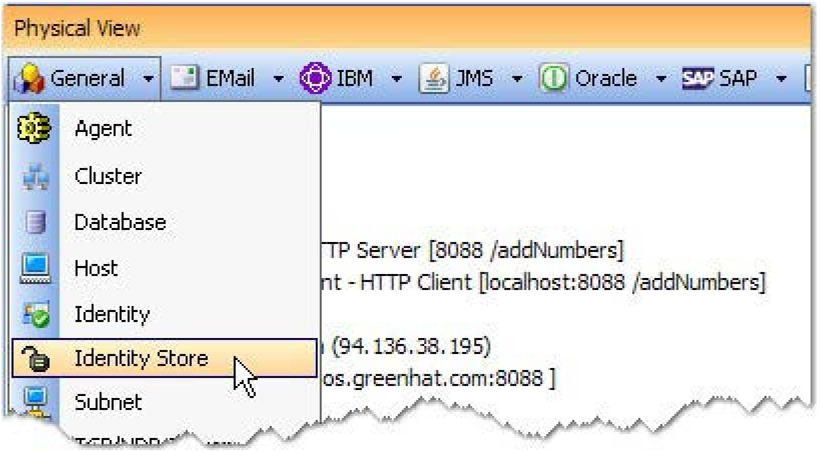
Specify the desired options under **Provided Certificates** (client settings) and **Trusted Certificates** (server settings), and select the desired Identity Store to use for each one.

Before using HTTPS (HTTP over SSL), an identity store must be created in the Rational Integration Tester project to store certificates. For information about this, refer to Creating Identity Stores.

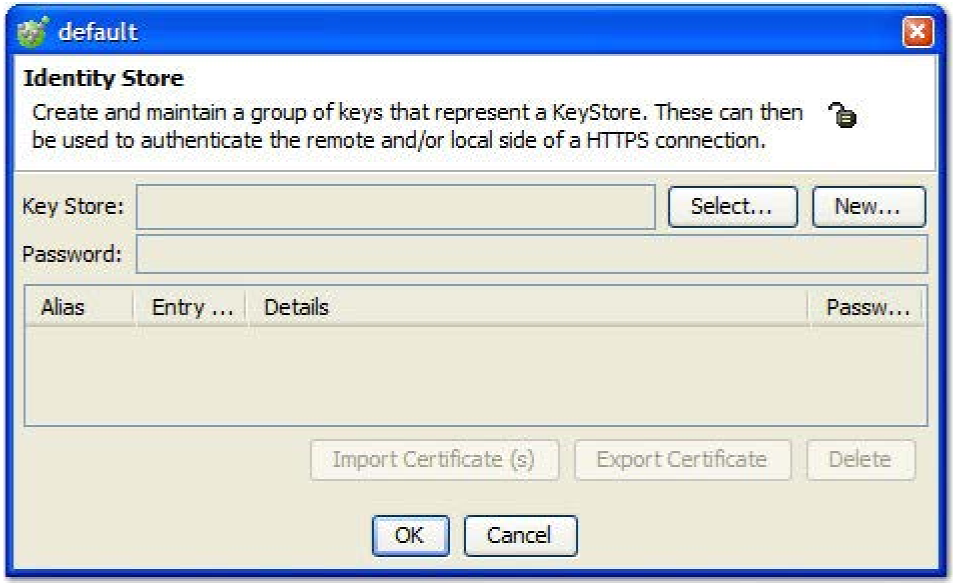
### 2.2.4.1 Creating Identity Stores

An identity store is added to a project in Architecture School's Physical View.

1. In the Physical View, select **Identity Store** from the **General** toolbar option.



2. Double-click the new identity store to edit it.



default

**Identity Store**

Create and maintain a group of keys that represent a KeyStore. These can then be used to authenticate the remote and/or local side of a HTTPS connection.

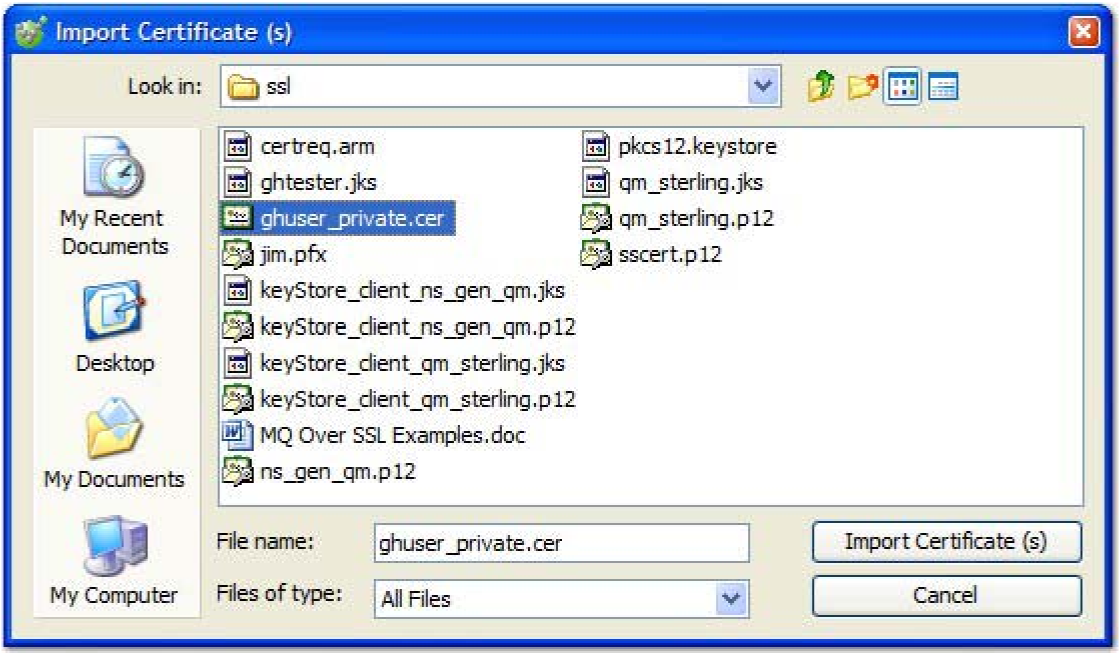
Key Store:

Password:

| Alias | Entry ... | Details | Passw... |
|---|---|---|---|

3. Create a new Key Store by clicking **New**, or click **Select** to browse to an existing store. When creating a new Key Store, you will need to provide a location and name, and specify a password.

4. Click **Import Certificate(s)** to load certificates into the store.



Look in: ssl

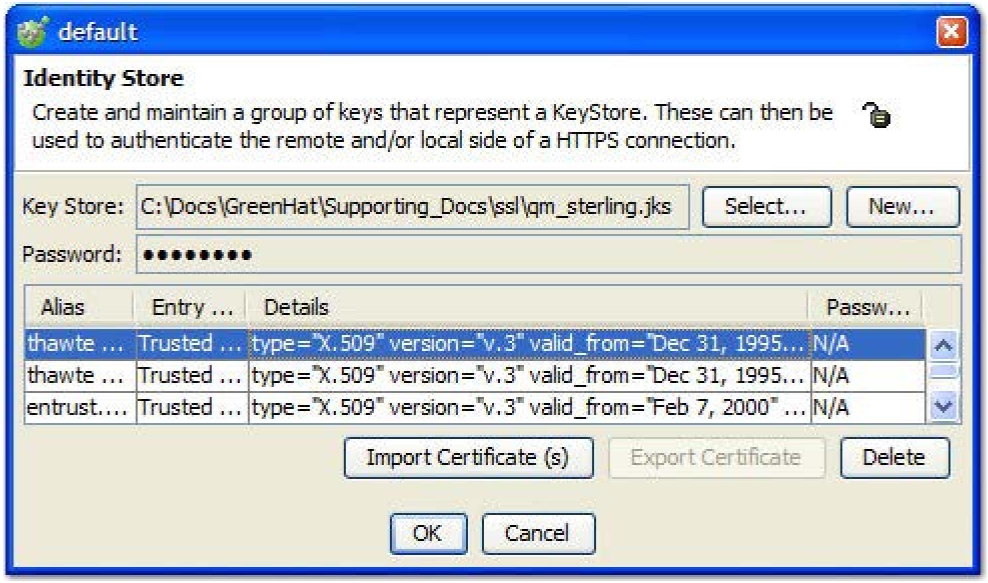
File name: ghuser_private.cer

Files of type: All Files

5. Locate and select the certificate, then click **Import Certificate(s)**.

**NOTE:** If desired, you can add multiple certificates to the Key Store.

6. When finished, click **OK** to create the Identity Store.



**Identity Store**

Create and maintain a group of keys that represent a KeyStore. These can then be used to authenticate the remote and/or local side of a HTTPS connection.

Key Store: C:\Docs\GreenHat\Supporting_Docs\ssl\qm_sterling.jks

Password: ••••••••

| Alias | Entry ... | Details | Passw... |
|---|---|---|---|
| thawte ... | Trusted ... | type="X.509" version="v.3" valid_from="Dec 31, 1995... | N/A |
| thawte ... | Trusted ... | type="X.509" version="v.3" valid_from="Dec 31, 1995... | N/A |
| entrust.... | Trusted ... | type="X.509" version="v.3" valid_from="Feb 7, 2000" ... | N/A |

## 2.2.5 Recording Settings

HTTP transport recording settings are configured under the **Recording** tab.

The following table describes the fields on the tab.

| Field | Description |
|---|---|
| Recording Mode | Options are follows:<br>• Packet Capture. This requires packet capture software. For information about installing packet capture software, refer to *IBM Rational Integration Tester Installation Guide*.<br>• External Proxy Server. The proxies in the Rational Integration Tester Platform Pack enable Rational Integration Tester and Rational Test Virtualization Server to record all HTTP(S) traffic routed through the proxy. For information about this option, refer to *IBM Rational Integration Tester Reference Guide* and *IBM Rational Integration Tester Platform Pack Installation Guide*. |
| Proxy Server Port | Clicking Internal Proxy Server in the Recording Mode list makes this optional field available for editing.<br>If you want to use a specific port number (on the local computer), enter the number in this field. |
| Proxy Server Bind Address | Clicking Internal Proxy Server in the Recording Mode list makes this optional field available for editing.<br>If you want to use a specific IP address (on the local computer), enter the address in this field. |

For more information about HTTP recording, refer to *IBM Rational Integration Tester Reference Guide*.

# WSDL and SOAP Messages

Contents

This chapter provides information about creating and applying schemas in Rational Integration Tester, as well as information about the structure of WSDL and SOAP messages.

## 3.1 Overview

A WSDL file is a schema that describes a Web Service, including information on how to locate that service and what operations are supported. Within Rational Integration Tester the WSDL file enables users to construct a SOAP message that can be sent to a Web Service and to construct a template message in order to validate replies from Web Services.

Within every WSDL document a **Binding** element defines the message format and protocol used for the operations and messages that it supports. Rational Integration Tester currently supports the SOAP protocol for this purpose and as such all requests sent to Web Services should make use of the SOAP Document-Literal or RPC encoded formatters.

Additional information relating to SOAP can be found at http://www.w3.org/TR/soap.

