HP Service Test

for the Windows operating system

Software Version: 11.10

User Guide

Document Release Date: February 2011
Software Release Date: February 2011

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.

Copyright Notices

© Copyright 2000 - 2011 Hewlett-Packard Development Company, L.P.

Trademark Notices

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- 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://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web 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 register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Table of Contents

Welcome to This Guide **11**
How This Guide Is Organized 11
Who Should Read This Guide 12
Service Test Updates 13
Service Test Documentation 13
Service Test Feature Movies 13

Chapter 1: Working with Service Test **15**

Concepts **16**

Service Test Overview 16
UFT (Unified Functional Testing) 31
JMS Transport Overview 31

Tasks **32**

How to Populate a Test 32

References **37**

Service Test User Interface 37

Chapter 2: Service Test Activities **75**

Concepts **76**

Activity Overview 76

Tasks **77**

How to Use Date and Time Activities 77
How to Send an Multipart HTTP Request 80
How to Execute Database Commands or Retrieve Data 81
How to Validate an XML file 84
How to Transform an XML File 86
How to Implement UFT Testing 88
How to Create a Call to a Java Class 91

References **93**

The Service Test Activities 93
Activity-Specific Dialog Boxes 116

Chapter 3: Creating and Saving Tests **127**

Concepts **128**

Test Overview 128
Business Process Testing 128

Tasks **131**

How to Create a New Test 131
How to Add a Test to a Solution 132
How to Create a New Business Component 134
How to Define Test Properties or User/System Variables 136

Reference **139**

Test User Interface 139

Chapter 4: Defining Services **147**

Concepts **148**

Preparing Service References 148
Updating Services 149
Negative Testing 151

Tasks **152**

How to Import a WSDL-Based Web Service 152
How to Update a Web Service 155
How to Create a Simple Test for a REST Service 158
How to Create a Template-Based REST Service 160

Reference **164**

Import Service User Interface 164
Update Service Interface 169

Chapter 5: Web Service Security **181**

Concepts **182**

Setting Security Overview 182
Security Scenarios Overview 183
WCF Scenario Settings 187
Advanced Security Settings 191

Tasks 192

How to Set Security for a Web Service on the Port Level 192
How to Set Security for a Specific Web Service Step 194
How to Set Security for a Standard Web Service 195
How to Set Security for a WCF Service 197
How to Set up Common Web Services Security Scenarios 198
How to Customize Security for WCF Type Web Services 202
How to Test Web Services that use WS-Security or SSL 207
How to Set up Advanced Standards Testing 208

Reference 211

Security User Interface 211

Chapter 6: Running Tests 239

Concepts 240

Running Tests Overview 240

Tasks 243

How to Run a Test 243
How to Prepare and Run a Load Test 247

Reference 251

Command Line Syntax 251
Run Test User Interface 252
Debug User Interface 255

Chapter 7: Data Handling 261

Concepts 262

Data Handling Overview 262
Defining Data 262
Unique Data 271
Data Relations and Navigation 272

Tasks 276

How to Create Data Sources 276
How to Assign Data to Test Steps 283
How to Access Data Through Event Handler Coding 287
How to Set the Navigation Properties 290
How to Data Drive a Test Step 291

Reference **293**

Data Handling User Interface 293
Database Data Source User Interface 313

Chapter 8: Run Results Viewer **317**

Concepts **318**

Run Results Viewer Overview 318
Run Results File Location 320
Captured Data 321
Custom Fields 325

Tasks **326**

How to Open Run Results 326
How to Navigate the Run Results Tree 327
How to Manually Submit Defects to ALM/QC 329
How to Export Run Results 330

Reference **332**

Run Results Viewer User Interface 332
Run Results Viewer Commands 334
Run Results Viewer Panes 337
Run Results Viewer Dialog Boxes 350

Chapter 9: ALM/QC Integration **365**

Concepts **366**

Managing Tests Through HP ALM/QC Overview 366
ALM/QC Tests and Version Control 367

Tasks **368**

How to Work with Tests in ALM/QC 368
How to Save Tests to ALM/QC Projects 370
How to View a List of Previous Test Versions 371

Reference **372**

ALM/QC User Interface 372

Appendix A: Coding Service Test Events **379**

Concepts **380**

Writing Code for Events Overview 380

Tasks **382**

How to Open a Window for Writing Custom Code 382
How to Access Input and Output Properties 384
How to Set Input and Output Properties 385
How to Use the Logging Function 387
How to Use the Report Function 388
How to Write Checkpoint Events 389
How to Retrieve and Set Test Variables 391
How to Encrypt and Decrypt Passwords 393
How to Use the OnReceiveResponse Web Service Event 394

Reference **396**

Search and Replace for Events User Interface 396
Context Methods 400
Logging Options 401
Assert Options 402

Appendix B: Extensibility in Service Test **403**

Concepts **404**

Creating New Activities - Overview 404
Runtime Files 405
Signature Files 411
Addin Files 421
Resource Files 424

Tasks **425**

How to Create a New Custom Activity 425
How to Debug an Activity 430

Index **431**

Welcome to This Guide

Welcome to HP Service Test, HP's tool for creating tests, with a focus on testing SOA and headless technologies.

This chapter includes:

- How This Guide Is Organized on page 11
- Who Should Read This Guide on page 12
- Service Test Updates on page 13
- Service Test Documentation on page 13
- Service Test Feature Movies on page 13

How This Guide Is Organized

This guide contains the following chapters:

Chapter 1 Working with Service Test

This chapter describes the Service Test interface, its layouts, and dockable window panes.

Chapter 2 Service Test Activities

This chapter describes the built-in activities provided with Service Test and their properties.

Chapter 3 Creating and Saving Tests

This chapter describes how to create a new solution and project.

Chapter 4 Defining Services

This chapter describes how to import services and maintain them.

Chapter 5 Web Service Security

This chapter describes how to set security for Web services on both port and test step levels.

Chapter 6 Data Handling

This chapter describes how to add data to your test and how to assign it to your test's parameters.

Chapter 7 Running Tests

This chapter describes how to run tests from the user interface or command line.

Chapter 8 Run Results Viewer

This chapter describes how to view the results in the Run Results Viewer and filter or export the information according to your needs.

Chapter 9 ALM/QC Integration

This chapter describes how to integrate with HP ALM/QC (Application Lifecycle Management/Quality Center) to manage your tests and implement version control.

Who Should Read This Guide

This guide is for the following users of Service Test:

- Application developers or configurators
- System or instance administrators
- Functional and load testers

Readers of this guide should be moderately knowledgeable about SOA systems and Web services.

Service Test Updates

You can check online for updates to all HP products installed on your computer at any time by choosing **Start** > **Programs** > **HP Service Test 11.10** > **HP Update**. You can then choose which updates to download and (optionally) install.

Service Test Documentation

Service Test includes the following online documentation:

Readme. Choose **HP Service Test 11.10** > **Readme** from the Start menu.

HP Service Test Online Help. Available from the Service Test user interface by clicking in the window and pressing F1 or clicking the **Help** button.

User Friendly Documentation. Online books can be viewed and printed using Adobe Reader, which can be downloaded from the Adobe Web site (www.adobe.com).

- **HP Service Test User Guide.** Choose **HP Service Test 11.10** > **Documentation** > **User Guide** from the Start menu.
- **HP Service Test Installation Guide.** Choose **HP Service Test 11.10** > **Documentation** > **Installation Guide** from the Start menu.
- **HP Service Test Tutorial.** Choose **HP Service Test 11.10** > **Documentation** >**Service Test Tutorial** from the Start menu.

Service Test Feature Movies

Service Test includes a movie which introduces the application and illustrates some of its main features.

To access the movie, select **Help** > **Product Feature Movies** > **Introduction to HP Service Test**.

1

Working with Service Test

This chapter includes:

Concepts

- Service Test Overview on page 16
- UFT (Unified Functional Testing) on page 31
- JMS Transport Overview on page 31

Tasks

- How to Populate a Test on page 32

References

- Service Test User Interface on page 37

Troubleshooting and Limitations - General on page 73

Concepts

Service Test Overview

Welcome to HP Service Test, HP"s tool for the construction and execution of functional tests for headless systems.

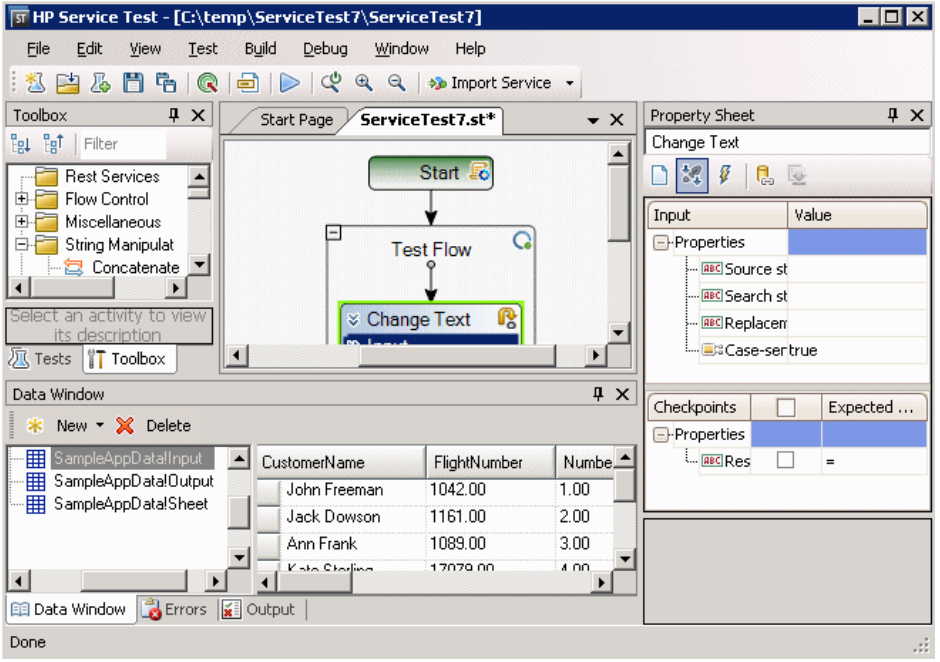
You create a test by dragging and dropping activities from the Service Test toolbox into a canvas. The toolbox provides a collection of activities for functional testing in areas such as REST, Web Services, JMS, and HTTP. You can add more activities to the toolbox by importing WSDLs or providing other contract definitions.

This section includes:

- "Introducing Service Test" on page 17
- "Performing Integrated Testing" on page 17
- "The Canvas" on page 18
- "The Toolbox" on page 23
- "Data Sources" on page 24
- "Using Service Test for Testing Web Services" on page 25
- "Using Service Test for Testing REST Services" on page 25
- "Java Testing" on page 26
- "The Property Sheet" on page 27

Introducing Service Test

Service Test uses a visual designer as a canvas for creating steps in a test. You create tests by dragging activities from a toolbox palette into the canvas and defining their parameters.



Performing Integrated Testing

In testing your systems, you perform known activities and observe the response. By checking the response, you determine whether your system is performing as expected.

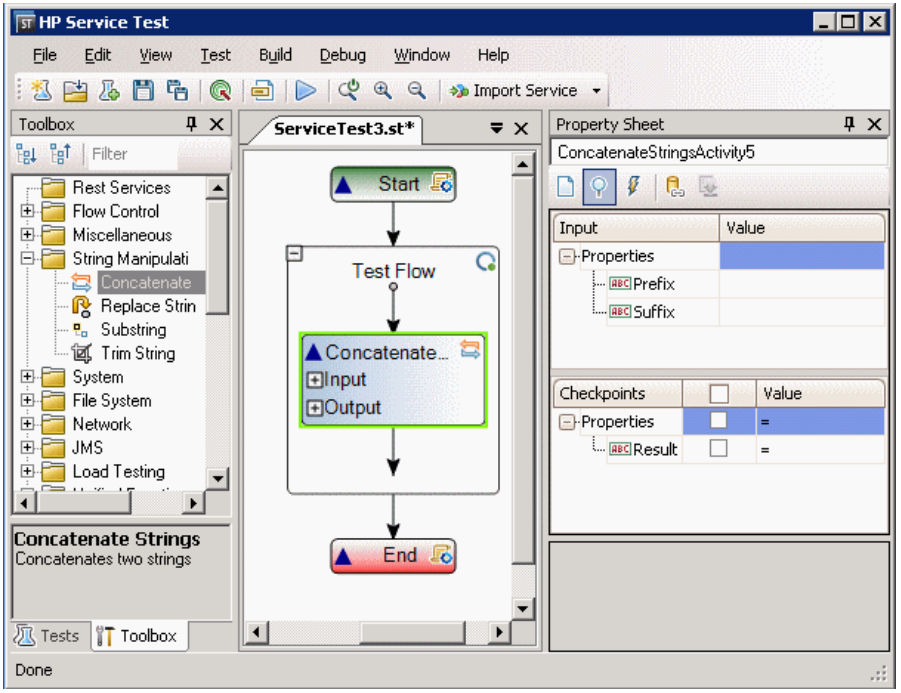
For example, in a **Database** > **Open Connection** activity, you drag the activity into the canvas and provide a Database connection string. You then can drag a **Select Data** activity into the canvas. After running these actions, you can check the response with an expected value to confirm that your service is functioning properly.