## 3.2 Adding a WSDL to Rational Integration Tester

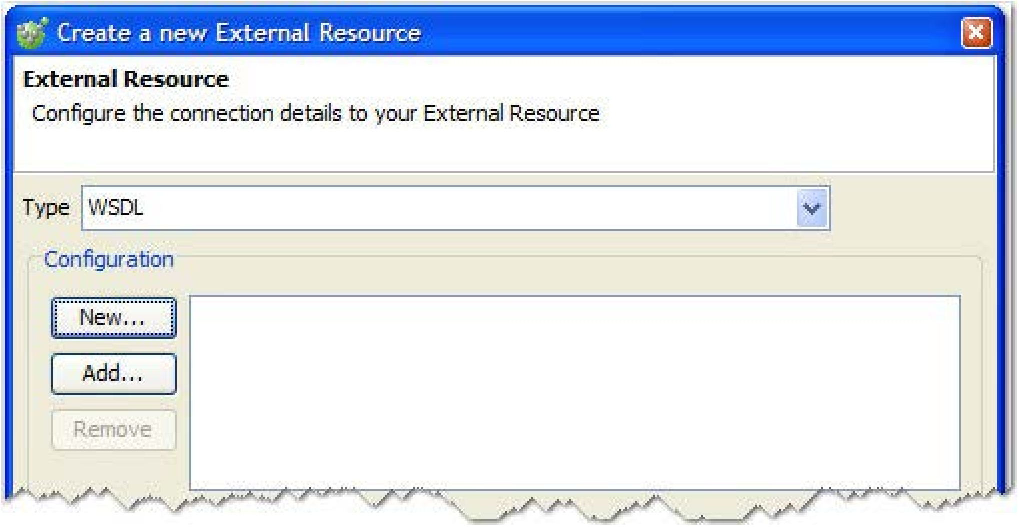
A WSDL can be added to Rational Integration Tester as a service component (an external resource that can be synchronised with the project) or as a schema that can be applied to messages. This section describes how to add a WSDL as a component.

**NOTE:** For more information about adding a WSDL as a schema or applying one to a message, and about synchronization and external resources, refer to *IBM Rational Integration Tester Reference Guide.*

Follow the steps below to add a WSDL as an external resource:

1. Drag a WSDL from the file system into Architecture School's Logical View, or select **WSDL** from the **General** component menu in Architecture School's Logical View.

The **External Resource** wizard is displayed.



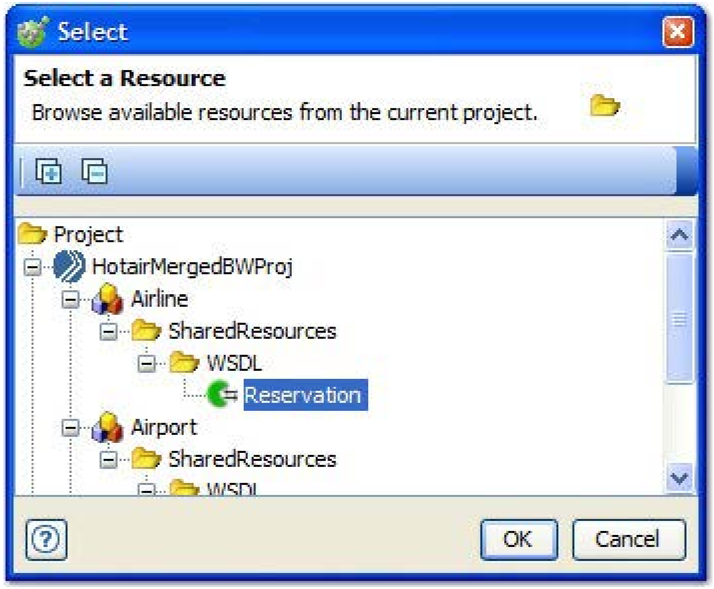
Create a new External Resource

**External Resource**

Configure the connection details to your External Resource

Type: WSDL

2. To add an existing WSDL resource from the current Rational Integration Tester project, click **Add**, then locate and select the WSDL from the project resource dialog.



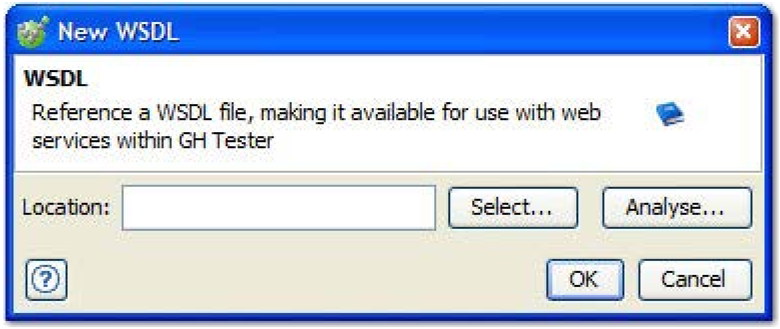
**Select a Resource**

Browse available resources from the current project.

**NOTE:** You can also drag and drop a WSDL into the Logical View of the Architecture School perspective.

3. To add a new WSDL, click **New**.

The **New WSDL** dialog is displayed.



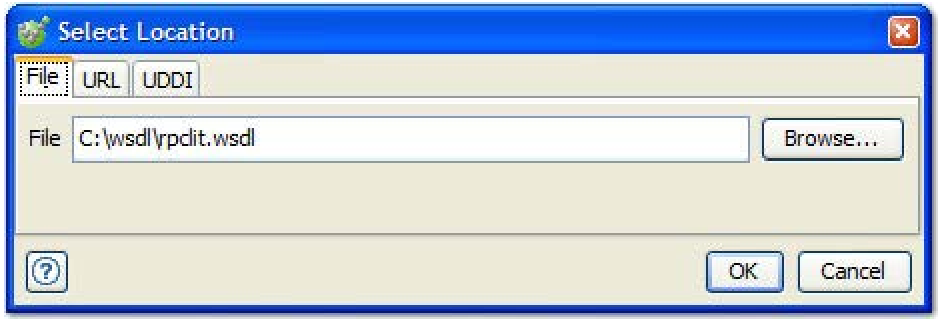
**WSDL**

Reference a WSDL file, making it available for use with web services within GH Tester

Location:

4. Click **Select** to locate the WSDL.

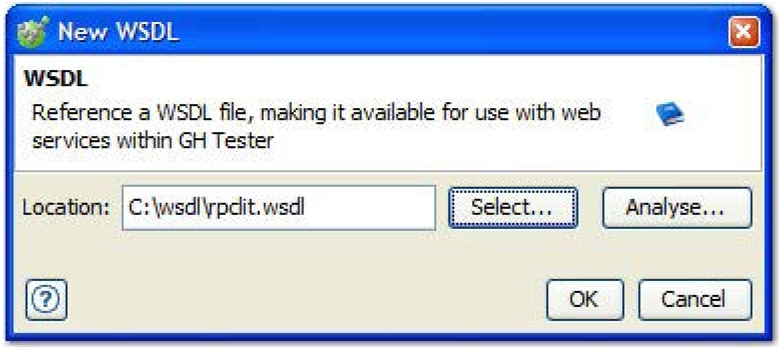
The **Select Location** dialog is displayed.



File C:\wsdl\rpclit.wsdl

5. Locate the desired WSDL using one of the three tabs in the dialog, as follows: Add a WSDL from File, Add a WSDL By Means of a URL, or Add a WSDL By Means of UDDI.

6. Click **OK** after you have entered the location of the desired WSDL.



WSDL

Reference a WSDL file, making it available for use with web services within GH Tester

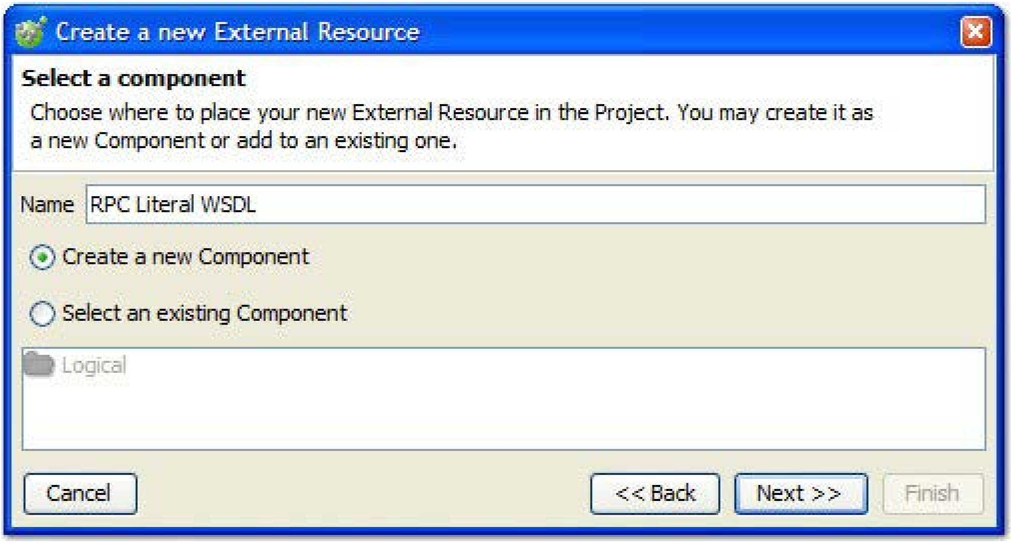
Location: C:\wsdl\rpclit.wsdl

7. If desired, click **Analyze** to perform an analysis (including a WS-I Conformance Report) on the selected WSDL (for more information about this, refer to *IBM Rational Integration Tester Reference Guide*). Otherwise, click **OK** to select the WSDL and return to the external resource wizard.

8. You can add more WSDL files to the resource wizard, if desired. To remove an existing resource, select it and click **Remove**.

9. Once all of the desired WSDLs have been added, click **Next**.

The **Select a Component** dialog is displayed.


Create a new External Resource

**Select a component**

Choose where to place your new External Resource in the Project. You may create it as a new Component or add to an existing one.

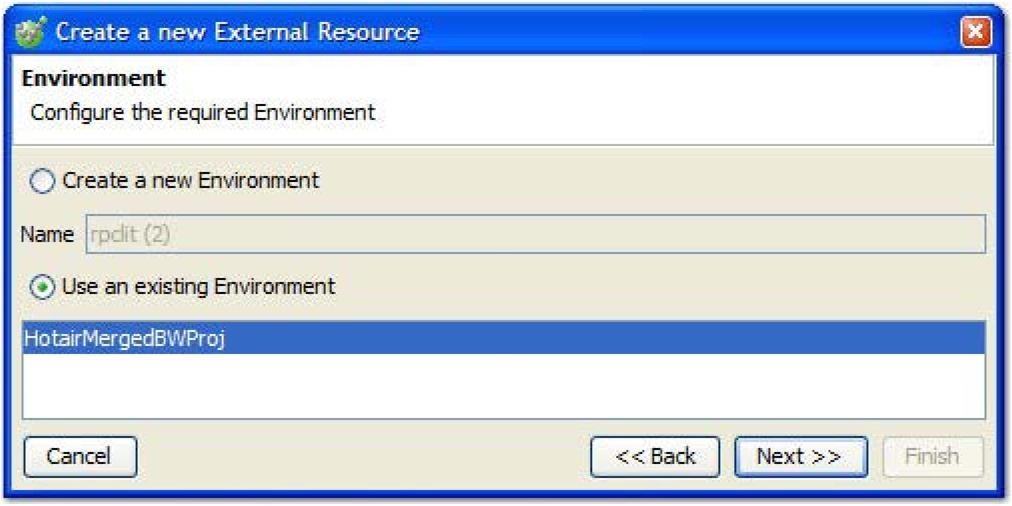
Name: RPC Literal WSDL

● Create a new Component

○ Select an existing Component

10. Select the desired option for adding the new resource to a project component – add it to a new component with a user-defined name, or add it to an existing component – then click **Next**.

11. If at least one environment already exists in the project, the **Environment** dialog is displayed. If no environments exist, this dialog is skipped and a new environment is created automatically, named according to WSDL file name.