The steps to create a basic test are:

- Drag an activity into the canvas
- Assign input data
- Set checkpoints to verify the response

The Canvas

The canvas in Service Test provides a visual representation of the test flow. You populate the canvas by dragging in activities from the Toolbox palette.



You can drag Web Services, REST services, or any Toolbox activity, into the canvas. For a service to appear in the Toolbox, you must first import it (Web services) or define it (REST services) in your project. For details, see "How to Import a WSDL-Based Web Service" on page 152.

Tab Title

The canvas tab title is **<Test name>.st**. An asterisk denotes changes that were not yet saved.

Manipulating Steps

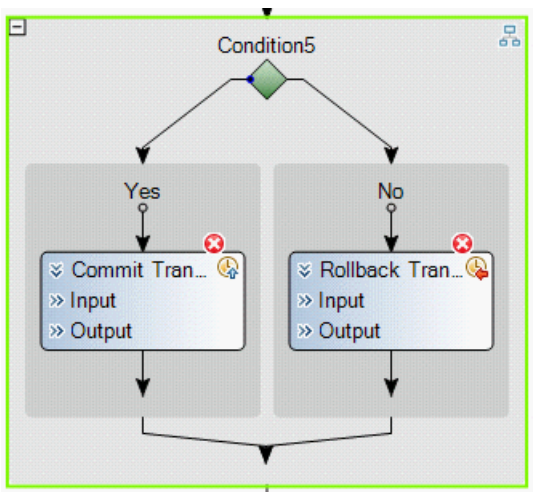
The canvas allows you to manipulate the steps in the following ways:

- **Reorder**. Drag a step from one location to another.
- **Copy and Paste**. Select a step and press **Ctrl+C** to copy it to the clipboard. Press **Ctrl+V** to paste it into another location within the Test Flow or to another loop.
- **Delete**. Select a step and press the keyboard's **Delete** button.

Flow Control

The canvas allows you to control the flow of the steps in the following ways:

- **Loop**. The main **Test Flow** area provides a default loop for test steps. You can add steps to the Test Flow or add additional loops. Using the Properties Sheet, you can specify the type of loop, number of iterations, and conditions for the Test Flow or loop. For details, see "Input/ Checkpoints View" on page 55.
- **Conditional steps**. Adds a two-branched conditional node to the test.

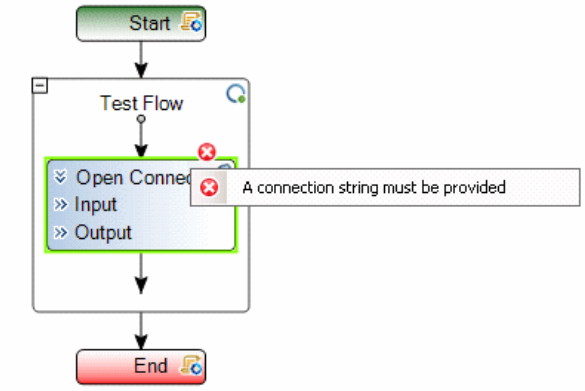


For conditional steps, you drag the **Condition** activity from the Toolbox into the canvas, and then drag an activity into each of the conditional branches.

- **Break.** Moves control to the step after the loop.
- **Continue.** Moves control back to the first step in the **Test Flow** or **Loop** and increments it to the next iteration.

Alerts

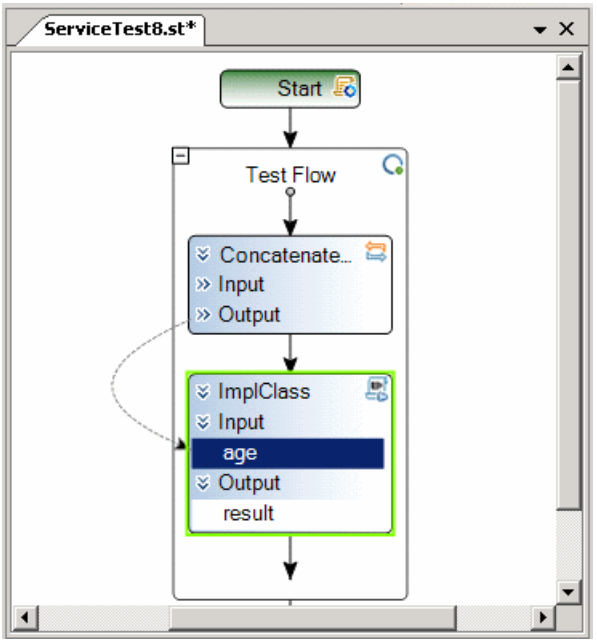
Steps that require intervention in order to run, such as **HTTP**, **Custom Code**, and **SOAP Request**, post an alert in their top right corner. Clicking an alert button displays the action that you are required to do.



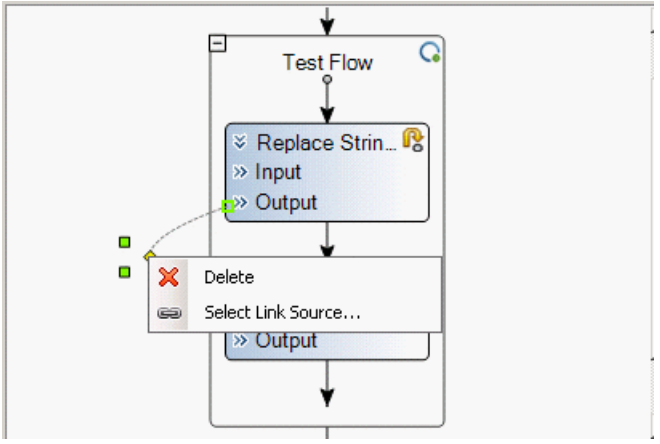
You can also view this information in the Errors tab by selecting **View** > **Errors**.

Connecting Data

The canvas displays the relationship between output and input properties. An arrow indicates the direction of the relationship.



You can edit the link between steps by selecting the linking arrow and choosing **Select Link Source** from the shortcut menu.



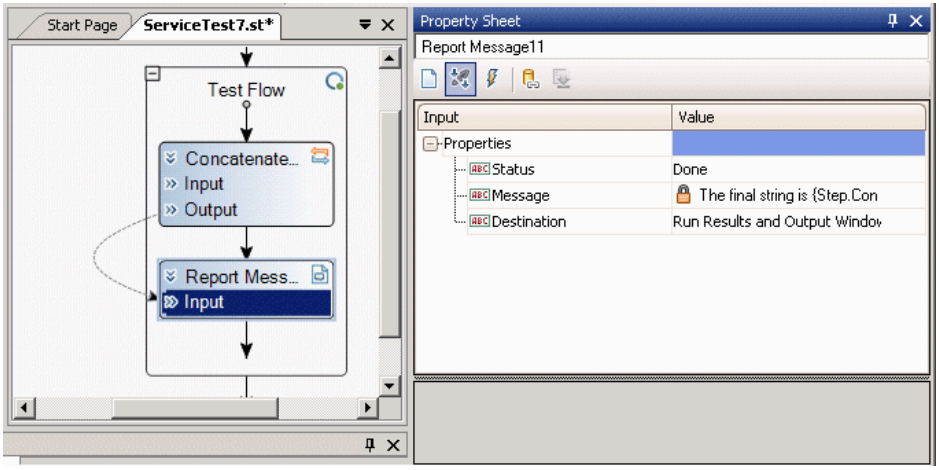
For more information, see the "Select Link Source Dialog Box" on page 305.

For details about linking multiple steps to a single result, see "Outgoing Links" on page 268.

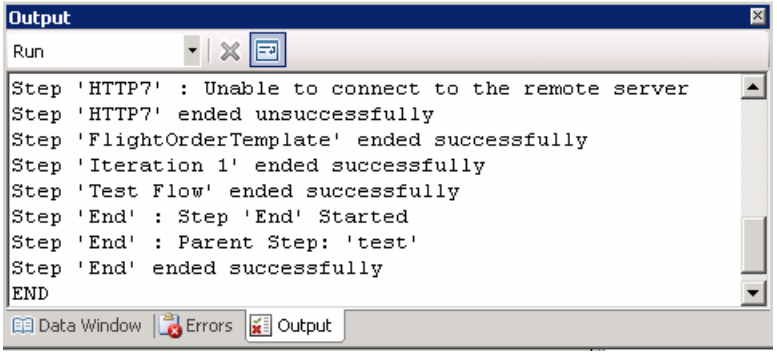
Sending Messages to the Output and Results Viewer

The **Report Message** activity lets you send custom messages to the Output log and/or the Results viewer. You drag the activity to the canvas and set its properties in the Property sheet.

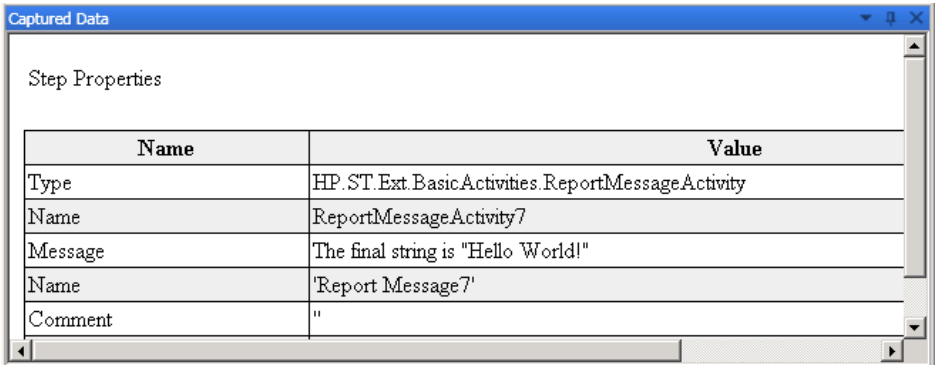
As with any step, you can link its input to the output of a prior step and show the results in the Output window.



In the following example, the Output window shows the message string.



When you use this activity, Service Test adds a **Message** row to the Results report, with the contents of the message.

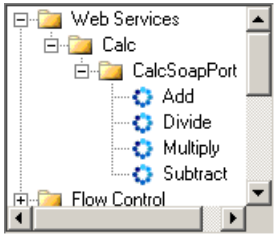


For more information, see the "Run Results Viewer" on page 317.

The Toolbox

The Toolbox palette contains all of the activities and flow control objects that you need in order to create a test.

After you import a Web service, it appears in the Toolbox, under the **Web Services** node. When you expand the Web Services node, the Toolbox displays the service's name, port, and operations.



For additional details about each of the nodes, see the "Toolbox Palette" on page 46.

Extensibility

For greater control over test steps, you can define custom code activities that seamlessly integrate with Service Test. You can also customize the behavior of existing activities using event handlers. For more information, see "Coding Service Test Events" on page 379.

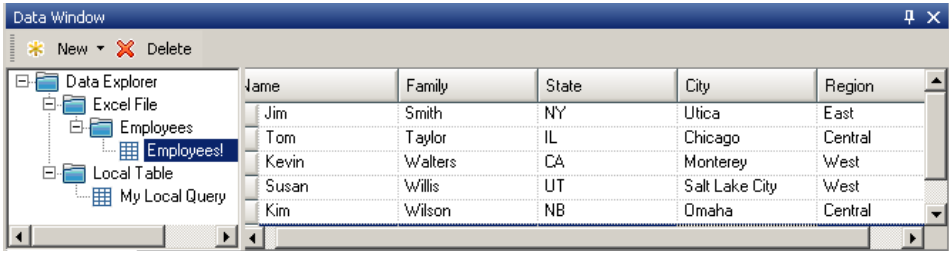
For information about adding new activities, speak with you HP representative.