Create a new External Resource

**Environment**

Configure the required Environment

○ Create a new Environment

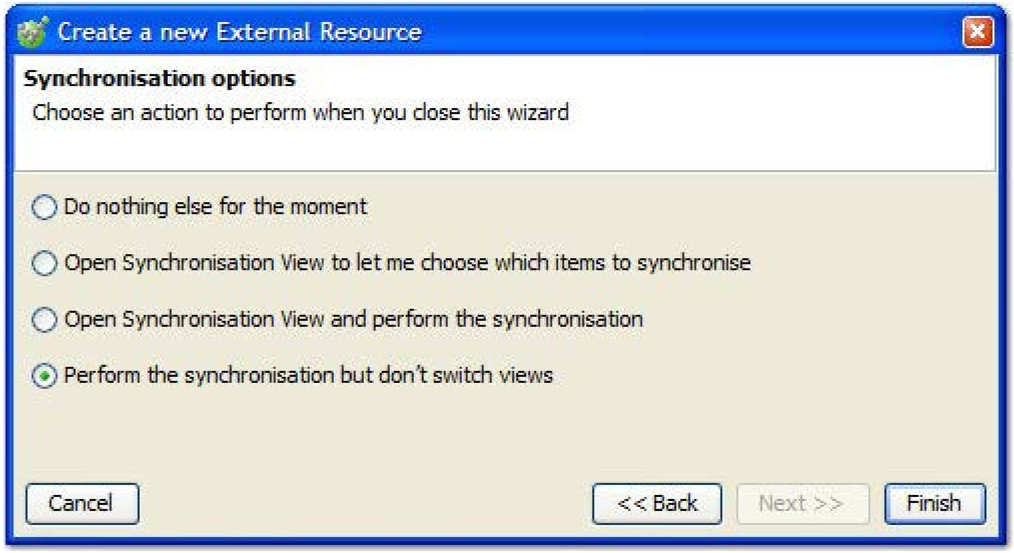
Name: rpclit (2)

● Use an existing Environment

HotairMergedBWProj

12. Select the desired option for your environment – create a new environment with a user-defined name, or use an existing environment – then click **Next**.

13. The **Synchronisation Options** dialog is displayed.



Create a new External Resource

**Synchronisation options**

Choose an action to perform when you close this wizard

○ Do nothing else for the moment

○ Open Synchronisation View to let me choose which items to synchronise

○ Open Synchronisation View and perform the synchronisation

● Perform the synchronisation but don't switch views

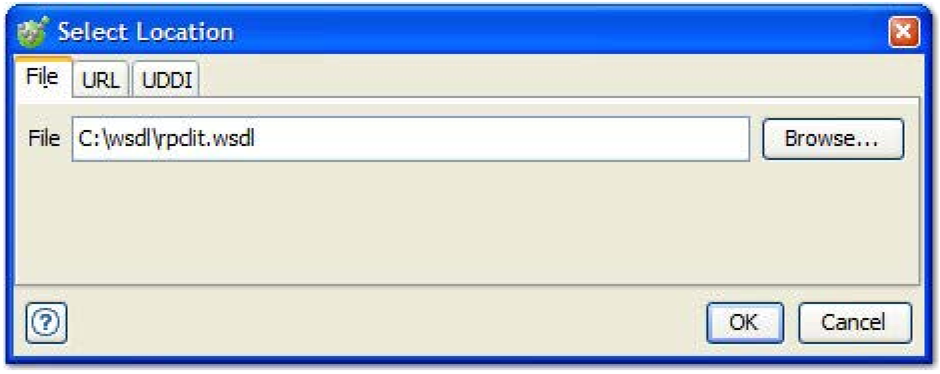
14. Select the desired synchronisation option, then click **Finish**.

**NOTE:** You should synchronize the WSDL to ensure that all of the required test artifacts are created in the Rational Integration Tester project.

The WSDL is now a component in your project and it can also be used as a schema to apply to other messages in the project.

### 3.2.1 Add a WSDL from File

When adding a WSDL to a Rational Integration Tester project, you can add a single document by browsing to it under the **File** tab of the **Select Location** dialog.

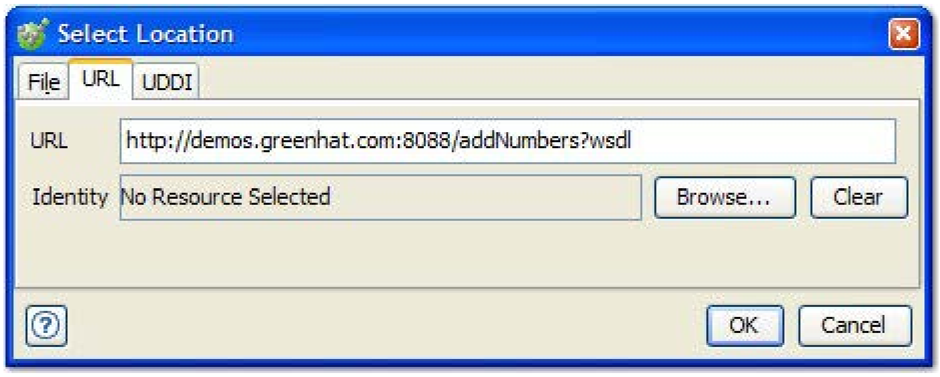


File: C:\wsdl\rpclit.wsdl

1. Click **Browse** to locate and select the WSDL document you want to import.
2. Click **OK** to return to the WSDL wizard.

### 3.2.2 Add a WSDL By Means of a URL

When adding a WSDL to a Rational Integration Tester project, you can add a single document by entering or pasting its URL under the **URL** tab of the **Select Location** dialog.



URL http://demos.greenhat.com:8088/addNumbers?wsdl

Identity No Resource Selected

1. In the URL field, enter or paste the URL of the WSDL document you want to import – SSL connections are supported.

2. If using SSL (https), click **Browse** to select an Identity that has already been configured in your project. For more information about this, refer to *IBM Rational Integration Tester Reference Guide.*

   **NOTE:** In the **Identity Selection** dialog that is displayed, only Certificate/ Private Key identities are available for selection.

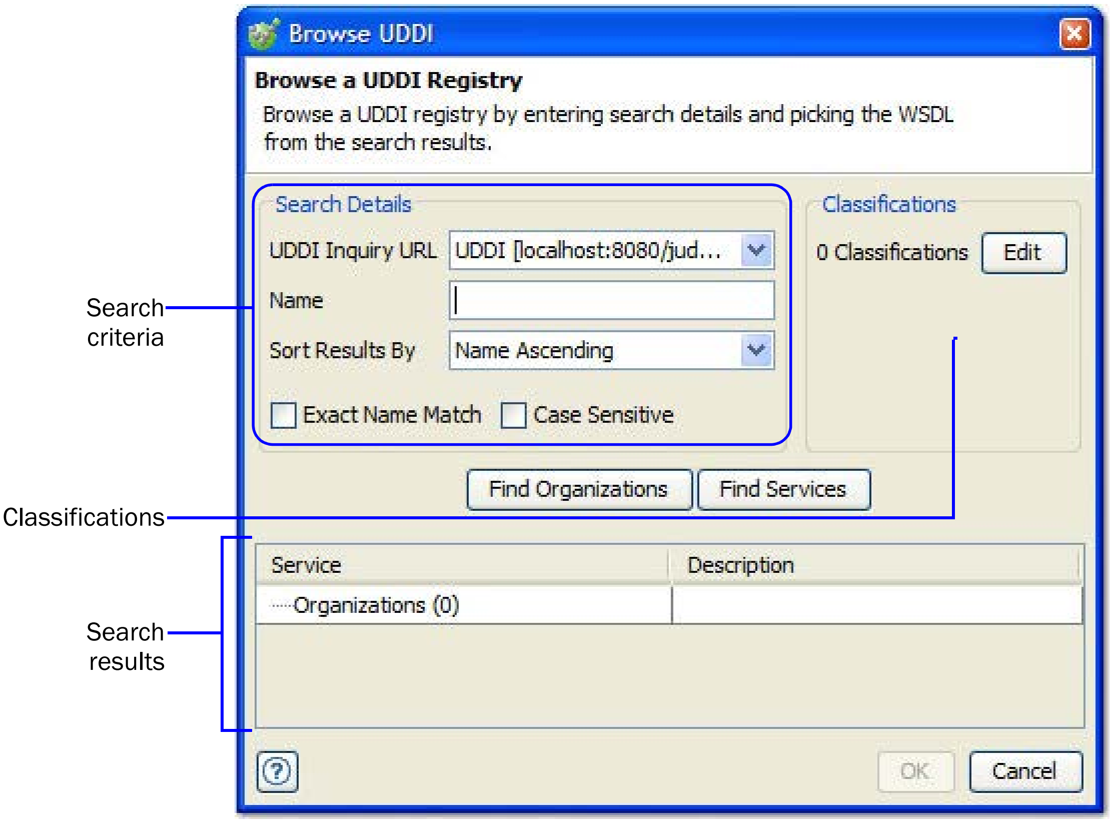
3. Click **OK** to return to the WSDL wizard.

### 3.2.3 Add a WSDL By Means of UDDI

When adding a WSDL to a Rational Integration Tester project, you can search a configured UDDI registry under the **UDDI** tab of the **Select Location** dialog.

The UDDI browser lets users search for web services (WSDL documents) in a specified UDDI registry. Users can search by service or organization name, then browse the search results and select the desired WSDL document.

The main UDDI Browser is shown below.



Browse UDDI

**Browse a UDDI Registry**

Browse a UDDI registry by entering search details and picking the WSDL from the search results.

Search Details

UDDI Inquiry URL: UDDI [localhost:8080/jud...

Name:

Sort Results By: Name Ascending

□ Exact Name Match □ Case Sensitive

Classifications

0 Classifications

| Service | Description |
|---|---|
| Organizations (0) | |

To search for matching organizations, click **Find Organizations**. To search for matching services, click **Find Services**.

See the following sections for more information about the browser:

- Search Criteria
- Classifications
- Search Results

### Search Criteria

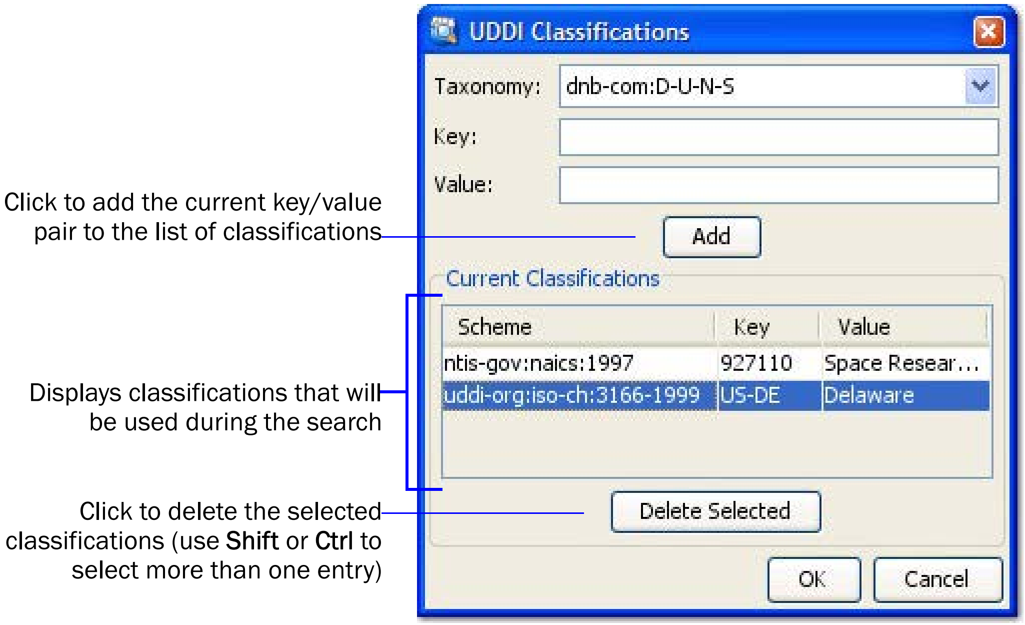
The search criteria is comprised of the following fields:

| Field | Description |
|---|---|
| UDDI Inquiry URL | Select the inquiry URL (physical UDDI server) of the registry to search. |
| Name | The organization or service name for which to search. |
| Sort Results | Select the desired sort method for the search results.<br>**NOTE:** The UDDI webservice specification does not provide date information in its response. Therefore, while you can sort by date, no date field is displayed in the results since it is not available. |
| Exact Name Match | Enable this option to only return results that exactly match the search string (that is, **Name**). |
| Case Sensitive | Enable this option to perform a case sensitive search. |

### Classifications

UDDI classifications represent a taxonomy that provides useful values for categorizing the technical information of web services. Classifications associate keywords with an entry, especially those that are not part of the name of the entry.