Data Sources

Using the Data window, you can define data for your test's properties. You can link to a variety of data sources, such as Microsoft Excel or XML files. You can also define data locally for the test steps.

When you load data from Excel, you can indicate whether to make a local copy of the data, or a referenced data source which refers to the data at its original source.

A data grid displays the Excel and locally defined data. If you created a referenced data source, you will be unable to edit the values in the data grid.



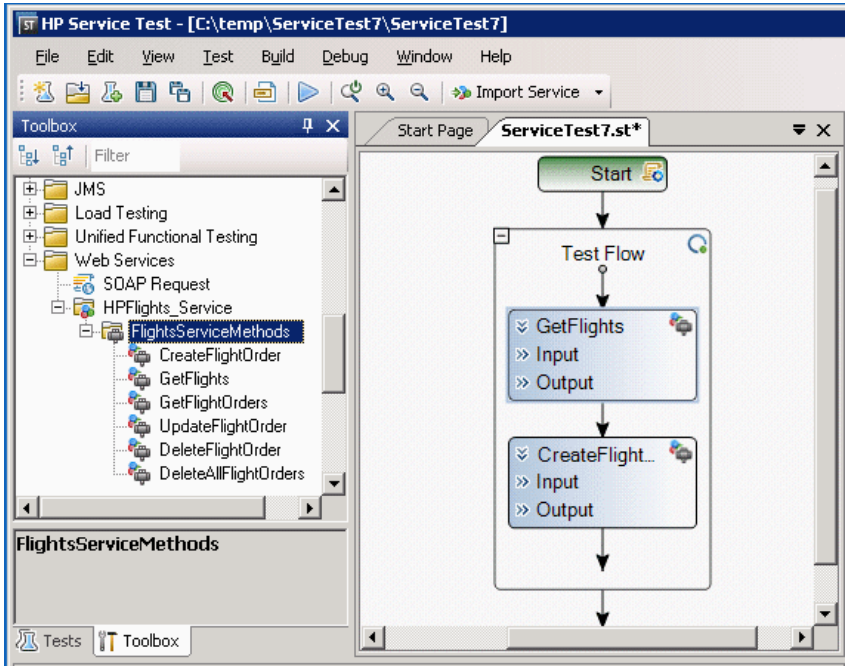
After you load data, you can assign it to properties in your test.

For information on using data in your test, see "How to Assign Data to Test Steps" on page 283.

Using Service Test for Testing Web Services

You can use Service Test to test standard Web Services. For Web services, you import the service's WSDL file.

After importing the WSDL file, the Toolbox window lists the operations in the Toolbox. You then drag the operations directly into canvas and provide input values.



For task details, see "How to Import a WSDL-Based Web Service" on page 152.

Using Service Test for Testing REST Services

You can use Service Test to test non-SOAP Web Services, such as REST.

For REST services, you manually define the metadata and model your activities based on the design of your service's requests.

After setting the REST resource definitions, you can create REST method calls based on the metadata. To validate the results, you set checkpoints for the REST method's response.

Service Test allows you to create templates for individual REST methods. The templates include all of the request and header information, as well as custom parameters. Using this template, you can drag the method into the canvas without having to configure its details repeatedly.

For task details, see "How to Create a Template-Based REST Service" on page 160.

Java Testing

Service Test allows you to call Java classes with its **Call Java Class** activity.

In order to use this step, you must implement the Service Test Java interface. The aim of this interface is to define a contract or bridge between the Java artifacts owner and Service Test. The interface contains three methods:

- **getInputProperties.** Returns a mapping of the input property names and their Java class.
- **getOutputProperties.** Returns a mapping of the output property names and their Java class.
- **Execute.** A method that receives the mapping of the input properties names and their actual values i.e. their object instance. In this method, you process input properties and delegate them to your own Java artifacts. Afterwards you process the output properties and send their mappings and their actual values as the method's output.

The following table lists the relevant paths:

File	Path
Interface code	➤ **Path**: %ST_DIR%\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java\ServiceTestCall.java ➤ **Default installation**: C:\Program Files\HP\HP Service Test\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java\ServiceTestCall.java
Sample code	➤ **Path**: %ST_DIR%\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java\sample ➤ **Default installation**: C:\Program Files\HP\HP Service Test\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java\sample

To configure a Java step, you select a root path, a jar file containing the class, and the actual Java class. You can provide additional classpaths and jar files for the Java call. For details about selecting the Java class and packaging, see the "Call Java Class Settings Dialog Box" on page 124.

Service Test supports the following Java classes: **Byte**, **Short**, **Int**, **Long**, **Float**, **Double**, **Boolean**, **Date**, and **String**. The **Char** class is not supported.

For task details, see "How to Create a Call to a Java Class" on page 91.

The Property Sheet

The Property Sheet provides several views that allow you to set properties for each of the test steps.

General View

The **General** view shows the **Step ID** and **Name** properties of the test step. Several activities have additional General properties. For example, for Web Service operations, this view also shows the **Endpoint Address, SoapAction, ContentType** and **IsOneWay** setting. For details, see the "General View" on page 54.

Input/Checkpoints View

The Input/Checkpoints view displays a grid of properties and their values. This view shows properties that apply to the selected activity or step:

Selected step	Title	Displayed in view
Activities	**Input/Checkpoint**	Input and checkpoint properties and their values.
Canvas, Start step, End step	**Test Input Parameters**	Input and Output properties for the test.
Loop (Test Flow or additional loops)	**Input**	The type of loop and the relevant properties such as the number of iterations. For details, see "Input Properties - Loop" on page 95.

The display grid shows important information about each of the elements such as whether they will be encoded as base64 or whether they can be set to a NIL value. A green triangle indicates an optional element. The element is included when the toggable triangle is filled and excluded when it is cleared.

Checkpoints allow you to validate the service, by providing an expected value. Service Test also provides structural checkpoints to check if the service is in conformance with its schema and the WS-I standard.

You can automatically select data for the properties from data sources, system variables, and so forth. For details, see "Select Link Source Dialog Box" on page 305.

For a list of the user interface elements in the grid, see the "Input/Checkpoints View" on page 55.

Events View

The **Events** view lets you select or set a handler function for each of the step's events, or to open an editor to write code describing what to do when the event occurs.

For information on creating an event handler, see "Specify an event handler - optional" on page 36.

Test Variables View

The **Test Variables** view, is available for the **Start** and **End** steps and when standing on an empty part of the canvas. It displays the user and system-defined test variables. You can save a set of user variables in a **profile**. You can create multiple profiles and compare them. For details, see "Define user variables" on page 136.

REST HTTP Input View

The **REST HTTP Input** view, available when selecting a REST service method in the toolbox, stores the **Endpoint Address**, **HTTP Method**, **HTTP Version**, and the **Request Headers**. For details, see the "REST HTTP Input View" on page 63.

HTTP View

The **HTTP** view, available for the HTTP activity and REST service methods, displays the HTTP request, response and version. You can also set the headers and import the body of the request. For details, see "HTTP View" on page 63.

Multipart View

The **Multipart** view is available for the HTTP activity with a multipart message. This view lets you set the properties for the second and all subsequent parts. For details, see "Multipart View" on page 64.

SOAP Fault View

The **SOAP Fault** view is available for Web Services only. It allows you to specify whether to expect a SOAP fault and to set the SOAP fault properties. For details, see the "SOAP Fault View" on page 61.

Attachments View

The **Attachments** view is available for Web Services. It allows you to view the input and output attachments and their properties. For details, see the "Attachments View" on page 66.

Security Settings View

The **Security Settings** view, available for Web services, allows you to add tokens, message signatures, and encryption to the service calls. You can also select security scenarios and customize them. For details, see "Security Settings View" on page 67.

Data Sources View

The **Data sources** view is available when you click in the empty area of a Loop box. It allows you to add data sources that appear in the **Data Window** tab, and set its data navigation policies. For details, see "Navigating within the Data" on page 272.

Data Source Properties View

The **Data Source Properties** view is available when you select a data source in the Data Window tab, other than a database source. For details, see "Data Source Properties View" on page 68.

Database Data Source Properties View

The **Database Data Source Properties** view, is available after you create a Database data source in the **Data Window** tab. For details, see "Database Data Source Properties View" on page 69.

UFT (Unified Functional Testing)

UFT (Unified Functional Testing) activities allow you to create tests with steps from Service Test, QuickTest Professional, or LoadRunner's Virtual User Generator. QuickTest Professional is HP's tool for functional testing of client and Web applications. HP LoadRunner allows you to perform Load Testing on your applications and environment.

You create the tests or scripts in the original application and call them within the Service Test flow. You can also call another Service Test test from within the Test Flow.

For user interface details, see the "Toolbox Palette" on page 46.

For task details, see "How to Implement UFT Testing" on page 88.

JMS Transport Overview

"The Property Sheet" on page 27 is a J2EE standard for sending messages—either text or Java objects—between Java clients. Service Test supports the sending of text messages between Java clients. There are two scenarios for communication:

- **Peer-to-Peer.** Also known as **Point-to-Point**. JMS implements point-to-point messaging by defining a message queue as the target for a message. Multiple senders send messages to a message queue, and the receiver gets the message from the queue.
- **Publish-Subscribe.** Each message is sent from one publisher to many subscribers through a designated topic. The subscribers only receive messages sent after they have subscribed.

To interpret Topic and Queue names, Service Test calls a lookup method on a JNDI context, defined in the Property Sheet's Test Settings. For details, see the "Test Settings View" on page 52.

For a list of the activities and their properties, see "JMS Activities" on page 110.

For task details, see "Set up a JMS message - optional" on page 35.

Tasks

How to Populate a Test

This task describes how to set up the workflow and add steps to your tests. For details on creating the test and defining user or test variables, see "Creating and Saving Tests" on page 127.

This task includes the following steps:

- "Prepare the service references - optional" on page 32
- "Create the test flow" on page 33
- "Add activities to the test" on page 33
- "Define data for the test" on page 33
- "Create a Custom Code activity - optional" on page 33
- "Add attachments to the test - optional" on page 34
- "Validate an output attachment- optional" on page 34
- "Set up a JMS message - optional" on page 35
- "Configure SOAP Fault information - optional" on page 35
- "Specify an event handler - optional" on page 36

1 Prepare the service references - optional

For Web Service based tests, import your WSDL at this point. For details, see "How to Import a WSDL-Based Web Service" on page 152.

For REST services, build the metadata for the service. For details, see "How to Create a Template-Based REST Service" on page 160.

Skip this step when using the built-in activities, such as string manipulation.

2 Create the test flow

Expand the nodes of the Toolbox palette and drag **Flow Control** activities into the canvas:

- **Loop.** Drag the **Loop** activity into the canvas to add a loop other than the standard **Test Flow**. Specify a loop condition in the input properties.
- **Condition**. Drag a **Condition** activity into the Test Flow or Loop frame. Specify a condition in the input properties.
- **Sleep.** Drag a **Sleep** activity into the Test Flow or Loop frame. Indicate a time delay in milliseconds.

3 Add activities to the test

Expand the nodes of the Toolbox palette and drag activities into the Text Flow box within the canvas. If you added a **Condition** step, drag activities into the condition branches.

For a list of the activities, see the "Toolbox Palette" on page 46.

4 Define data for the test

For details, see "How to Assign Data to Test Steps" on page 283 or "How to Data Drive a Test Step" on page 291.

5 Create a Custom Code activity - optional

Select the **Custom Code** activity from the **Miscellaneous** category and drag it into a loop.

a Open the **Input/Checkpoints** view in the Property Sheet.

b Add new input and output properties based on your needs.

c Open the **Events** view in the Property Sheet.

d Double-click the **Handler** column of the **ExecuteEvent** row. Service Test opens a new tab **TestUserCode.cs**.

e Locate the **Todo** section and enter your custom code. Follow the sample code in the comments and use autocompletion to write your code.

f Click **File** > **Save All** to save all buffers of the test.

For details and examples, see Appendix A, "Coding Service Test Events."

6 Add attachments to the test - optional

For Web Service activities that support input attachments, add an input attachment:

a Select the Web Service in the canvas and open the **Attachments** view in the Property Sheet.

b In the upper pane, select an attachment **Type**: **DIME** or **MIME**.

c Click in the **Attachments** row and click the **Add** button to add an array element.

d Select an **Origin** for the attachment using the **Browse** button.

e Select an **Origin Type**, **Content Type**. Specify a **Content ID** or keep the default value, **Auto**.