To add or remove classifications from the search, click the **Edit** button under Classifications. The UDDI Classifications window is displayed, as shown below.



Taxonomy: dnb-com:D-U-N-S

Key:

Value:

| Scheme | Key | Value |
|---|---|---|
| ntis-gov:naics:1997 | 927110 | Space Resear... |
| uddi-org:iso-ch:3166-1999 | US-DE | Delaware |

Click to add the current key/value pair to the list of classifications

Displays classifications that will be used during the search

Click to delete the selected classifications (use **Shift** or **Ctrl** to select more than one entry)

The classifications dialog contains the following fields:

| Field | Description |
|---|---|
| Taxonomy | The classification scheme to use (see below) |
| Key | The category identifier for the selected taxonomy |
| Value | The category name corresponding to **Key** |

The Rational Integration Tester UDDI browser supports the following taxonomies:

| Selection | Description |
|---|---|
| dnb-com:D-U-N-S | Dun & Bradstreet D-U-N-S® Number Identifier System |
| ntis-gov:naics:1997 | North American Industry Classification System (NAICS) 1997 Release |
| thomasregister-com:supplierID | Thomas Register Supplier Identifier Code System |
| uddi-org:general_keywords | UDDI v2 General Keywords Category System |
| uddi-org:iso-ch:3166-1999 | ISO 3166 Geographic Code System |
| uddi-org:isReplacedBy | UDDI v2 IsReplacedBy Identifier System |
| uddi-org:operators | UDDI v2 Operators Category System |
| uddi-org:owningBusiness | UDDI v2 OwningBusiness Category System |
| uddi-org:relationships | UDDI v2 Relationships Category System |
| uddi-org:types | UDDI v2 Types Category System |
| unspsc-org:unspsc | United Nations Standard Products and Services Code System (UNSPSC) Version 6.0501 |
| unspsc-org:unspsc:3-1 | United Nations Standard Products and Services Code System (UNSPSC) Version 3.1 |

- To add a classification using the selected taxonomy, enter the key and value and click **Add**.
- To edit an existing classification, double-click the Key or Value entry to activate the field for modifications.
- To remove one or more existing classifications, select the desired entries and click **Delete Selected**.

- To save changes to the classifications and return to the UDDI browser, click **OK**.
- To return to the browser without making any changes, click **Cancel**.

### Search Results

The results of your search are displayed in the bottom portion of the UDDI browser, according to the search criteria and the search type (organization or service). Results are displayed in a standard tree format that uses the following hierarchy:
**Organization > Service Name > Service Bindings > WSDL Documents**.

**NOTE:** When searching for services, the higher-level organization is not displayed.

Each node of the results tree can be expanded to show the items it contains.

**NOTE:** When searching for organizations, you can right-click an organization name to display additional properties or to show related organizations. All related organizations (that is, those containing the selected organization and those contained by the selected organization) are displayed.

After locating the desired WSDL document, select it and click **OK** to return to the WSDL wizard.

## 3.3 Structure of a WSDL Message

The WSDL schema defines operations and their associated XML elements, which define the payloads of the operations' messages. Each operation can have input (request) and output (response) messages. When Rational Integration Tester processes the WSDL file, it creates a schema constrained structure for each of these messages. The basic structure for these messages is similar for all of the WSDL messages and is explained using examples from the UK Lottery service provided by 4duk.net and the XMethods Demo Currency Converter WSDL.

### 3.3.1 The Root Message and SOAP Envelope

The root level message represents the SOAP Envelope and is defined by the operation that is being invoked. The name of the root message consists of three parts; the name of the operation, the direction of the SOAP message (input or output) and the name of the actual WSDL message being created. This name is in the format *<Operation_Name>*__*<Direction>*__*<SOAP Message Name>*. For example, a WSDL defines an operation called LOTTO_Num_Single_Hist that has an output SOAP message called LOTTO_Num_Single_HistResponse. When this message is selected, the root message will have the following name:

```
LOTTO_Num_Single_Hist__OUTPUT__LOTTO_Num_Single_HistResponse
```

The children of the root message fall into two categories; any number of namespace fields and the SOAP Body element.

### 3.3.2 SOAP Envelope Namespace Attributes

The WSDL definition can define any number of namespace attributes that are assigned to its SOAP Envelopes. These namespaces fall into two categories; the standard SOAP namespaces that appear on all SOAP Envelopes, and instance specific namespaces. When you select a SOAP Envelope root message only the instance specific namespace attributes will be created. The standard namespaces are assumed and will be added to the XML payload by the formatters during message compilation or removed during message de-compilation.

### 3.3.3 Overview of the SOAP Body Element

The SOAP Envelope root message can only have one child message. This message represents the contents of the SOAP Body element and is defined by the WSDL file's message definition that corresponds to the selected SOAP Envelope. The exact structure of the root message within Rational Integration Tester depends on whether the SOAP Envelope defines a Document Literal or an RPC Encoded message.

### 3.3.4 Document Literal Body Elements

Below is an extract from a WSDL schema definition taken from the 4duk lottery service WSDL. This extract contains the operation definition and the input message definition for an operation called LOTTO_NumCheck_SingleSet_HistRequest and is defined as part of a Document Literal WSDL definition.

```
<definitions xmlns:ns0=http://www.4d.com/namespace/default>
<operation name="LOTTO_NumCheck_SingleSet_Hist">
<input message="tns:LOTTO_NumCheck_SingleSet_HistRequest"/>
<output message="tns:LOTTO_NumCheck_SingleSet_HistResponse"/>
</operation>
<message name="LOTTO_NumCheck_SingleSet_HistRequest">
<part name="FourD_ArgIn"
element="tns:LOTTO_NumCheck_SingleSet_Hist"/>
</message>
</defintions>
```

As this WSDL defines Document Literal style SOAP Envelopes, then Rational Integration Tester will create corresponding Document Literal style SOAP Body messages.

Document Literal WSDL message elements have only one child part element. This part element describes the children of the SOAP Body Element. The part element generally contains an element attribute. This attribute is a reference to an XML element defined within the WSDL's XML Schema definition. In the above example, the part refers to the tns:LOTTO_NumCheck_SingleSet_Hist element. This reference indicates the schema defined XML element that the SOAP Body's message structure will be created from. Within Rational Integration Tester, the creation of the SOAP Body's child structure will continue in an identical manner to that of normal XML Schema constrained message population found elsewhere within the tool (for example, when using the XML Message Text Formatter).

The name of the SOAP Body message is set as the name of the element referred to by the element attribute and will include any required namespace prefixes. In the above example this will be `LOTTO_NumCheck_SingleSet_Hist.`

The LOTTO_NumCheck_SingleSet_Hist element is defined within the XML Schema as having six child elements with a text child of type integer and one element with a Boolean field.

### 3.3.5 RPC Encoded Body Elements

Below is an extract taken from the XMethods Demo Currency Converter WSDL. This extract contains the operation definition and the input message for an operation called getRate and is defined as part of an RPC Encoded WSDL definition.

```
<definitions
xmlns:tns=http://www.xmethods.net/sd/CurrencyExchangeService.wsdl>
<operation name="getRate">
<input message="tns:getRateRequest"/>
<output message="tns:getRateResponse"/>
</operation>
<message name="getRateRequest">
<part name="country1" type="xsd:string"/>
<part name="country2" type="xsd:string"/>
</message>
</defintions>
```

An RPC Encoded WSDL message definition can have any number of parts. These parts define the child elements of the SOAP Body element. Each part has a name and a type attribute. The type attribute can either contain a simple XML Schema type (integer, string, decimal, and so on) or a reference to the WSDL's XML Schema definition.

If the type references a scalar type then the resultant Rational Integration Tester message will be created as an XML element message with a child of the specified type. The name of the message will be that provided by the part's name attribute.

If the type references the XML Schema definition then the resultant Rational Integration Tester message will be created according to the referenced XML Schema object, with its name being the name of that XML Schema object.

If the whole message represents an input (request) message then the name of the SOAP Body Element is set as <Operation Name>.

If the message represents an output (response) message then the name of the SOAP Body Element is set as the <Operation name>Response.

The names are then qualified using any applicable namespace prefixes. For example, for the Demo Currency Converter the name of the request message's SOAP Body Element is set as tns:getRate and the response is named tns:getRateResponse.

## 3.4 WSDL Message to XML Document Conversion

The Document Literal and RPC Encoded styles define two different mechanisms for constructing the XML document that makes up the body of a SOAP Envelope. When the Document Literal style is used, the XML document that is created will be in a tree structure that resembles a common XML document. The RPC Encoded style creates a flat XML document with cross-references from an element to its children.

The Document-Literal and RPC-Encoded message formatters handle the two styles within Rational Integration Tester. During the message compilation process that occurs during publishing, these formatters take the Rational Integration Tester messages created using the WSDL schemas. The format is specified in section 6.1, and this is used to create the required XML document structure. During the de-compilation process that occurs within a subscriber, they take the received XML Document and create a Rational Integration Tester message structure that adheres to the formats specified in section 6.1.

### 3.4.1 Document Literal Message to XML Conversion

The Document-Literal formatter takes a WSDL based Rational Integration Tester Message and creates an XML Document that represents a SOAP Envelope based on the Rational Integration Tester message. The root message represents the SOAP Envelope. The formatter will create an envelope element that contains all of the instance specific namespaces as well as the standard SOAP namespace definitions (see SOAP Envelope Namespace Attributes). A SOAP Body element is created as a child of the SOAP Envelope. Within the output XML the SOAP Body contains the XML data that is to be sent.

The only child message of the root represents the contents of the SOAP Body element. This message and all of its children are converted to XML in the same manner as XML is created within the XML Text Message Formatter. The XML generated for this message is added as the child of the SOAP Body.

Below is the XML output generated by the Document Literal formatter for the example message provided in section 6.1.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:tns="http://www.4d.com/namespace/default">
<soapenv:Body>
<tns:LOTTO_NumCheck_SingleSet_Hist xsi:type="xsd:any">
<tns:inN1 xsi:type="xsd:int">4</tns:inN1>
<tns:inN2 xsi:type="xsd:int">6</tns:inN2>
<tns:inN3 xsi:type="xsd:int">13</tns:inN3>
<tns:inN4 xsi:type="xsd:int">23</tns:inN4>
<tns:inN5 xsi:type="xsd:int">34</tns:inN5>
<tns:inN6 xsi:type="xsd:int">45</tns:inN6>
<tns:inHistoric xsi:type="xsd:boolean">
true
</tns:inHistoric>
</tns:LOTTO_NumCheck_SingleSet_Hist>
</soapenv:Body>
</soapenv:Envelope>
```

### 3.4.2 RPC Encoded Message to XML Conversion

The RPC-Encoded Formatter takes a WSDL based Rational Integration Tester message and creates an XML Document that represents a SOAP Envelope based on the Rational Integration Tester message. The first few levels of the XML are created in the same style as the Document-Literal Formatter. The root message represents the SOAP Envelope. The formatter will create an envelope element that contains all of the instance specific namespaces as well as the standard SOAP namespace definitions (see SOAP Envelope Namespace Attributes). A SOAP Body element is created as a child of the SOAP Envelope. Within the output XML the SOAP Body contains the XML data that is to be sent.

The only child message (data root) of the root represents the contents of the SOAP Body. Unlike the Document literal style where the XML document is created in a tree structure, the RPC Encoded style creates XML that is in a flat structure with elements containing references to their children.

The data root has an element created as a child of the SOAP Body element that contains the name of the data root. For each child of the data root, a child element is added to the data root element that has the name of the child node and an attribute called href that contains a reference id. For each of the child messages a multiref element is also created that is added as a child of the SOAP Body element. This multiref element has an id attribute that is used for linking an element to its children. This multiref element can then either contain textual content or child elements with href attributes that link the multiref element to its child elements. Namespace information and other encoding information are also added to each multiref element.

Below is the XML output generated by the RPC Encoded formatter for the example message provided in section 6.1.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="urn:xmethods-CurrencyExchange">
<soapenv:Body>
<ns0:getRate
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<country2 href="#id0"/>
<country1 href="#id1"/>
</ns0:getRate>
<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
xsi:type="xsd:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
united kingdom
</multiRef>
<multiRef id="id1" soapenc:root="0"
soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
xsi:type="xsd:string"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
united states
</multiRef>
</soapenv:Body>
</soapenv:Envelope>
```

## 3.5 Operation Header Properties

Each operation contained within the WSDL defines a set of header properties that can be sent with the message. During the processing of the WSDL these properties are extracted from the operation and when an operation message is selected these properties are automatically added to the header properties of the Rational Integration Tester message. These properties generally consist of the Content-type of the message (usually xml/text) and the SOAP Action.

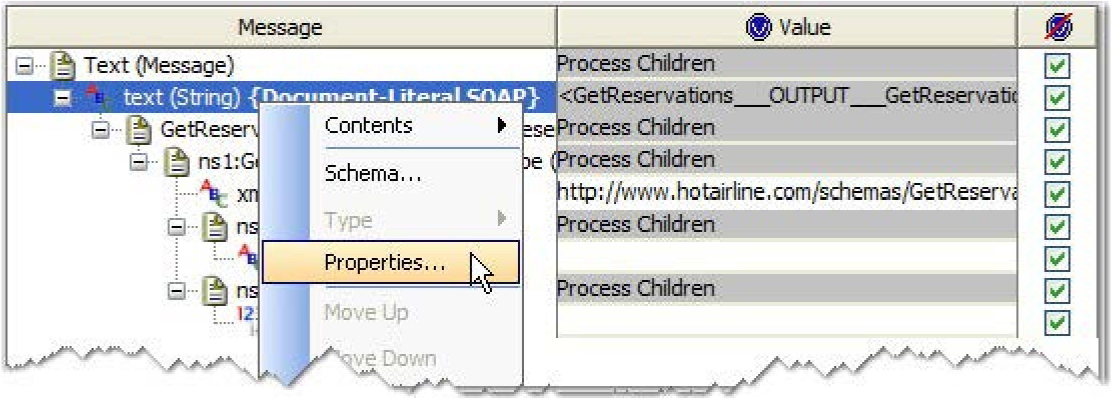
## 3.6 SOAP Faults

When a web service request generates an error within the server, the server will generally send a reply message back that contains a SOAP Fault element instead of a SOAP Body element. This SOAP Fault is defined within the SOAP Envelope XML schema definition. When a SOAP Fault is received, both the RPC-Encoded and the Document-Literal Formatters will generate a Rational Integration Tester message that has the same structure. They will both extract the SOAP Fault element and generate a standard XML Message structure in the same manner as the XML Text Message Formatter. Since SOAP Faults are not defined within the WSDL file they are handled by means of the standard XML Schema process.

Within Rational Integration Tester an XSD Schema can be defined that references a SOAP Envelope XML Schema definition. There are several of these definitions and they can vary from one to another. Select the SOAP Envelope that contains the SOAP Fault definition that is expected. If it is expected that a SOAP Fault is to be returned from a request then within the receive request select the desired SOAP Envelope XSD as the schema for the message. Next, select the Fault root and build the expected structure accordingly with whatever validation is required.

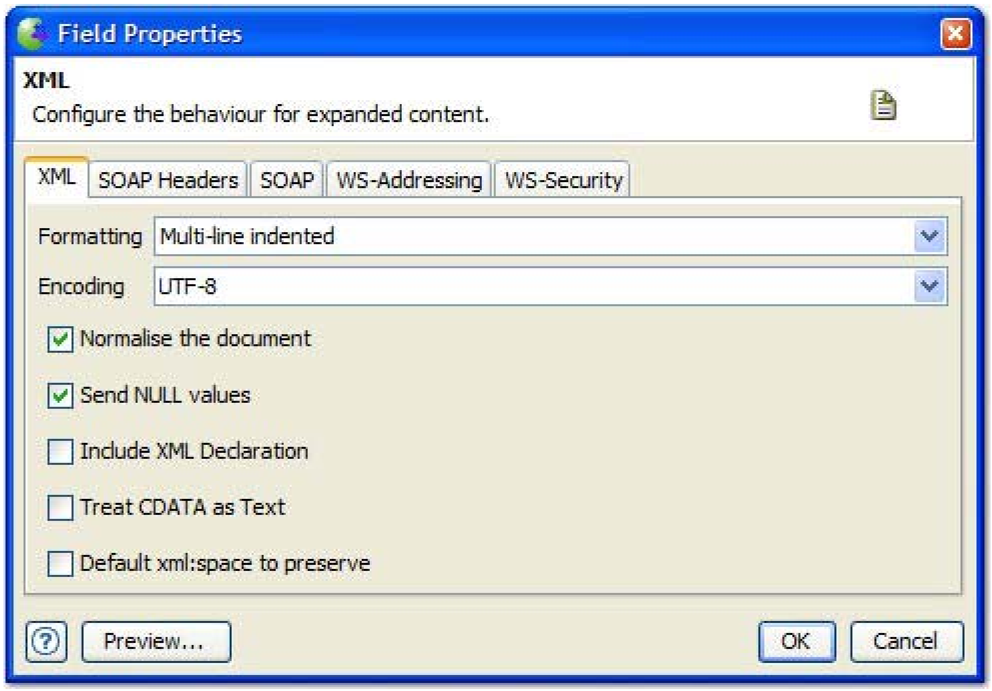
## 3.7 XML and SOAP Message Properties

Message properties in Rational Integration Tester let you configure how message content is treated in test actions (for example, Publish, Subscribe, and so on). To view or modify message properties, right-click on the root of the SOAP message in the requirement or in the body of the message editor.



| Message | Value | |
|---|---|---|
| Text (Message) | Process Children | ✓ |
| text (String) {Document-Literal SOAP} | <GetReservations___OUTPUT___GetReservatic | ✓ |
| GetReserv… ese | Process Children | ✓ |
| ns1:G… pe ( | Process Children | ✓ |
| xn | http://www.hotairline.com/schemas/GetReserva | ✓ |
| ns | Process Children | ✓ |
| | | ✓ |
| ns | Process Children | ✓ |
| 12 | | ✓ |

Use the **Field Properties** dialog to view or edit the message properties.



**XML**

Configure the behaviour for expanded content.

Formatting: Multi-line indented

Encoding: UTF-8

☑ Normalise the document

☑ Send NULL values

☐ Include XML Declaration

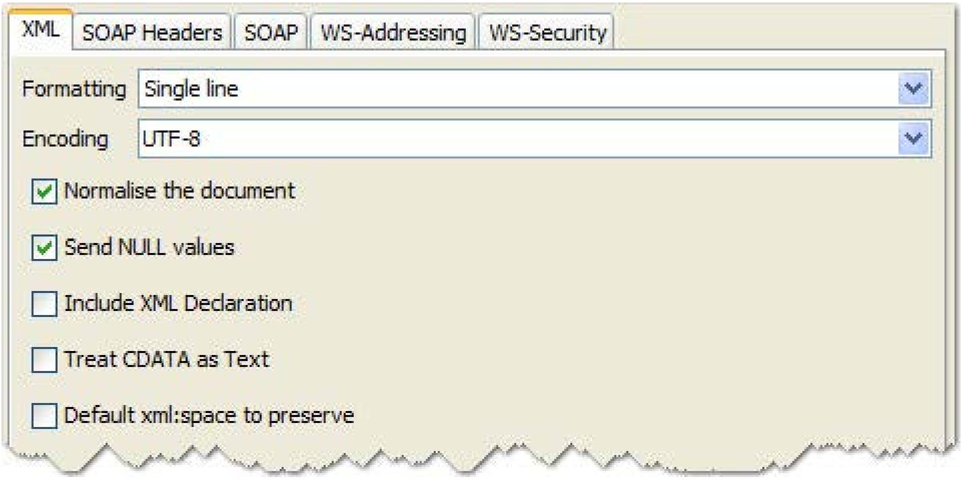
☐ Treat CDATA as Text

☐ Default xml:space to preserve

**NOTE:** Click **Preview** at any time to view the message as it will be applied using the properties that you have modified.

## 3.7.1 XML Properties

The **XML** tab lets you configure how XML content in the message should be handled.



Formatting: Single line

Encoding: UTF-8

☑ Normalise the document

☑ Send NULL values

□ Include XML Declaration

□ Treat CDATA as Text

□ Default xml:space to preserve

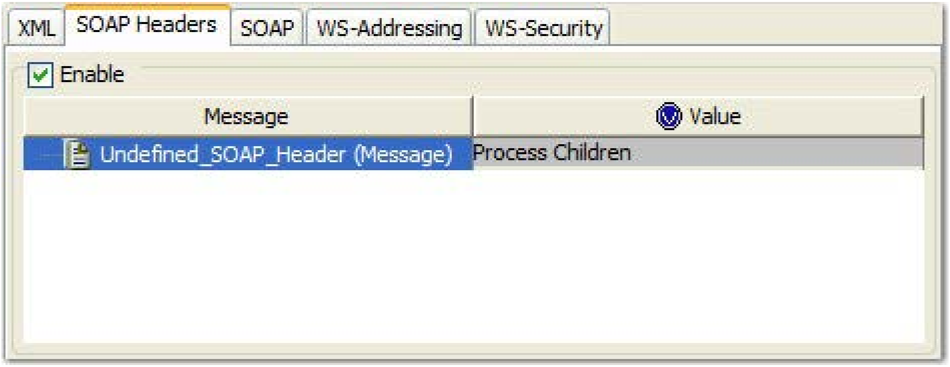
The options available for handling XML are described below:

| | |
|---|---|
| Formatting | Select how XML should be formatted, either Single-line or Multi-line indented. |
| Encoding | Select the encoding to use, either UTF-8 or UTF-16. |
| Normalize empty text nodes | If enabled, extra spaces will be removed from the XML. |
| Send NULL values | If enabled, null can be sent for empty fields. |
| Include XML Declaration | Enable this option to force the inclusion of the XML declaration at the beginning of the message. |
| Treat CDATA as Text | Enable this option to treat CDATA fields in the XML as text. |
| Default xml:space to preserve | Enables or disables the preservation of white space in XML. |

**NOTE:** The default values for the XML options can be modified by means of the Rational Integration Tester preferences (**Window > Preferences**, then select the XML option).

## 3.7.2 SOAP Headers

The SOAP Headers tab lets you enable or disable the sending of SOAP headers with the message, and displays any headers that are included in the current message.