7 Validate an output attachment- optional

To validate an output attachment:

a Select the Web Service in the canvas and open the **Attachments** view in the Property Sheet.

b Click in the **Attachments** row in the Checkpoints pane, and click **Add** to add an array element (it may be necessary to expand the column).

c Select the check box adjacent to each item that you want to validate. Specify values for the elements being validated: **Content Type** and/or **Content ID**.

d To validate content, click the **Calculate the file checksum** button icon in the **Content** row. Service Test calculates the file's checksum using the MD5 Hash function.

For user interface details, see "Attachments View" on page 66. You can also check for received attachments in the test's folder, stored as files with a .**bin** extension.

8 Set up a JMS message - optional

For JMS messages, you set up the environment before adding a test step.

a Click in the canvas and open the **Test Settings** view in the Property Sheet. Set the Java test settings for the VM and JMS. For details, see the "Test Settings View" on page 52.

b Expand the **JMS** node in the toolbox and drag a JMS activity into the canvas.

c Set the step's properties. Click on the step in the canvas, and open the **Input/Checkpoints view** in the Property Sheet. Enter a Queue, Subscription, Topic name, and any other relevant property values.

d For Send activities, specify a message.

e For Receive activities, select the output properties you want to validate in the **Checkpoints** pane and specify their values.

For more details, see "JMS Transport Overview" on page 31.

9 Configure SOAP Fault information - optional

To apply negative testing to a Web service:

a Open the **SOAP Fault** view.

b Select **Fault is expected**.

c Provide SOAP Fault checkpoint values for the negative testing:

- To work In the **XML** layout: Expand the SOAP nodes and define **Any** elements for the SOAP Header and Body. If relevant, provide values for **faultcode**, **faultstring**, or **faultactor**.
- To work with **XPath** expressions: Click the **XPath** tab and use the Add button to add new XPath entries. Copy the XPATH entry into the cell.

For details, see the "SOAP Fault View" on page 61.

10 Specify an event handler - optional

For non-custom code activities, you can define default event handlers for checkpoints, before step executions, and after step executions.

The checkpoint event handlers help you verify the output values in your test. You can use a Report, Assert, or Log function to gather information about your service.

To add a checkpoint event handler for an activity:

a Select an activity title in the canvas and open the Events view in the Property Sheet.

b In the **CodeCheckPoint** row, select **Create a default handler**.

c Edit the code in the **TestUserCode.cs** tab. Locate the **Todo** section and add your custom code. Follow the sample code in the comments and use auto-complete to create an expression.

d To access the properties of an activity, cast it beforehand. For example:

```
ConcatenateStringsActivity cat = args.Activity as ConcatenateStringsActivity;
args.Checkpoint.Assert.Equals(cat.Prefix+cat.Suffix, cat.Result);
```

e Click **File** > **Save All** to save all buffers of the test.

For details and examples, see Appendix A, "Coding Service Test Events."

Tip: To instruct Service Test to ignore all event handlers, select a handler and press DELETE to clear the field in the grid.

References

Service Test User Interface

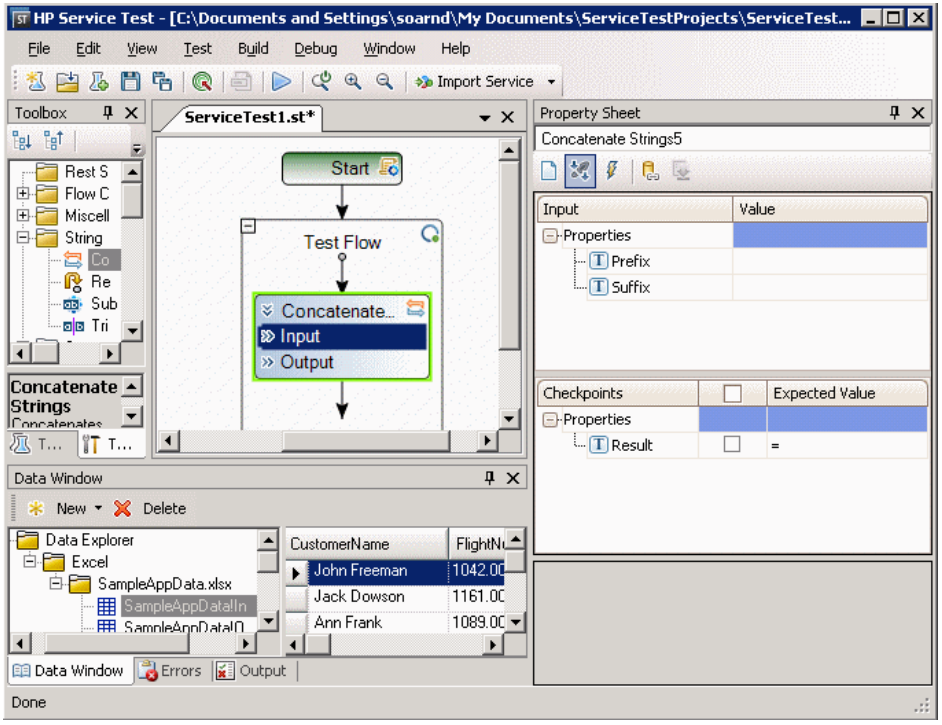
The Service Test interface lets you design a test by dragging activities into a canvas.

This section includes:

- Service Test Main Window on page 38
- Tests Pane on page 43
- Toolbox Palette on page 46
- Property Sheet on page 48
- Output Tab on page 71
- Errors Tab on page 72

Service Test Main Window

This screen is Service Test's main window for creating and populating tests.



To access	Select **Start** > **HP Service Test 11.10** > **HP Service Test 11.10.**
Relevant tasks	"How to Populate a Test" on page 32

User interface elements are described below (unlabeled elements are indicated by angle brackets):

UI Elements	Description
	Create New Test. Opens the Create New Test dialog box.
	Open Test. Prompts you to open an existing test.
	Add New Test. Opens the Add New Test dialog box.
	Save. Saves the file in the active tab. **Tip:** Select **File** > **Save Test As...** to save the test to a new location.
	Save All. Saves all open files.
	ALM/QC Connection. Opens the HP ALM/QC Connection dialog box for defining server information and logging into an ALM/QC project.
	Enable Load Testing. Enables the test for Load Testing. **Note**: ➤ This button adds a menu to the **Run** button, to run the test in Load Testing mode. ➤ The button is disabled if you created the test with the Load Test template.
	➤ **Run Project.** Saves, compiles and runs the current project. ➤ **Run Test (Load Test).** Saves, compiles and runs the current test in Load Test mode with the **mdrv** driver. **Note:** The **Run Test** option is only available for Load Test type tests (created with the Load Test template or converted from the standard template to a load test).
	➤ **Break Test.** Pauses the test run. ➤ **Continue Test.** Resumes the test run from the point of the Break.

UI Elements	Description
	Stop. Stops the current test run.
	Zoom Controls: ➤ **Default Zoom.** Restores the default enlargement of the objects in the canvas. ➤ **Zoom in.** Enlarges the objects in the test canvas. ➤ **Zoom Out.** Reduces the objects in the test canvas.
	The following step commands are only active when the test is in **Break** mode. ➤ **Step Over.** Executes the current statement. If the statement calls another procedure, Service Test will not step into that procedure. It will run the procedure, and advance to the next statement in the current procedure. ➤ **Step Into.** Executes the current statement. If the statement calls another procedure, Service Test steps into that procedure. ➤ **Step Out.** If you used **Step Into** and the control moved to a called procedure, **Step Out** runs the current procedure, and returns to the procedure from which it was called.
	Opens one of the Import dialog boxes: ➤ **Import Service from URL or UDDI.** Opens the Import Service from URL or UDDI dialog box. For user interface details, see "Import Service from URL or UDDI Dialog Box" on page 167. ➤ **Import Service from FIle or ALM/QC Application Component.** Opens the Select Service dialog box for importing a WSDL from the file system or HP ALM/QC with HP Service Test Management. For user interface details, see the "Select Service Dialog Box" on page 165.

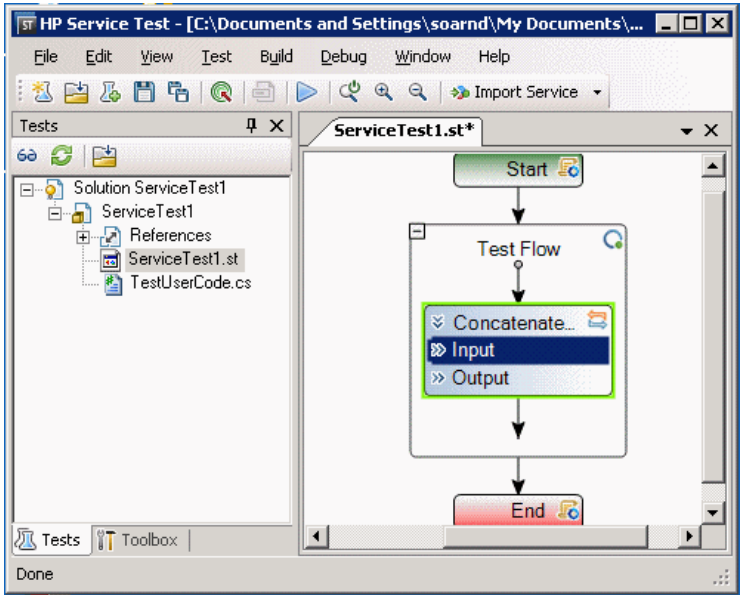
UI Elements	Description
<Editor / Designer pane>	A series of tabs, to view and edit the file selected in the left pane: For example: ➤ Visual display of the test (default) ➤ C Sharp files (**.cs**) ➤ XSD files (**.xsd**) ➤ Solution file (**.sln**)
Data Window tab	Displays the data from an Excel or XML file, a database, or local tables. You can use this data in your tests. ➤ **Data Explorer**. An expandable/collapsible tree hierarchy of the table data. ➤ **New.** Creates a new data node in the Data Explorer through: ➤ **Excel File.** Opens the **Add New Excel File Data Source** dialog box for creating a new set of data. For details, see the "Add New Excel File Data Source Dialog Box" on page 299. ➤ **Local Table**. Opens the **Add New Local Table Data Source** wizard. For details, see the "Add New Local Table Data Source Wizard" on page 300. ➤ **XML**. Opens the **Add New XML Data Source** dialog box. For details, see the "Add New XML Data Source Dialog Box" on page 302. ➤ **Database**. Opens the **Add New Database Data Source** wizard. For details, see the "Add New Database Data Source Wizard" on page 313. ➤ **Delete.** Removes the selected data source. For details, see "Data Window Tab" on page 294.
Errors tab	Displays errors, warning, and messages about the test run. The default location is the bottom pane.
Output tab	Displays the Output log for the compilation and test run. For details, see "Output Tab" on page 71. The default location is the bottom pane.

UI Elements	Description
Tests pane	Displays the test components in a tree hierarchy. For details, see the "Tests Pane" on page 43. The default location is the left pane.
Property sheet	A series of screens showing the general properties, schemas, snapshots, and event information for the activity selected in the canvas. For details, see "Property Sheet" on page 48. **Note:** Only active when viewing the canvas.
Toolbox palette	A tree view of all of the available activities. This includes built-in activities and operations of imported services. For details, see the "Toolbox Palette" on page 46. The default location is the left pane.

Tests Pane

The **Tests** pane enables you to view all the test components in a tree hierarchy.

This screen is Service Test's main window for creating and populating tests.



To access	Select **View** > **Tests** (CTRL+ALT+L).
Relevant tasks	"How to Create a New Test" on page 131
See also	"Test User Interface" on page 139

The following elements are included (unlabeled UI elements are shown in angle brackets):

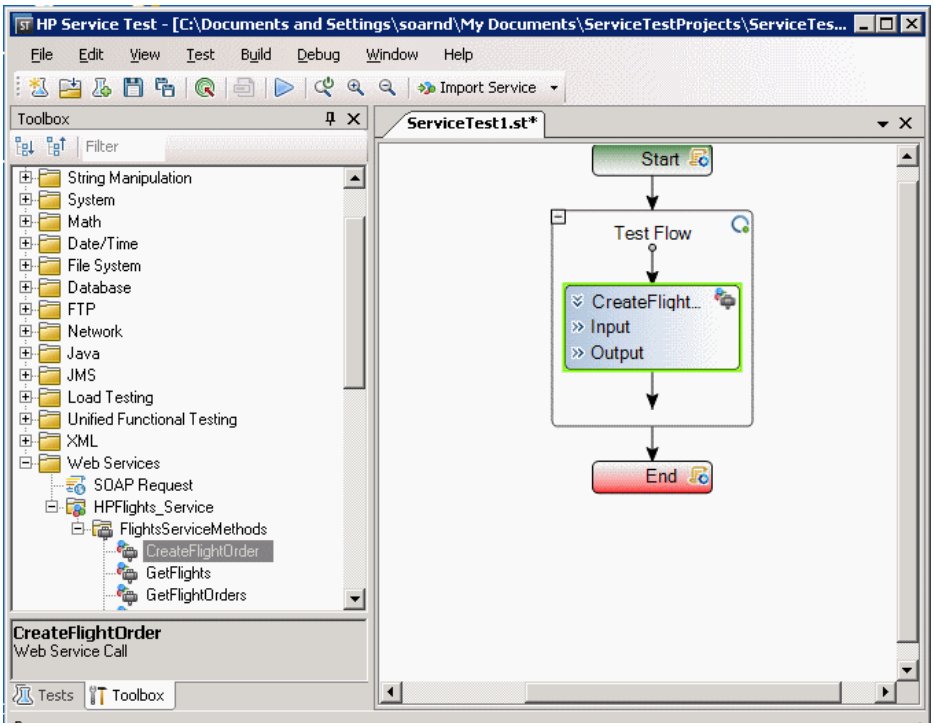
UI Elements	Description
	Show All. Shows/hides all files associated with current test.
	Refresh. Refreshes the **Tests** tree.
	Open. Opens a tab with the contents of the selected file. **Note:** Only available for the st, canvas file.
<Test> Tree	Tree hierarchy of the test components. Expanding the parent node displays all the references and dependent files. **Tip.** To view all files, click the **Show All** button.
<Test> Tree **- Solution context menu**	➤ **Build Solution**. Begins a new build (F8). ➤ **Rebuild Solution**. Rebuilds the current solution (ALT+F8). ➤ **Clean Solution.** Performs a clean build. ➤ **Add.** Allows you to add a New Test, Existing Test, New Solution Folder, or another Item to the solution tree. ➤ **Paste.** Pastes a solution from the clipboard. ➤ **Rename.** Renames the current solution node.
<Test> Tree - Project context menu	➤ Build commands: **Build**, **Rebuild**, **Clean** ➤ References: **Add Reference** ➤ Open/Run: **Open With...**, **Run Project** ➤ Edit: **Cut**, **Paste**, **Remove**, **Rename**
<Test> Tree - References context menu	➤ **Add Reference.** Opens the Add Reference dialog box.

UI Elements	Description
<Test> Tree - File context menu	➤ **Refresh**. Reloads the entire tree. ➤ **Remove.** Deletes the selected node.
<Test> Tree - Test file (*.st) context menu	➤ Open commands: **Open**, **Open With…**, **Open Containing Folder in Explorer** ➤ Add commands: **New Item**, **Existing Item**, **New Folder**, **Existing Folder**, **New Dependent Item**, **Existing Dependent item** ➤ Edit commands: **Cut**, **Copy**, **Paste**, **Delete**, **Rename**

Toolbox Palette

The Toolbox palette enables you to view all of the available activities. This includes imported services and built-in activities such as that you can use in your visual design, such as **Replace String** or **File Exists**.

This screen is Service Test's main window for creating and populating tests.



To access	Use one of the following: ➤ Click the **Toolbox** tab in the left pane (Default layout). ➤ Select **View** > **Toolbox.**
Important information	➤ You drag activities from this tab into the canvas. ➤ The activity becomes a step in the test. ➤ Only available when viewing the **Toolbox** tab.
Relevant tasks	"Create the test flow" on page 33

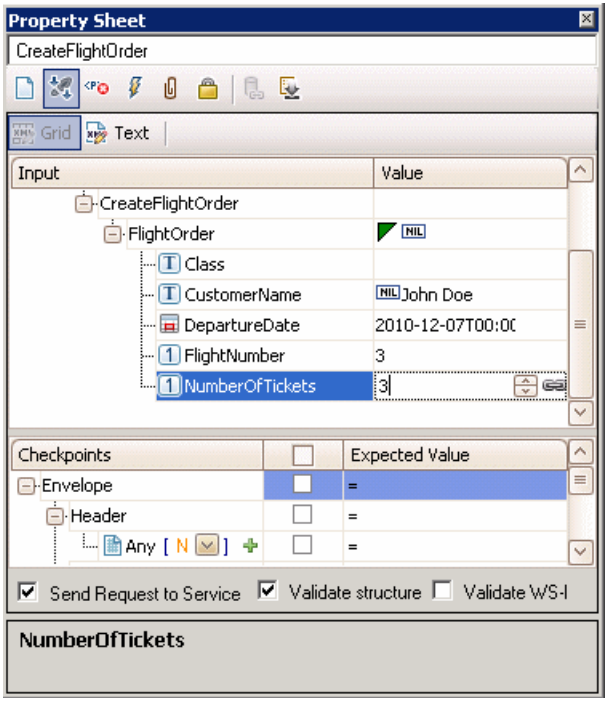
Toolbar Controls

The following toolbar controls elements are included in the Toolbox Palette:

Tree Elements	Description
	Expand All. Expands all nodes in the Activity list.
	Collapse All. Collapses all node in the Activity list.
	Security Settings. Opens the Security Settings for Port <Port_Name> dialog box. These settings apply to all operations in the port. For details, see the "Security Settings for Port <Port_Name> Dialog Box" on page 212. **Note:** Only available when selecting the Port node of a Web service.
	Add Resource. Adds a resource under the selected node. **Note:** Located adjacent to the **Filter** box, for the REST Services node only.
	Add Method. Adds a method under the selected node. **Note:** Located adjacent to the **Filter** box, for the REST Services node only.
	Delete. Delete a service, resource, or method. **Note:** Located adjacent to the **Filter** box, for the REST Services node only.
	Rename. Rename a service, resource, or method. **Note:** Located adjacent to the **Filter** box, for the REST Services node only.
<Filter box>	Filters the toolbox display by the entered text.
<activity list>	A tree hierarchy of built-in and custom activities. You drag activities into the canvas to create test steps. For details, see Chapter 2, "Service Test Activities."

Property Sheet

The Property Sheet displays a series of tabs showing the general properties, schemas, and event information for the activity selected in the canvas.



To access	Select **View** > **Property Sheet.**
Important information	➤ Only active when the canvas is visible. ➤ The toolbar displays different buttons based on the type of activity or view. ➤ The buttons to the left of the horizontal divider, are **View** buttons, such as **General** or **Events** views. ➤ The buttons to the right of the horizontal divider, are **Action** buttons, such as **Data-Drive** and **Load XML**. These differ based on the view. For details, see the specific view.
Relevant tasks	➤ "How to Populate a Test" on page 32 ➤ "How to Set Security for a Web Service on the Port Level" on page 192

The following elements are included (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
	Test Settings view. Java related settings that apply to all test steps. This is only visible when clicking in the canvas, outside of the Test Flow. For details, see "Test Settings View" on page 52.
	Service Test test view. The path and properties of the Service Test test to be called from this step. This is only visible when selecting a **Call Service Test test** activity.
	General view. Shows the general properties of the selected element. By default it shows the **Step ID** as it appears in the Toolbox, and the **Name** as it is shown in the canvas. For details, see "General View" on page 54.

UI Elements	Description
	Input/Checkpoints view. The input properties and the checkpoint values for those properties. The top section shows the input properties; the bottom shows the checkpoints. **Tip:** A triangular icon indicates an optional element. Toggle the triangle to include or exclude an element. A filled-in triangle indicates an included element. **For REST** and **Custom Code**: Use the Add button to add array elements. For general information about this pane, see "Input/Checkpoints View" on page 55. For information about specific input and output properties, see Chapter 2, "Service Test Activities."
	SOAP Fault view. The SOAP fault settings for the Web Service steps.
	HTTP view. The body of the input request and the checkpoints (HTTP activities and REST services only). For details, see the "HTTP View" on page 29.
	Multipart view. Allows you to configure multipart HTTP requests. For details, see the "Multipart View" on page 29.
	REST HTTP Input view. The HTTP input elements for. REST services. For details, see the "HTTP - General Properties" on page 106.
	Events view. Lets you select a handler function for a step's events, or to open an editor to write code describing what to do when the event occurs. **Tip**: The **Create a default handler** option lets you create a default event handler. For details, see the "Events View" on page 65.

UI Elements	Description
	Attachments view. A list of attachments associated with the Web Service step. For details, see "Attachments View" on page 66. **Note:** This view is only available for Web Service steps.
	Security Settings view. Allows the setting of Security Scenarios and security tokens for the Web Service. For details, see "Security Settings View" on page 67. **Note:** This view is only available for Web Service steps.
	Test Variables view. Displays the User and Test variables, and allows you to create Test Variable profiles. For details, see "Test Variables View" on page 62. **Note:** This view is only available when selecting the **Start** or **End** steps, or by clicking in the canvas outside of the Test Flow.
	Data Sources view. Allows you to select data sources for the loop, and set their policies. For details, see the "Data Sources View" on page 68. **Note:** This view is available when selecting the Test Flow frame or a loop's frame in the canvas.
	Data Source Properties view. Allows you to define and edit data relations between tables. For details, see the "Data Source Properties View" on page 68. **Note:** This view is only available when selecting a data source node in the **Data Window** tab.
	Database Data Source Properties view. Allows you to view and edit the database connection string and SQL statement. For details, see the "Database Data Source Properties View" on page 30. **Note:** This view is only available when selecting a database type data source in the **Data Window** tab.

UI Elements	Description
<data grid>	Information about the step selected in the canvas, depending on the view: ➤ **General view.** Activity name and comments. ➤ **Input/Checkpoints view.** Properties, checkpoints, and their values. ➤ **Events view.** A list of the events and their handlers. ➤ **Data Source view.** A list of data sources for the selected step.
Comments	An editable note describing the purpose of the step. Service Test also places the comment in the test report.

Test Settings View

Test settings apply to all the activities in the test. The **Test Settings** view's user interface elements are described below.

Load Settings

UI Elements	Description
Load enabled (read-only)	Indicates whether the test is enabled for load testing. To enable load testing, select **Test** > **Enable Test for Load**. **Note:** Once a test is enabled for load testing, you cannot disable it.

JVM (Java Virtual Machine) Settings

UI Elements	Description
Classpath	The vendor implementation of JMS classes together with any other required supporting classes, as determined by the JMS implementation vendor.
Additional VM Parameters	Extra parameters to send to the JVM such as Xbootclasspath, and any parameters specified by the JVM documentation.

JMS Settings

UI Elements	Description
JNDI initial context factory	The fully qualified class name of the factory class that will create an initial context. Provides a list of context factories and allows manual entries.
JNDI provider URL	The URL of the service provider. For example: Websphere - iiop://myserver:myport
JMS connection factory	The JNDI name of the JMS connection factory. This setting is unique per test.
JMS security principal	Identity of the principal (for example the user) for the authentication scheme.
JMS security credentials	The principal's credentials for the authentication scheme.
Number of JMS connections per process	The number of JMS connections each execution process creates. The default is 1, and the maximum is 50. The fewer connections you have, the better your performance.
Received message timeout options	The timeout for received messages. The default is **No wait**. ➤ **Indefinite wait.** Wait as long as required for the message before continuing. ➤ **No wait.** Do not wait for the Receive message, and return control to the script immediately. If there was no message in the queue, the operation fails. ➤ **User defined timeout.** Wait a specified number of seconds for the message. If it does not arrive, the operation fails. (default)

UI Elements	Description
User defined timeout	The amount of seconds to wait for the message before timing out. The default is 20 seconds.
Automatically generate selector	Generates a selector for the response message with the correlation ID of the request (**No** by default). Each JMS message sent to the server has a specific ID. Enable this option if you want Service Test to automatically create a selector that includes the message ID. **Note**: This option only affects the **Send and Receive Message from Queue** activity.

General View

The **General** view's user interface elements are described below. The properties differ based on the activity type.

For more information about activity-specific general properties, see Chapter 2, "Service Test Activities."