☑ Enable

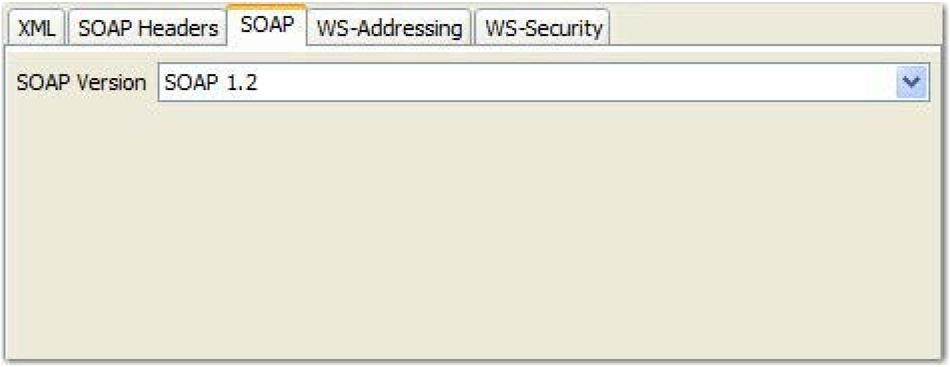
| Message | Value |
|---|---|
| Undefined_SOAP_Header (Message) | Process Children |

Tick or untick the **Enable** option to toggle whether or not SOAP headers should be included.

**NOTE:** The default setting for SOAP headers can be specified in the Rational Integration Tester preferences (**Window > Preferences**, then select the SOAP option).

## 3.7.3 SOAP

The SOAP tab lets you specify which version of SOAP (1.1 or 1.2) to apply when sending the message.



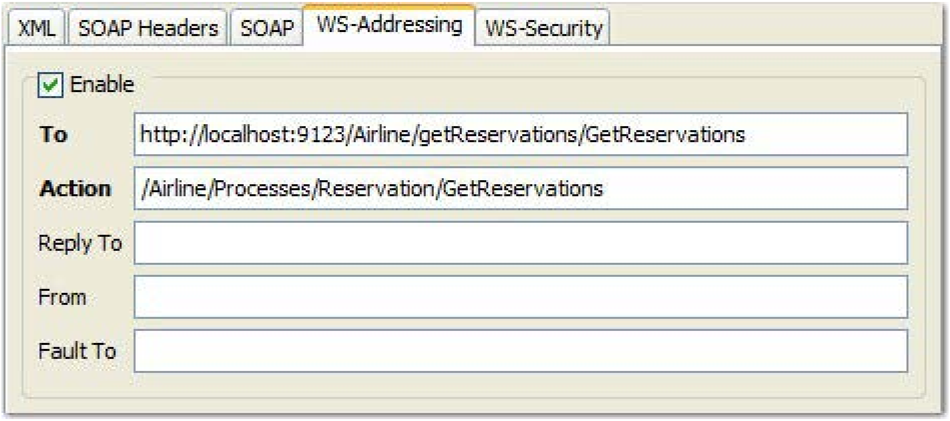
SOAP Version: SOAP 1.2

Select the version to use from the **SOAP Version** dropdown menu.

**NOTE:** The default version can be specified in the Rational Integration Tester preferences (**Window > Preferences**, then select the SOAP option).

## 3.7.4 WS-Addressing

The WS-Addressing tab lets you enable or disable message constructs (that is, endpoints and information headers) to be sent with the SOAP message. You can also modify the WS-Addressing properties to be sent.



☑ Enable

**To** http://localhost:9123/Airline/getReservations/GetReservations

**Action** /Airline/Processes/Reservation/GetReservations

Reply To

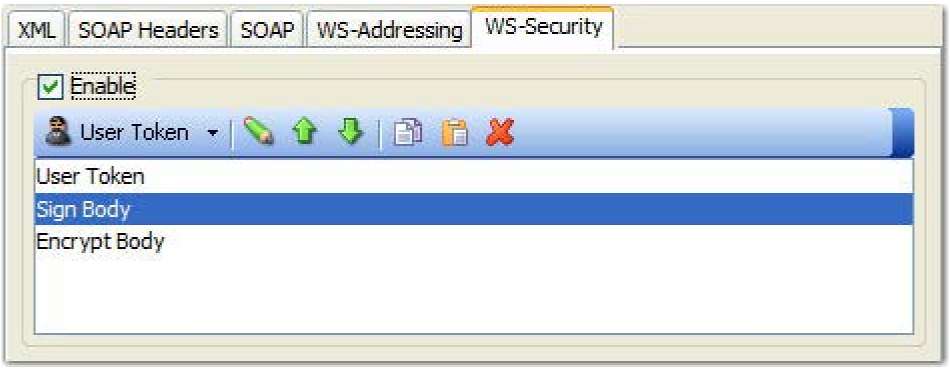
From

Fault To

**NOTE:** The sending of WS-Addressing constructs can be enabled or disabled by default in the Rational Integration Tester preferences (**Window > Preferences**, then select the WS-* Extensions option).

## 3.7.5 WS-Security

The WS-Security tab lets you enable or disable Web Services security actions to be sent with the SOAP message. Details about creating and configuring the available actions can be found in Web Services Security Actions.



☑ Enable

User Token

Sign Body

Encrypt Body

**NOTE:** The sending of WS-Security actions can be enabled or disabled by default in the Rational Integration Tester preferences (**Window > Preferences**, then select the WS-* Extensions option).

# Web Services Security Actions

Contents

This chapter provides information the various Web Services Security (WSS) actions that can be applied to published SOAP messages in Rational Integration Tester. Details about creating and configuring each action are also provided.

## 4.1 Overview

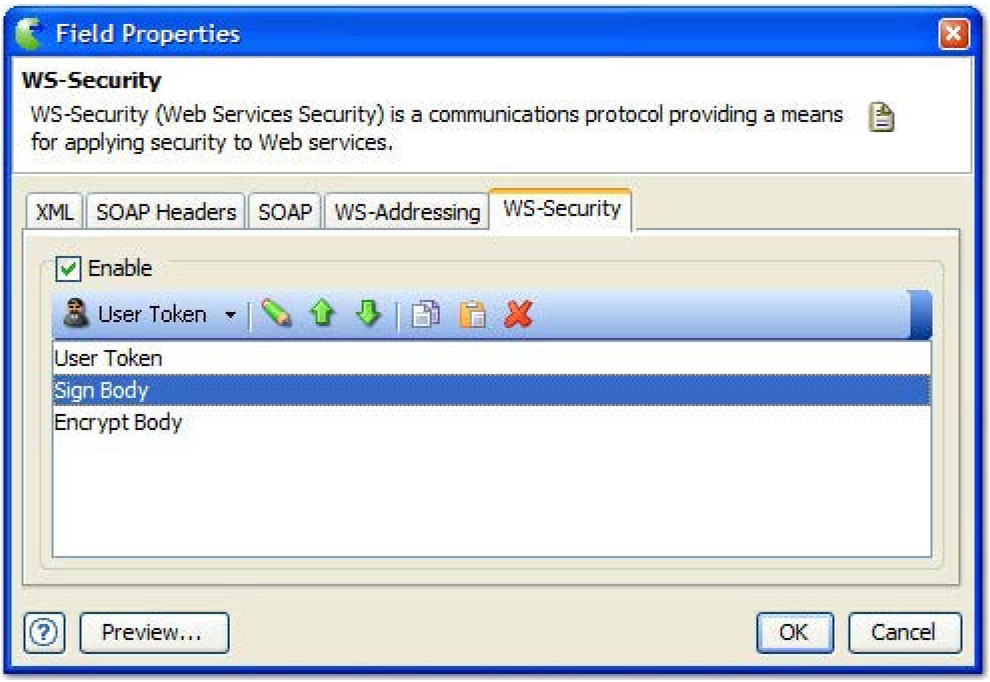
Rational Integration Tester supports the use of one or more security actions that can be layered on top of one another in an outgoing SOAP message, as follows:

- **User Tokens** add simple user name and password authentication (see User Tokens).
- **Timestamp Tokens** let users define a period of time for during which the SOAP envelope is valid (see Timestamp Tokens).
- **Binary Tokens** add authentication using a keystore and certificate alias (see Binary Tokens).
- **Digital Signatures** can be applied to the body of a SOAP message (see Signatures).
- **Encryption** can be applied to the body of a SOAP message (see Encryption).

**NOTE:** Security actions are not supported for incoming messages (for example, subscribers).

## 4.2 Creating Security Actions

Security actions are created and modified under the **WS-Security** tab of the Field Properties dialog, which is opened when viewing the properties of a SOAP message (see XML and SOAP Message Properties). One or more security actions can be created, and the inclusion of those actions can be enabled or disabled by ticking or unticking the **Enabled** option at the top of the dialog.



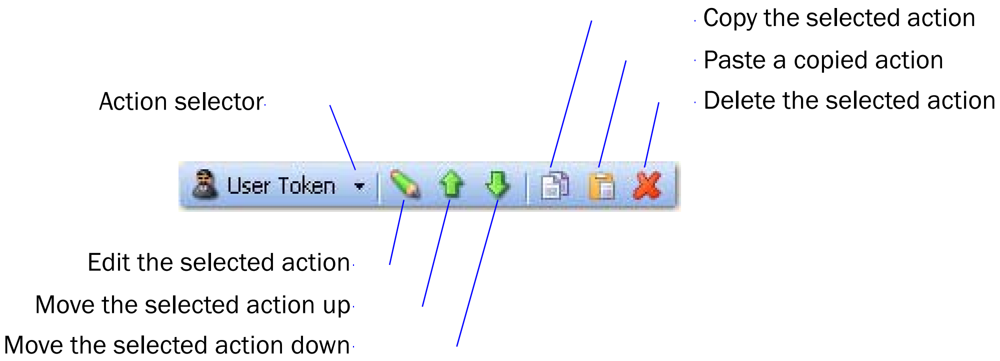
**WS-Security**

WS-Security (Web Services Security) is a communications protocol providing a means for applying security to Web services.

☑ Enable

User Token
Sign Body
Encrypt Body

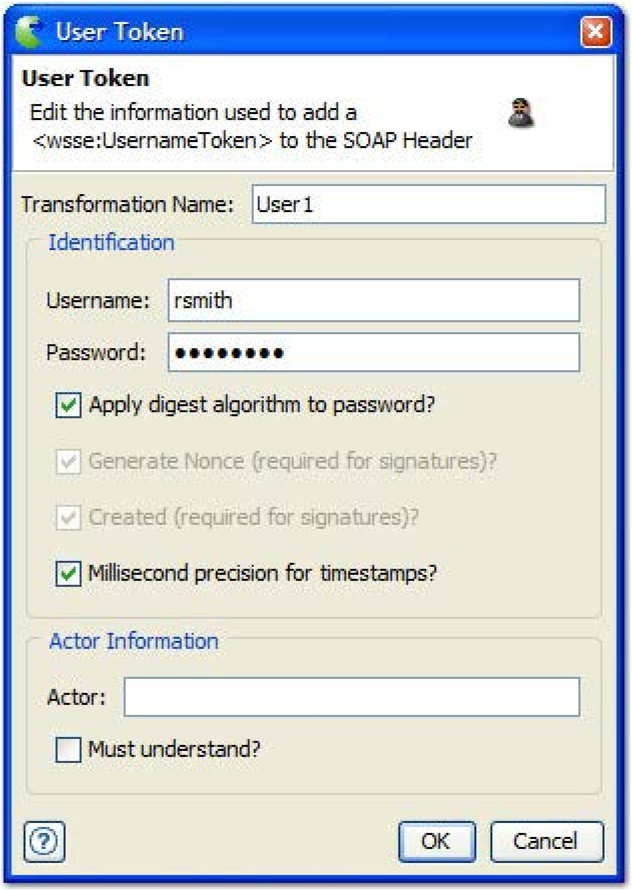
Security actions can be managed using the toolbar at the top of the actions window:



Copy the selected action
Paste a copied action
Delete the selected action
Action selector
Edit the selected action
Move the selected action up
Move the selected action down

### 4.2.1 User Tokens

To add a user token, click the action selector and choose **User Token**. The **User Token Editor** is displayed.