UI Elements (A-Z)	Description
Properties - default	➤ **Step ID.** A unique ID for the step. ➤ **Name.** Step name as it should appear in the canvas.
Properties - Service Test test/ QTP test / VuGen script	**Script Path.** The path of the Service Test or QuickTest test, or VuGen (LoadRunner's Virtual User Generator) script to run.
Properties - Service Test test	**Result Directory.** The path of the results relative to the solution directory. **Default:** RunStReport
Comment	An editable note describing the purpose of the step. Service Test also places the comment in the test report.

Input/Checkpoints View

The **Input/Checkpoints** view's user interface elements are described below (unlabeled elements are shown in angle brackets).

For more information about activity-specific properties, see Chapter 2, "Service Test Activities."

UI Elements	Description
<**properties list**>	A list of all the properties. The fields differ per step type: ➤ For the Start step: **Test Input Parameter** ➤ For the End step: **Test Output Parameter** ➤ For most activities: **Input** properties ➤ For SOAP Requests (under Web Services category), tools to work with a schema file. ➤ For loops, the loop type and properties. See "Input Properties - Loop" on page 95.
<**checkpoint list**>	A list of the step's output parameters, displaying the following columns: ➤ **Checkpoints.** A list of the output parameters. ➤ **Validate**. When checked, validates the current output parameters when running the step. When unchecked, ignores the parameters. ➤ **Value**. The expected value for the output parameter. Provides a drop-down list for comparison operators, number scrolling, and boolean values. You can manually enter the expected value into the cell.

UI Elements	Description
<context menu>	Provides shortcuts for including properties and setting their values: ➤ **Set Auto-value**. Inserts a sample value for the argument, based on its data type. ➤ **Include All.** Includes all properties. ➤ **Exclude All.** Excludes all properties. ➤ **Copy XPath**. Copies the XPath expression of the value to the clipboard. ➤ **Link to a data source.** Opens the Select Link Source dialog box. ➤ **Clear cell contents.** Clears the contents of the cell. **Tip:** Right-click in **Values** column to see the menu.
Checkpoint properties	Additional checkpoint properties: ➤ **Trim whitespace (from start and end of string).** Removes whitespace before and after the text. ➤ **Ignore case.** When looking for a match, ignores the case. ➤ **Stop test if checkpoint fails.** Exits the test run if the checkpoint fails. You set this individually for each checkpoint. **Note:** You must click on a row in the Checkpoint section to see these properties. The first two properties are only available for string data types.
Value column	The property value as a constant value or a link to a data source. An icon adjacent to the value provides information about the property, such as read-only, data type, optional, and allows you to choose values when appropriate. For a list of the most common icons, see the "Value Column Icons" on page 58. **Tip:** Click inside a row to display the relevant icons. **Note:** To view the buttons and icons, click within a row of the **Value** column.

UI Elements	Description
Send request to service	Sends the step's request to the service when running the test. This is the normal behavior for Web Services and most types of actions. It is activated by default. To send messages using JMS transport, clear this check box. Link a subsequent JMS step to the output of this step, to obtain the value of a JMS key.
Validate Structure	Checks that the service is in conformance with the schema defined for the SOAP XML response—either for the response defined in the operation or a SOAP Fault response. The Run Results Viewer indicates whether or not the validation succeeded. **Note:** This option is available only when the **Send request to server** option is enabled. **Tip**: If you modify the elements in the response, this will not affect the response schema. For example, adding array elements, selecting specific Choice elements, changing the derived type, and so forth, only affect the response—not the schema. To validate the element values, specify expected values in the **Value** column or use an XPath expression.
Validate WS-I	Checks the SOAP response's compliance against the WS-I standards. The Run Results Viewer indicates whether the SOAP complies with WS-I. It also provides a View Report link that opens the WS-I Validation report in a separate window.

UI Elements	Description
Generate Output	Retrieves table data from the database and constructs an array with the current table structure. If you enabled **Generate output** in the Query Builder, you do not need to generate the output again. For details, see "Query Builder Dialog Box" on page 118. **Note:** Only relevant for a **Database** > **Select Data** step.
Manage Columns	If the table structure changed since you generated the output, the Manage Columns dialog box lets you manage the columns. This is common when columns names were a parameter, or if a column was deleted from the table. **Note:** Only relevant for a **Database** > **Select Data** step.

Value Column Icons

The following buttons represent information, drop-down lists, or actions within the **Input/Checkpoints** view's **Value** column:

UI Elements	Description
	Include argument in the request. Toggle to clear the triangle and exclude the argument.
	Include an array element. Toggle to clear the box and exclude the array element.
	Set to NIL. Toggle to clear the icon and remove the NIL value assignment.
	Read-only node. Values that cannot be changed such as properties linked to a data source other than a constant value, or nodes with read-only attribute in the activity signature, first HTTP header, and so forth.
	Warning. A warning related to the data source. For example, **The data types of the link source and link destination do not match**.

UI Elements	Description
	➤ For **Input properties**: A drop-down list of possible values. For example, for boolean values it provides a list of values for boolean type data: **true**, **false**, **0**, or **1**. For date types, it opens a calendar. This drop-down arrow is located adjacent to the **Link to a data source** button . ➤ For **Checkpoints**: A drop-down list of the relevant comparison operators such as: =,**!=**, >, >=, <, <=, **Starts**, **Ends**, **Contains**, or **Regex**. The greater than operator, >, when used with strings, indicates that it appears later in the alphabet.
	Scroll. A scroll control for integer data types.
	Browse. Allows you to locate a file or folder for a **File System** type activity. For the **Open Connection** step, this opens the Connection Builder dialog box. For details, see "Connection Builder Dialog Box" on page 117.
	Link to a data source. Opens the Select Link Source Dialog Box allowing you to select values for the property from a data source. For details see "Select Link Source Dialog Box" on page 305.
	Display list of outgoing links. Shows a list of all input properties that link to this output property. For details, see "Outgoing Links" on page 268.

Action Buttons - Input/Checkpoints View

The following buttons represent actions within the **Input/Checkpoints** view:

UI Elements	Description
	Data Drive Entire Step. Data-drives all values for the step. It adds data expressions to the **Value** columns of all input properties. For details, see "How to Data Drive a Test Step" on page 291.
	Load from Replay. Loads XML data in order to populate a service's schema with values. For details, see "How to Run a Test" on page 243.
	Select Java File. Opens the Call Java Class Settings Dialog Box for setting additional classpaths for the call. For details, see the "Call Java Class Settings Dialog Box" on page 124.
	Add New Input/Output Property. Opens the Add New Input/Output Property dialog box, allowing you to define a new property and its data type. **Note:** This only applies to the Custom Code/Property steps, REST steps, and the **Start/End** steps, or when selecting the entire canvas. For details, see "Coding Service Test Events" on page 379.
	Remove Property. Deletes the selected property from the list. **Note:** This only applies to the Custom Code/Property steps, REST steps, and the **Start/End** steps, or when selecting the entire canvas.
Select QTP/ST Test	For **Call QTP Test** and **Call Service Test** steps: Invokes the Open Test dialog box for selecting the test to call from this step.
Update QTP/ST Test	For **Call QTP Test** and **Call Service Test** steps: Updates the QTP or Service Test test from its original location.

SOAP Fault View

The **SOAP Fault** view's user interface elements are described below. This view is only available for Web Service or SOAP activities.

UI Elements	Description
Fault is expected	Indicates that a SOAP fault is expected during the test run. This setting confirms that the application did not perform a task that it was not designed to perform.
XML tab	The SOAP envelope, containing the Header and Body of the SOAP message. In this section you enter the expected response. You can define **Any** type elements or values for the following properties: ➤ **faultcode.** A status code indicating that the SOAP request was invalid. ➤ **faultstring.** A response string indicating that the SOAP request was invalid. ➤ **faultactor.** An indication of the source of the fault.
XPath tab	A list of XPATH expressions used to evaluate the SOAP response. ➤ Adds a line for an XPATH expression. ➤ Removes the selected entry.

For more information about SOAP faults and how they can be used for Negative testing, see "Negative Testing" on page 151.

For task details, see "Configure SOAP Fault information - optional" on page 35.

Test Variables View

The **Test Variables** view's user interface elements are described below (unlabeled elements are shown in angle brackets). This view is available only when selecting the canvas, **Start** step, or **End** step.

UI Elements	Description
User variables	A list of the user variables for all profiles. When enabling the **Compare Profiles** option, the grid displays the variable values for each profile in separate columns.
System variables	A read-only list of common system variables and their values: ScenarioID, SystemTempDir, GroupName, TestDir, TestName, LocalHostName, OS, OSVersion, ProductDir, ProductName, ProductVer, and UserName.

Action Buttons - Test Variables View

UI Elements	Description
	Add New User Variable. Adds a new user variable to the all profiles.
	Remove User Variable. Deletes the selected user variable from all profiles.
	Add New Profile. Adds a new User Variable profile to the list.
	Edit Active Profile. Opens the Manage Profiles dialog box that lets you to remove or rename profiles.
	Compare Profiles. Displays the profiles side-by-side for comparison.
<Profile list>	A list showing the existing User Variable profiles.

REST HTTP Input View

For REST Services, you can set the method's HTTP settings and use them as a template for that method. The **REST HTTP Input** view's user interface elements are described below.

UI Elements	Description
Input	The REST Input properties: ➤ **Endpoint Address**. The endpoint URL for the REST service. ➤ **HTTP Method.** GET, POST, PUT, DELETE, TRACE, OPTIONS, HEAD, and CONNECT. ➤ **HTTP Version.** 1.0 or 1.1 ➤ **Request Headers.** Keys and their values, and arrays.
Checkpoints	The REST output properties for validation: ➤ **HTTP Version.** 1.0 or 1.1 ➤ **Status Code**. An integer indicating the status code. ➤ **Status Description**. A textual description of the status. ➤ **Response Body**. The body returned by the server. **Tip**: To validate a property, make sure to select its check box.

HTTP View

The **HTTP** view's user interface elements, for REST Services and HTTP activities, are described below (unlabeled elements are shown in angle brackets).

UI Elements	Description
	Clear. Clears the contents of the: ➤ Request Body (upper pane) ➤ Checkpoint keys (lower pane)
	For **Text** body types, imports a file containing the body of the request. For **XML** body types, imports a schema file.

UI Elements	Description
	Loads an XML file with Body request values, that correspond to the loaded schema. (**Body: XML** only)
	Browse. Allows you to load a non-XML file containing the request body (**Body: File** only)
<checkpoint list>	A list of the checkpoints keys and their values in the bottom pane.
<request body>	The body of the request loaded through a file, pasted from a clipboard (**Edit** > **Paste**), or manually entered (POST form)
Body (top)	The data type of the request body. **Text**, **XML**, **File**, or **Post Form.** **Default:** Text
Body (bottom)	The data type of the response body. **Text** or **XML.** **Default:** Text

Multipart View

For HTTP steps, this view allows you to configure multipart requests. Multipart messages are sent with the HTTP request and consist of two or more header/body sets.

The **Mulitpart** view's user interface elements are described below.

UI Elements	Description
Enable Multipart	Enables the **Input/Checkpoints** view for multipart HTTP requests.

For details about the input and output properties, see "Network Activities" on page 106.

Events View

The **Events** view's user interface elements are described below (unlabeled elements are shown in angle brackets).

UI Elements	Description
<Events list> - default	A list of events for the activity: ➤ **CodeCheckPointEvent.** Triggered when the activity is executed. ➤ **AfterExecuteStepEvent.** Triggered after the activity was executed. ➤ **BeforeExecuteStepEvent.** Triggered before the activity is executed. **Tip:** For details about an event, select it and choose **Create a default handler**. Refer to comments in the template.
<Events list> - Web Services	➤ AfterGenerateRequest ➤ AfterProcessRequestSecurity ➤ AfterProcessRequestAttachments ➤ OnSendRequest ➤ OnReceiveResponse ➤ BeforeProcessResponseAttachments ➤ BeforeProcessResponseSecurity ➤ BeforeSaveResponse ➤ BeforeApplyProtocolSettings For details about Web Services, see "Defining Services" on page 147.
Handler	The handler to call when the event is detected. Choose **Create a default handler** if no handler is available. This opens the **TestUserCode.cs** tab for editing the handler template.

For details about writing event handlers, see "Specify an event handler - optional" on page 36.

Attachments View

The **Attachments** view's user interface elements are described below (unlabeled elements are shown in angle brackets).