User Token

**User Token**

Edit the information used to add a <wsse:UsernameToken> to the SOAP Header

Transformation Name: User1

Identification

Username: rsmith

Password: ●●●●●●●●●

☑ Apply digest algorithm to password?

☑ Generate Nonce (required for signatures)?

☑ Created (required for signatures)?

☑ Millisecond precision for timestamps?

Actor Information

Actor:

□ Must understand?

A user token adds a **<wsse:UsernameToken>** element to the SOAP header. At a minimum, the user token defines a user name and password to authenticate the SOAP message.

Additional (optional) steps can be taken to further secure the token and the message (that is, **password digest**, **nonce**, **created**, **actor/role**, and **mustUnderstand**).

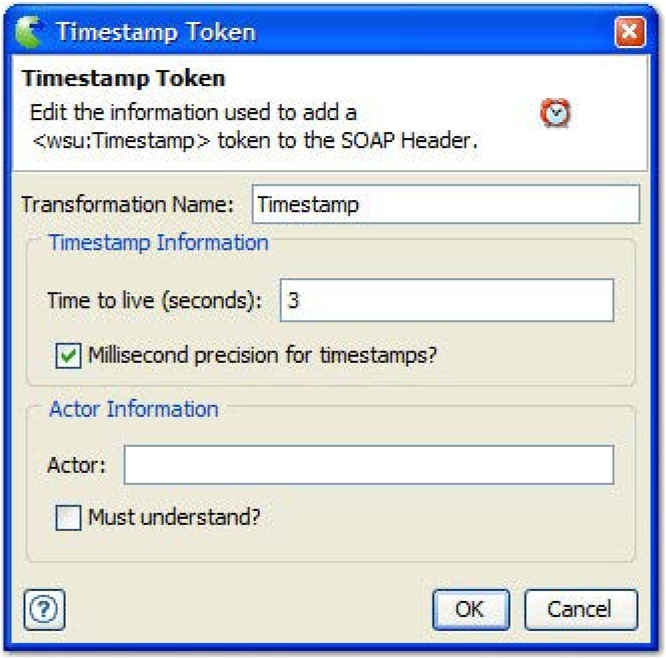
**NOTE:** If you want to sign a SOAP body with a User Token, the token must include the **Nonce** and **Created** elements.

The following fields and options are part of the user token:

| Field | Description |
|---|---|
| Transformation Name (**Required**) | User-defined name for the security action (helps identify the action in the main list). |
| Username (**Required**) | User name to include with the token. |
| Password (**Required**) | Password for the corresponding user name. |
| Apply digest algorithm to password | If enabled, encrypts the clear-text password using the nonce and created elements. |
| Generate Nonce | A random number that makes the message more unique, designed to prevent relay attacks (that is, by replaying recorded SOAP packets). This option is enabled automatically if a digest is applied to the password. |
| Created | Adds a timestamp to the outgoing message. This option is enabled automatically if a digest is applied to the password. |
| Millisecond precision for timestamps | If enabled, the timestamp for the <wsu:Created> element will be generated using milliseconds (2009-03-31T02:41:16.817Z). If disabled, the timestamp will be generated without milliseconds (2009-03-31T02:41:16Z). |
| Actor/Role | Indicates a specific message receiver (either the ultimate receiver or an intermediary). For each actor/role that is defined (that is, in multiple tokens), a separate security header is added to the SOAP header. |
| Must understand | If enabled, makes the SOAP header mandatory for the specified actor/role. In this case, either the header block must be processed or the entire SOAP message should be ignored, and a SOAP fault should be generated.<br>If not enabled, the specified actor/role may or may not process the SOAP header |

### 4.2.2 Timestamp Tokens

To add a timestamp token, click the action selector and choose **Timestamp Token**. The **Timestamp Token Editor** is displayed.



**Timestamp Token**

Edit the information used to add a <wsu:Timestamp> token to the SOAP Header.

Transformation Name: Timestamp

Timestamp Information

Time to live (seconds): 3

☑ Millisecond precision for timestamps?

Actor Information

Actor:

□ Must understand?

**NOTE:** Timestamp tokens are defined in GMT.

A timestamp token adds a **<wsu:Timestamp>** element to the SOAP header, which defines a period of time during which the token (that is, the message) is valid. The "Created" element of the timestamp equals the date and time at which the message is sent. The "Expires" element is defined by the "Time to live" field in the token editor (that is, Created + Time to live = Expires).

Optionally, you can define Actor/Role information to further secure the token and the message (**actor/role** and **mustUnderstand**).

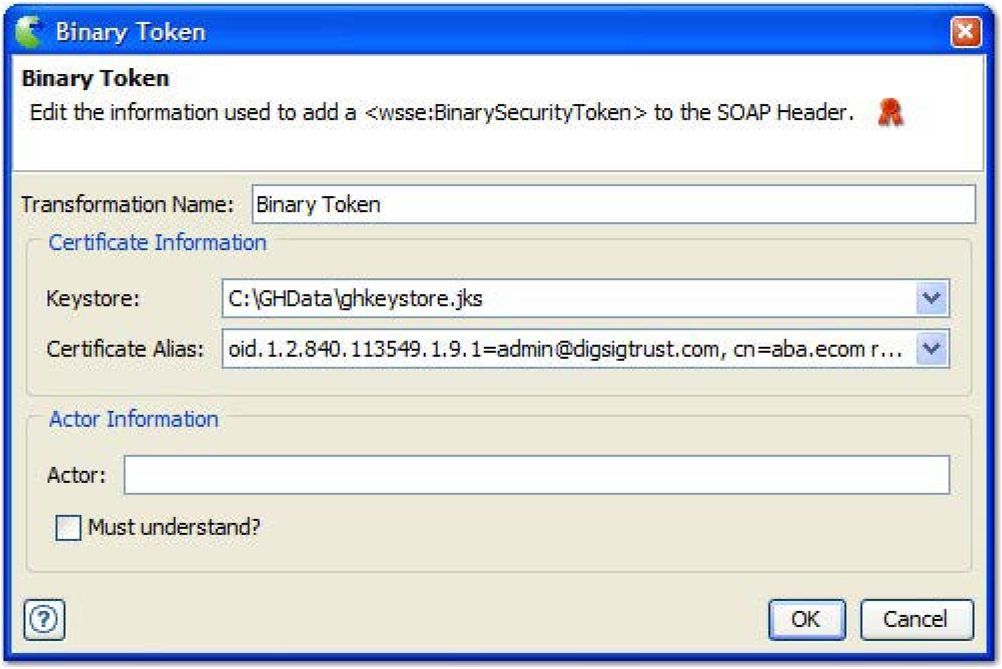
**NOTE:** If you do not modify the default value (zero) for **Time to live**, the timestamp token will never expire.

The following fields and options are part of the timestamp token:

| Field | Description |
|---|---|
| Transformation Name (**Required**) | User-defined name for the security action (helps identify the action in the main list). |
| Time to live (**Required**) | The amount of time (in seconds) for which the token should be valid after sending the message. |
| Millisecond precision for timestamps | If enabled, the timestamp for the <wsu:Created> element will be generated using milliseconds (2009-03-31T02:41:16.817Z). If disabled, the timestamp will be generated without milliseconds (2009-03-31T02:41:16Z). |
| Actor/Role | Indicates a specific message receiver (either the ultimate receiver or an intermediary). For each actor/role that is defined (that is, in multiple tokens), a separate security header is added to the SOAP header. |
| Must understand | If enabled, makes the SOAP header mandatory for the specified actor/role. In this case, either the header block must be processed or the entire SOAP message should be ignored, and a SOAP fault should be generated.<br>If not enabled, the specified actor/role may or may not process the SOAP header. |

### 4.2.3 Binary Tokens

To add a binary token, click the action selector and choose **Binary Token**. The **Binary Token Editor** is displayed.



Binary Token

**Binary Token**

Edit the information used to add a <wsse:BinarySecurityToken> to the SOAP Header.

Transformation Name: Binary Token

Certificate Information

Keystore: C:\GHData\ghkeystore.jks

Certificate Alias: oid.1.2.840.113549.1.9.1=admin@digsigtrust.com, cn=aba.ecom r...

Actor Information

Actor:

□ Must understand?

**NOTE:** If you have not defined an identity store in Architecture School, (refer to *IBM Rational Integration Tester Reference Guide*), you must create one before creating a binary token. You can only create one binary token for each certificate alias in the keystore.

A binary token adds a **<wsse:BinarySecurityToken>** element to the SOAP header. The binary token defines a keystore (Rational Integration Tester Identity Store) and public key alias to authenticate the SOAP message.

Optionally, you can define Actor/Role information to further secure the token and the message (**actor/role** and **mustUnderstand**).

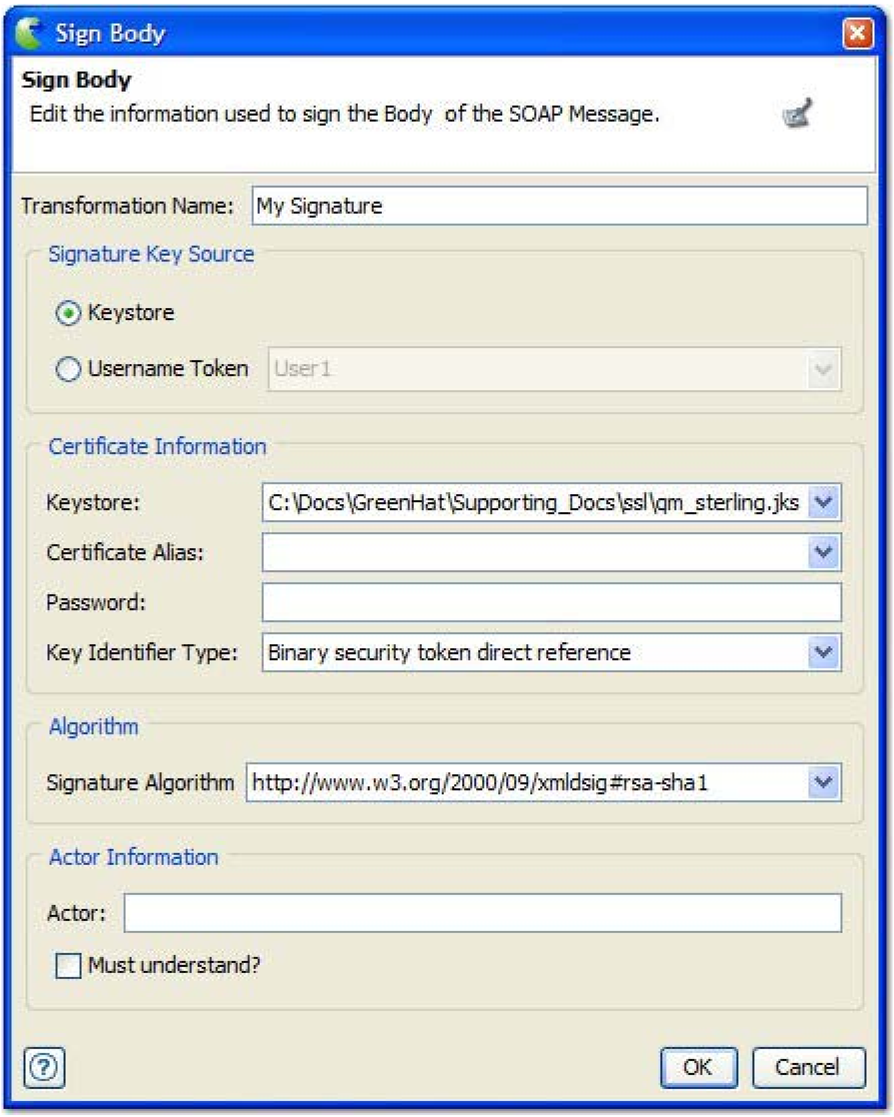
The following fields and options are part of the binary token:

| Field | Description |
|---:|---|
| Transformation Name (**Required**) | User-defined name for the security action (helps identify the action in the main list). |
| Keystore | The Rational Integration Tester identity store to use. |
| Certificate Alias | The public key alias to use (defined in the selected keystore). |
| Actor/Role | Indicates a specific message receiver (either the ultimate receiver or an intermediary). For each actor/role that is defined (that is, in multiple tokens), a separate security header is added to the SOAP header. |
| Must understand | If enabled, makes the SOAP header mandatory for the specified actor/role. In this case, either the header block must be processed or the entire SOAP message should be ignored, and a SOAP fault should be generated.<br>If not enabled, the specified actor/role may or may not process the SOAP header. |

### 4.2.4 Signatures

To sign the body of an outgoing SOAP message, click the action selector and choose **Sign Body**. After selecting the action, the **Sign Body** dialog is displayed.

**NOTE:** If you have not defined an identity store in Architecture School, (refer to *IBM Rational Integration Tester Reference Guide*), or if no acceptable user token exists, you must create an identity store before adding a signature.



Sign Body

**Sign Body**

Edit the information used to sign the Body of the SOAP Message.

Transformation Name: My Signature

Signature Key Source

● Keystore

○ Username Token: User1

Certificate Information

Keystore: C:\Docs\GreenHat\Supporting_Docs\ssl\qm_sterling.jks

Certificate Alias:

Password:

Key Identifier Type: Binary security token direct reference

Algorithm

Signature Algorithm: http://www.w3.org/2000/09/xmldsig#rsa-sha1

Actor Information

Actor:

□ Must understand?

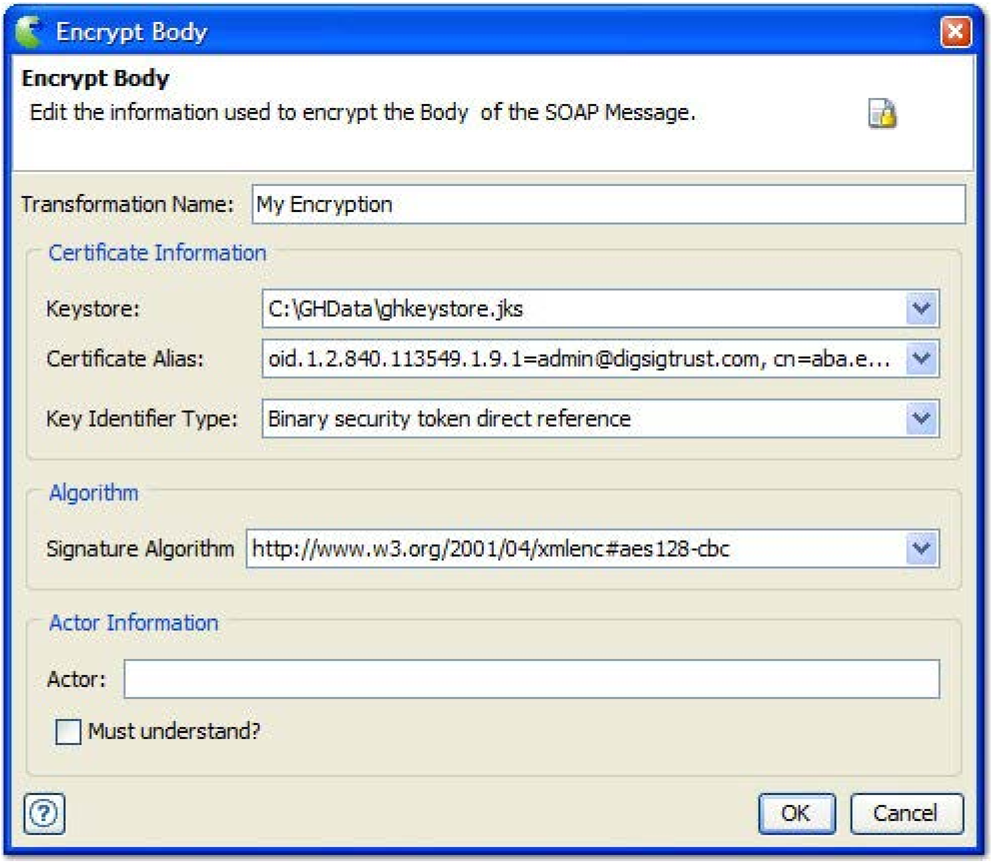
The following fields and options are used for signatures:

| Field | Description |
|---|---|
| Transformation Name (**Required**) | User-defined name for the security action (helps identify the action in the main list). |
| Signature Key Source | Select whether to sign with a keystore (identity store) or user token.<br>**NOTE:** You can only sign with a user token that was created with a **Digest**, or with the **Nonce** and **Created** options enabled. |
| Keystore | The Rational Integration Tester identity store to use |
| Certificate Alias | The private key alias to use (defined in the selected keystore). |
| Password (**Required**) | The password to use for the selected certificate alias. |
| Key identifier type | Indicates how to refer to the signature key:<br>- Binary security token direct reference<br>- Issuer serial<br>- X509 key identifier |
| Signature Algorithm | Indicates signature algorithm to use:<br>- RSA-SHA1: http://www.w3.org/2000/09/xmldsig#rsa-sha1<br>- DSA-SHA1: http://www.w3.org/2000/09/xmldsig#dsa-sha1 |
| Actor/Role | Indicates a specific message receiver (either the ultimate receiver or an intermediary). For each actor/role that is defined (that is, in multiple tokens), a separate security header is added to the SOAP header. |
| Must understand | If enabled, makes the SOAP header mandatory for the specified actor/role. In this case, either the header block must be processed or the entire SOAP message should be ignored, and a SOAP fault should be generated.<br>If not enabled, the specified actor/role may or may not process the SOAP header. |

## 4.2.5 Encryption

To encrypt the body of an outgoing SOAP message, click the action selector and choose **Encrypt Body**. After selecting the action, the **Encrypt Body** dialog is displayed.

**NOTE:** If you have not defined an identity store in Architecture School, (refer to *IBM Rational Integration Tester Reference Guide*), you must create one before adding encryption.



**Encrypt Body**

Edit the information used to encrypt the Body of the SOAP Message.

Transformation Name: My Encryption

Certificate Information

Keystore: C:\GHData\ghkeystore.jks

Certificate Alias: oid.1.2.840.113549.1.9.1=admin@digsigtrust.com, cn=aba.e...

Key Identifier Type: Binary security token direct reference

Algorithm

Signature Algorithm http://www.w3.org/2001/04/xmlenc#aes128-cbc

Actor Information

Actor:

□ Must understand?

The following fields and options are used for encryption:

| Field | Description |
|---|---|
| Transformation Name (**Required**) | User-defined name for the security action (helps identify the action in the main list). |
| Keystore | The Rational Integration Tester identity store to use.<br>**Note:** You can not encrypt with a PKCS keystore. |
| Certificate Alias | The public key alias to use (defined in the selected keystore). |
| Key identifier type | Indicates how to refer to the encryption key:<br>- Binary security token direct reference<br>- Issue serial<br>- X509 key identifier |
| Encryption Algorithm | Indicates encryption algorithm to use:<br>- AES128 in CBC: http://www.w3.org/2001/04/xmlenc#aes128-cbc<br>- AES192 in CBC: http://www.w3.org/2001/04/xmlenc#aes192-cbc<br>- AES256 in CBC: http://www.w3.org/2001/04/xmlenc#aes256-cbc<br>- Triple DES in CBC: http://www.w3.org/2001/04/xmlenc#tripledes-cbc |
| Actor/Role | Indicates a specific message receiver (either the ultimate receiver or an intermediary). For each actor/role that is defined (that is, in multiple tokens), a separate security header is added to the SOAP header. |
| Must understand | If enabled, makes the SOAP header mandatory for the specified actor/role. In this case, either the header block must be processed or the entire SOAP message should be ignored, and a SOAP fault should be generated.<br>If not enabled, the specified actor/role may or may not process the SOAP header. |

# Troubleshooting

**Contents**

This chapter provides answers to common questions and issues that may arise when using HTTP and Web Services.

## 5.1 No Messages Received

Check that the formatter on the publisher and subscriber are the same; if these do not match then any messages sent by the publisher will be ignored by the subscriber.

## 5.2 Address Already in Use: JVM_Bind

Check that the port number configured in the HTTP transport is not already being used by another transport or application on the host machine. Use netstat to view all open ports on a Windows machine. Any open ports being held by Rational Integration Tester will be listed when executing `netstat -n -b` as `GHTester.exe`.

Since port 80 is used by many applications it may be necessary to deactivate some running services such as IIS or any running web servers.

## 5.3 Reply not Received - Receive Reply Timed Out

Client socket timeout can be configured in the HTTP transport properties, if this value is set then the socket will close after the required period of time. If when sending a request to an HTTP resource the reply is not received before the socket timeout is reached then the reply will be discarded and any further test execution will continue.

## 5.4 Red Crosses in SOAP Message

When a message does not conform to a schema, for example if a message is stored in the message repository and the schema it was built from was changed, then upon opening the message there will be a series of red crosses next to each field that no longer conforms to the schema and every parent node above it. Hovering the mouse over the offending field will provide a description of the reason why the field does not conform. If the change to the schema was intentional then the message will have to be rebuilt to correct the problem.

## 5.5 No Messages Shown when "Watching" a Web Service

If the web site or web service is hosted on a server cluster then any requests that you send to the original address may be routed to another server in the cluster, which is not the server originally specified. This Subscriber will then not be able to "watch" those messages. This may be noticeable when monitoring requests to popular web sites, such as www.google.co.uk, during periods of heavy usage.

## 5.6 No Messages Shown when "Watching" a Local Web Service

On Windows the WinPCap libraries that the Subscriber utilises will not pick up messages being sent to and received from a service running on the local machine.

# Glossary

The following table below lists some of the key terms used in this document, and provides a description of each.

| Term | Description |
|---|---|
| Field | A bit of data constituent to a message. Most fields are scalar and therefore unitary, equivalent to data attributes. Vector fields are an aggregation of fields both scalar and vector, and are usually referred to as Messages. See also Message. |
| Message | A unit of information made up of a header consisting of meta-information and a body consisting of the message data. |
| Host | The computer on which a software process runs. |
| Publisher-Subscriber | A messaging paradigm whereby a messaging network consists of Publishers and Subscribers. |
| Transport | Informally, the messaging software in use. For instance, TIBCO Rendezvous, TIBCO ActiveEnterprise, IBM WebSphere® MQ (JMS). |
| Publishing | Making a message (data) available on a message channel. |
| Subscribing | Receiving a stream of messages (data) on a given message channel. |
| Server | A host computer on a network shared by more than one user. |
| SOAP | Simple Object Access Protocol. A platform independent XML based communication protocol used for sending messages between applications over the internet. More information on SOAP definitions can be found at http://www.w3.org/TR/soap. |
| WSDL | Web Service Definition Language. An XML document which describes a Web Service, how it can be found and what services/ methods it provides. More information on WSDL definitions can be found at http://www.w3.org/TR/wsdl. |
| HTTP | Hyper Text Transfer Protocol. Provides client server communication built on top of the TCP/IP Transport layer. |

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited
Intellectual Property Law
Hursley Park
Winchester
SO21 2JN
Hampshire
United Kingdom

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corporation 2001, 2012.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

IBM®