UI Elements	Description
<Input Attachment>	Input attachments for the current Web Service step: ➤ **Type.** Attachment type: NONE, DIME, or MIME. ➤ **Attachments.** A list of the input attachments and their properties: ➤ **Origin**. The file to send as an attachment. ➤ **Origin Type**. The attachment type. **File** is default. ➤ **Content Type**. The attachment's content type: application/zip, image/gif, image/ief, image/jpeg, image/png, image/tiff, text/plain, text/richtext, text/html, or text/xml. ➤ **Content ID**. A unique ID for the attachment or **Auto** for an ID generated by Service Test. **Note:** To add input attachments, click the **Add** button in the parent node.
<Output Attachment>	The server response saved as an attachment. ➤ **Attachments.** A list of the output attachments and their properties to validate: ➤ **Content**. The file content's checksum value. The checksum is computed by applying the MD5 hash function to the file. ➤ **Content Type**. The attachment's content type. ➤ **Content ID**. The unique ID of the attachment. **Note**: To add output attachments, click the **Attachments** row in the Checkpoints pane, and click **Add** to add an array element. (You may need to expand the column width to access the button). **Tip:** To validate an output attachment, select the check box adjacent to the elements you want to validate. Received attachments are saved in the test's folder, with **bin** extensions.

For task details, see "Add attachments to the test - optional" on page 34.

Security Settings View

Using the Property Sheet's Security Settings view, you can configure security settings for a specific step. To configure security for an entire port, see the "Security Settings for Port <Port_Name> Dialog Box" on page 212.

For information about choosing a scenario, see "Setting Security Overview" on page 182.

For task details, see "How to Set Security for a Web Service on the Port Level" on page 192.

The **Security Settings** view's user interface elements are described below.

UI Elements	Description
	Saves the security scenario settings to an **.stss** (Service Test Security Scenario) file, for use by other tests.
	Loads security settings from a previously saved stss file.
Use the port's security setting	Uses the port's security settings for the current step.
Edit port's settings	Opens the Security Settings dialog box for configuring the port security. For details, see the "Security Settings for Port <Port_Name> Dialog Box" on page 212.
Service Details	The type of Web service. After selecting a type, Service Test provides an interface for modifying the relevant security settings. The service types are: ➤ "Web Service Scenario" on page 214 ➤ "WCF Service (Custom Binding) Scenario" on page 221 ➤ "WCF Service (Federation) Scenario" on page 223 ➤ "WCF Service (WSHttpBinding) Scenario" on page 225 **Note:** Visible only when clearing the **Use the port's security settings** option.

Data Sources View

The **Data Source Properties** view's user interface elements are described below.

UI Elements	Description
Add	Opens the **Attach Data Source to Loop** dialog box for selecting data sources from the **Data Window** tab.
Edit	Opens the **Data Navigation** dialog box for setting the data policy—the way to use the data from the table. For user interface details, see "Data Navigation Dialog Box" on page 309.
Remove	Deletes the selected data source from the list—not from the Data Window.
Data Navigation Policies	A list of the data sources sorted by: ➤ **Data Source Name.** The name of the data source as it appears in the Data Window. ➤ **Policy.** A summary of the data navigation policy, for example: **Start at first row, froward one row, wrap around.**

For details about setting policies, see "How to Set the Navigation Properties" on page 290.

Data Source Properties View

The **Data Source Properties** view's user interface elements are described below.

UI Elements	Description
Add	Opens the **Define New Data Relation** dialog box. For user interface details, see "Define/Edit New Data Relation Dialog Box" on page 312.
Edit	Opens the **Edit Data Relation** dialog box. For user interface details, see "Define/Edit New Data Relation Dialog Box" on page 312.

UI Elements	Description
Remove	Deletes the selected data relation from the list.
Relations	A list of the data sources sorted by: ➤ **Child data source.** The sheet or table to use as a data source. ➤ **Primary Key.** A column in the parent data source to use as a primary key for the child relation. ➤ **Foreign Key.** The column in the child data source to use as a foreign key for the child relation.

For details about defining data relations, see "Create a new child relation" on page 282.

Database Data Source Properties View

The **Database Data Source Properties** view's user interface elements are described below.

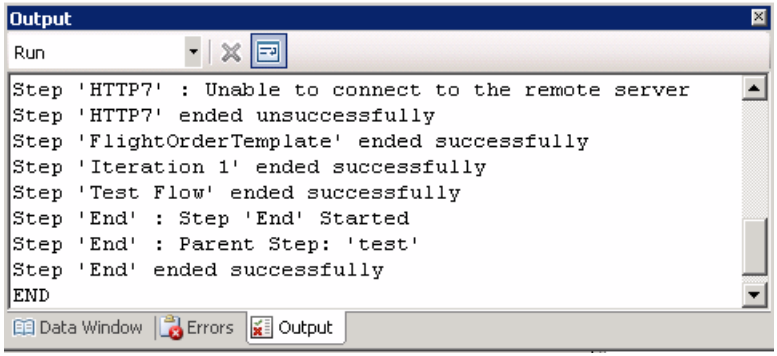
UI Elements	Description
Name	The name of the data source.
Connection string pane	The **Connection string** section contains the following elements: ➤ **Connection string.** A drop-down list of connection strings that were defined. ➤ Opens the Connection Builder dialog box. For details, see the "Connection Builder Dialog Box" on page 117. ➤ Opens a text editor for editing the connection string. ➤ **<Connection string text>.** A read-only window showing the selected connection string. ➤ **<Connection status>.** A string indicating whether or not there is an open connection to the database.

UI Elements	Description
SQL statement pane	The **SQL statement** section contains the following elements: ➤ **SQL statement.** A drop-down list of SQL statements strings that were defined in this test. ➤ Opens the **Query Designer**. For details, see the "Query Designer Dialog Box" on page 120. ➤ Opens a text editor for editing the SQL statement. ➤ **<SQL statement text>.** A read-only window showing the selected SQL statement. ➤ **<Query status>.** A string indicating whether or not there query was executed successfully.
Child relations pane	A list of the data sources sorted by: ➤ **Child data source.** The sheet or table to use for the data source. ➤ **Primary key.** A column in the parent data source to use as a primary key for the child relation. ➤ **Foreign key.** The column in the child data source to use as a foreign key for the child relation. For details about modifying or adding a relation, see the "Define/Edit New Data Relation Dialog Box" on page 312.

For details about creating a database type data source, see "Add a Database data source" on page 279.

Output Tab

Enables you to view the output messages that were generated while compiling and running the test.



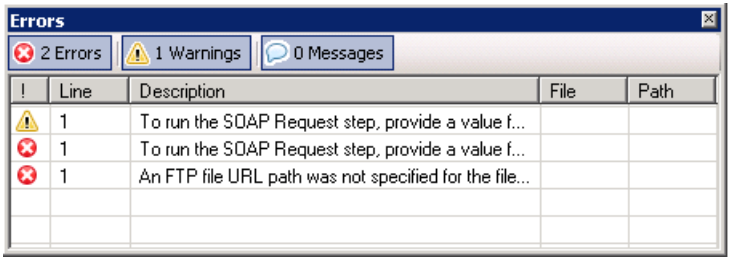
To access	Select **View** > **Output** (CTRL+ALT+O).
Important information	Click on a row to navigate to the code that generated the output message.
Relevant tasks	"How to Run a Test" on page 243

The following elements are included (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
	Clear All. Clears all the information in the message window.
	Toggle Word Wrap. Wraps the text of each message onto the next line
<Show Output from>	A drop-down list for selecting the output to display: **Build**, **Design**, **Debug**, or **Run.** **Default:** Run
<message window>	Displays all build and debug information issued to the log during the test run.

Errors Tab

Enables you to view a list of the errors generated during the test run, and their level of severity: Message, Warning, or Error.



To access	Click **Errors** tab in bottom pane or select **View** > **Errors** or CTRL+ALT+K.
Important information	➤ Click on a row to jump to the line in the code that generated the error. ➤ Click any column header, for example **Line** or **File**, to sort by that criteria.
Relevant tasks	"How to Run a Test" on page 243
See also	"Alerts" on page 20

The following elements are included:

UI Elements	Description
2 Errors	Shows or hides the errors detected during the test run.
1 Warnings	Shows or hides the warnings detected during the test run.
0 Messages	Shows or hides the informational messages detected during the test run.

UI Elements	Description
! (Exclamation Point)	Message Type: ➤ Error ➤ Warning ➤ Informative message
Description	Description of the error, warning or message and advice on how to fix the problem.
File	The file that generated the error, warning, or message.
Line	The line number in which the error was detected.
Path	The complete path of the file that generated the error.

Troubleshooting and Limitations - General

- It is not recommended to open two instances of Service Test on the same machine.
- On some operating systems or after installing certain Windows service packs or updates, you may be unable to view the content of the Help files. **Workaround**: Select the CHM file in the product's help folder, and click **Properties** from the shortcut menu. Select **Unblock**. For details, see http://support.microsoft.com/kb/902225.
- XPATH aggregate functions are not supported.
- The Run Results dashboard does not show iteration statistics.
- Opening file§ on a remote shared resource is not supported.
- The **Test Variables** view shows empty values for the **TestDir**,**TestName**, and **SystemTempDir** system variables. You can access the **TestDir** and **TestName** variables from the Select Link Source dialog box or through event handler code.

2

Service Test Activities

This chapter includes:

Concepts

➤ Activity Overview on page 76

Tasks

➤ How to Use Date and Time Activities on page 77

➤ How to Send an Multipart HTTP Request on page 80

➤ How to Execute Database Commands or Retrieve Data on page 81

➤ How to Validate an XML file on page 84

➤ How to Transform an XML File on page 86

➤ How to Implement UFT Testing on page 88

➤ How to Create a Call to a Java Class on page 91

References

➤ The Service Test Activities on page 93

➤ Activity-Specific Dialog Boxes on page 116

Troubleshooting and Limitations - Activities on page 126

Concepts

Activity Overview

You create a test by dragging and dropping **activities** from the Service Test Toolbox into a canvas. The Toolbox provides a collection of built-in activities for functional testing in areas such as REST, Web Services, JMS, and HTTP. You can add more activities to the toolbox by importing WSDLs or providing other contract definitions.

You can also create new custom activities, using the extensibility API provided with Service Test. For details, see "Extensibility in Service Test" on page 403.

For general information about the Service Test interface and creating tests, see Chapter 1, "Working with Service Test."

Tasks

How to Use Date and Time Activities

The following describes various tasks that you can perform with the Date/Time activities, such as incrementing a date or finding a differential.

This task includes:

- "Increment a date and time" on page 77
- "Find the difference between two date/time expressions" on page 78
- "Format a date or time" on page 79

Increment a date and time

1. Drag the **Increment Date/Time** activity into the canvas.
2. Open the **Input/Checkpoints** view in the Property Sheet.
3. Set the Input property—**Original Date/Time**. Click the arrow in the **Value** column to open the calendar. Click **Today** or select another date. You can also link to the output of another step using the **Link to a data source** button.
4. To set a time unit such as hour, minute, seconds, or milliseconds, edit the expression in the **Original Date/Time** row. For milliseconds, add a colon followed by a three digit millisecond value. For example, 2011-01-01T01:30:00:040
5. Set the Input property—**Unit**. Select a unit by which to increment: Milliseconds, Seconds, Minutes, Hours, Days, Months, or Years.
6. Set the Input property—**Amount**. Use the scroller to set the amount of units to increment.
7. Select the **Result** check box in the **Checkpoint** pane to verify the response. Use the calendar to select the expected date. If you need to set an expected value for a time unit, modify the time units manually.

Find the difference between two date/time expressions

1. Drag the **Date/Time Difference** activity into the canvas.
2. Open the **Input/Checkpoints** view in the Property Sheet.
3. Set the Input property— **Date/Time A**. Click the arrow in the **Value** column to open the calendar. Click **Today** or select another date.
4. To set a time unit such as hour, minute, seconds, or milliseconds, edit the expression in the **Date/Time A** row. For milliseconds, add a colon followed by a three digit millisecond value. For example, 2011-01-01T01:30:00:040
5. Set the Input property—**Date/Time B**. Click the arrow in the **Value** column to open the calendar. Select a date for the second expression.

   To set a time unit such as hour, minute, seconds, or milliseconds, follow the instructions in the above step.
6. Set the Input property—**Unit**. Select the unit by which you want to measure the difference: Milliseconds, Seconds, Minutes, Hours, Days, Months, or Years.
7. To verify the response, select the **Difference** check box in the **Checkpoint** pane. Use the scroller to specify the expected difference.

Format a date or time

1. Drag the **Format Date/Time** activity into the canvas.
2. Open the **Input/Checkpoints** view in the Property Sheet.
3. Set the Input property—**Date/Time**. Click the arrow in the **Value** column to open the calendar. Click **Today** or select another date.
4. To set a time unit such as hour, minute, seconds, or milliseconds, edit the expression in the **Date/Time** row. For milliseconds, add a colon followed by a three digit millisecond value. For example, 2011-01-01T01:30:00:040
5. Set the Input property—**Format**. Select a format type for the expression. If your expression contains milliseconds, select an expression with a time format and add a colon followed by fff. For example, dd/MM/yyyy hh:mm:ss:fff
6. Select the **Result** check box in the **Checkpoint** pane to verify the response's format. Type the expected expression. If you need to set an expected value for a time unit, modify the time units manually.

How to Send an Multipart HTTP Request

The following describes how to send an HTTP request that uses multiple parts. For details about the properties, see the "Multipart View" on page 64.

This task includes:

- "Add an HTTP Step" on page 80
- "Set the General properties" on page 80
- "Set the properties for the first part" on page 80
- "Set the properties for the next parts" on page 80

1 Add an HTTP Step

Drag the **Network** > **HTTP** activity into the canvas.

2 Set the General properties

Open the **General** view in the Property Sheet. Set the properties as described in the "General View" on page 54.

3 Set the properties for the first part

Open the **Input/Checkpoints** view in the Property Sheet. Set the properties for the first part as described in the "Network Activities" on page 106.

4 Set the properties for the next parts

a Open the **Multipart** view in the Property Sheet.

b Select the **Enable Multipart** option.

c Set the multipart type and global header information.

d Add elements to the Parts array corresponding to the number of parts.

e Set the properties for the parts as described in the "Multipart View" on page 64.

f If you want to validate a response, select the box in the **Validate** column and provide the expected values.

How to Execute Database Commands or Retrieve Data

The following describes how to use database activities. For details about the database activities, see "Database Activities" on page 99.

This task includes:

- "Open a connection" on page 81
- "Add a Select Data step" on page 82
- "Add an Execute Command step" on page 82
- "Add database transaction activities" on page 83
- "Add a Close Connection step" on page 83

Open a connection

This step is mandatory for all of the following steps.

1 Drag the **Database** > **Open Connection** activity into the canvas.

2 Open the **Input/Checkpoints** view in the Property Sheet.

3 Select the Input property—**Connection string**.

4 Click the **Connection Builder** button in the right side of the **Value** column. The Connection Builder dialog box opens. For details, see "Connection Builder Dialog Box" on page 117.

5 Paste in an existing string into the text area or click the **Connection Builder** button to open Microsoft's Data Link Properties dialog box. Provide the required information and click **OK**. For details, click the dialog box's **Help** button.

Note: The database connection must be an OLE DB type.

---

6 If you want to validate the connection, set the checkpoint properties - **Results** and **Results message**.

Add a Select Data step

1 Drag the **Database** > **Select Data** activity into the canvas, below an **Open connection** step.

2 Open the **Input/Checkpoints** view in the Property Sheet.

3 Set the Input property—**Connection**. Click the **Link to a data source** button by the right side of the **Value** column. In the Link to Source dialog box, select an earlier **Open connection** step in the left pane. In the right pane, select the step's result.

4 Set the Input property—**Query string**. Click the **Browse** button in the right side of the **Value** column to open the Query Builder. Paste in a query or use the Query Designer. For details, see the "Query Builder Dialog Box" on page 118.

5 Optionally, set the **Timeout**, the maximum time allowed for the database response. To allow an unlimited amount of time, enter 0.

Add an Execute Command step

1 Drag the **Database** > **Execute Command** activity into the canvas, below the **Open Connection** step.

2 Open the **Input/Checkpoints** view in the Property Sheet.

3 Set the Input property—**Connection**. Click the **Link to a data source** button by the right side of the **Value** column. In the Link to Source dialog box, select an earlier **Open connection** step in the left pane. In the right pane, select the step's result.

4 Set the Input property—**Command**. Click the arrow to open an edit box, Paste in a command and click **OK**.

5 Optionally, set the **Timeout**, the maximum time allowed for the database response.

Add database transaction activities

You can optionally add transaction activities to your test.

1. Drag the **Database** > **Begin Transaction** activity to the canvas after an **Open Connection** step and before the steps to be included in the transaction.
2. Drag a **Database** > **Commit Transaction** activity to the canvas after the database steps that make up the transaction.
3. Drag a **Database** > **Rollback Transaction** activity to the canvas after the database steps that make up the transaction.

Tip: A common use is to insert a **Condition** step to check a value after the database commands. Set one branch with **Commit Transaction** and the other branch to **Rollback Transaction**. For details, see "Flow Control" on page 19.

Add a Close Connection step

1. Drag the **Database** > **Close Connection** activity on to the canvas, after all of the database steps.
2. Open the **Input/Checkpoints** view in the Property Sheet.
3. Set the Input property—**Connection**. Click the **Link to a data source** button in the right side of the **Value** column In the Link to Source dialog box, select the **Available steps** option. In the left pane, select the earlier step, **Open Connection**, In the right pane, select the **Connection** node.

How to Validate an XML file

This task describes how to validate the code in an XML string. The Validate XML activity evaluates an XML string and checks its compliance with an XSD schema.

Since the **Validate XML** activity reads a string, you need to specify the actual XML string—not just a file name. If the string is short, you can copy it into the **Value** column directly. For more complex XML content, you can read the XML file using a **Read from File** activity.

This task includes the following steps:

- "Provide the XML code to validate" on page 84
- "Add a Validate XML activity" on page 85
- "Connect the steps - if using Read from File" on page 85
- "Specify an XSD file" on page 85
- "Set a checkpoint - optional" on page 85
- "Run the test" on page 86

1 Provide the XML code to validate

To provide the XML code, you can paste an XML directly into the **Value** column, or read the contents from the file using the **Read from File** activity.

To enter XML code directly, copy the XML string from the source document and paste it into the **Value** column.

To read the XML from a file:

a Drag the **File System** > **Read from File** activity into the canvas.

b Open the **Input/Checkpoints** view in the Property Sheet. Browse to the **XML file** you want to validate.

2 Add a Validate XML activity

Drag the **XML** > **Validate XML** activity into the canvas. If you used a **Read from File** step, drag this activity beneath the step.

3 Connect the steps - if using Read from File

If you used a **Read from File** step to obtain the XML, follow these steps. Otherwise, skip to step 4.

a Open the **Input/Checkpoints** view in the Property Sheet. Click in the **Value** column of the **XML String** property and click the **Link to a data source** button.

b In the Link to Source dialog box, select the **Available steps** option. In the right pane, select the **Read from File** step's output property, **Content,** and click **OK**.

4 Specify an XSD file

a Select the **Validate XML** step in the canvas. Open the **Input/ Checkpoints** view in the Property Sheet. Browse to an **XSD file**.

b To import the XSD file into the test's folder, set the **XSD file to test folder** option to true. This is useful if you plan on saving the test on QC/ALM. Setting this option will make the XSD file available to anyone who loads the test.

5 Set a checkpoint - optional

If you want to verify the results:

a Click in the **Checkpoints** section.

b Select the **Valid** check box. Set the value to true to verify that the XML code complies with the XSD. Set the value to false to confirm that the XML code did not comply with the XSD.

c To check for a specific error, set the **Valid** value to false, and specify the expected error in the **Error** row. You can link to a data source containing the expected value.

6 Run the test

Run the test. The Output tab and Run Results Viewer will display the results and indicate whether the XML is in compliance with the XSD and details about the errors.

How to Transform an XML File

This task describes how to transform an XML file to a different structure based on an XSLT file.

Since the Transform XML activity reads a string, you need to specify the actual XML string—not just a file name. If the string is short, you can copy it into the **Value** column directly. For more complex XML content, you can read the XML file using a **Read from File** activity.

This task includes the following steps:

- "Provide the XML code to transform" on page 86
- "Add a Transform XML activity" on page 87
- "Connect the steps - if using Read from File" on page 87
- "Specify an XLST file" on page 87
- "Set a checkpoint - optional" on page 88
- "Run the test" on page 88

1 Provide the XML code to transform

To provide the XML code, you can paste an XML directly into the **Value** column, or read the contents from the file using the **Read from File** activity.

To enter XML code directly, copy the XML string from the source document and paste it into the **Value** column.

To read the XML from a file:

a Drag the **File System** > **Read from File** activity into the canvas.

b Open the **Input/Checkpoints** view in the Property Sheet. Browse to the **XML file** you want to transform.

2 Add a Transform XML activity

Drag the **XML** > **Transform XML** activity into the canvas. If you used a **Read from File** step, drag this activity beneath the step.

3 Connect the steps - if using Read from File

If you used a **Read from File** step to obtain the XML, follow these steps. Otherwise, skip to step 4.

a Open the **Input/Checkpoints** view in the Property Sheet. Click in the **Value** column of the **XML String** property and click the **Link to a data source** button.

b In the Link to Source dialog box, select the **Available steps** option. In the right pane, select the **Read from File** step's output property, **Content,** and click **OK**.

4 Specify an XLST file

a Select the **Transform XML** step in the canvas. Open the **Input/Checkpoints** view in the Property Sheet. Browse to an **XSLT file**.

b To import the XLST file into the test's folder, set the **XLST file to test folder** option to true. This is useful if you plan on saving the test on QC/ALM. Setting this option will make the XLST file available to anyone who loads the test.

5 Set a checkpoint - optional

If you want to verify the results, select the **Transformed XML** check box in the **Checkpoints** section and specify the expected result. You can link to a data source containing the expected value.

6 Run the test

Run the test. The **Output** tab and the Run Results Viewer will indicate whether the transform operation succeeded.

How to Implement UFT Testing

This task describes how to incorporate tests from other HP applications to provide a Unified Functional Testing solution (UFT). These applications include QuickTest Professional, Virtual User Generator, and Service Test. For details, see "UFT (Unified Functional Testing)" on page 31.

This task includes the following steps:

- "Prerequisites" on page 88
- "Add a QuickTest test activity - optional" on page 89
- "Add a LoadRunner script activity - optional" on page 90
- "Add a Service Test test - optional" on page 90

1 Prerequisites

Make sure you have installed the application whose test/script you want to call:

- **QuickTest test step.** Make sure you have QuickTest Professional 11.00 or later. Create the test in QuickTest Professional and run it at least once.
- **Virtual User Generator step.** Make sure you have HP LoadRunner 11.00 or later. Generate a test with the Virtual User Generator, or use a test that you enabled for load testing in Service Test.

Note: The first time you invoke QuickTest in a test step, you may need to wait several seconds for Service Test to invoke QuickTest and load the test.

---

2 Add a QuickTest test activity - optional

a In Service Test, expand the **Unified Functional Testing** node in the Toolbox palette. Drag the **Call QuickTest Professional Test** activity into the canvas.

Tip: Do not insert a call to a QuickTest test that contains a call to a Service Test test, as this can cause unexpected behavior.

---

b Open the Property Sheet's **Input/Checkpoints** view, and click the **Select QTP Test** button. Select a QuickTest Professional version 11.00 (or later) test.

c If the QuickTest test changed at its source, you can update it in Service Test. In the **Input/Checkpoints** view, click **Update properties**.

d Service Test imports the properties defined in the QuickTest test. You can edit the properties manually, data drive them, or load response values from a replay.

e Add other relevant steps to your test.

f Save and run the test.

3 Add a LoadRunner script activity - optional

a In Service Test, expand the **Unified Functional Testing** node in the Toolbox palette. Drag the **Call Virtual User Generator Script** activity into the canvas.

b Open the Property Sheet's **General** view, and click the Browse button in the **Script Path** row. Select a VuGen (Virtual User Generator) **.usr** script file.

c Add other relevant steps to your test.

d Save and run the test.

4 Add a Service Test test - optional

a Expand the **Unified Functional Testing** node in the Toolbox palette. Drag the **Call Service Test test** activity into the canvas.

b Open the Property Sheet's **General** view., and click the **Select ST Test** button. Select a test version 11.00 (or higher).

c To update a test that has already been loaded, click **Update ST Test**.

d To specify a custom directory for the results, click the **Browse** button in the **Results Directory** row in the **General** view tab. Select a directory for the results.

e Add other relevant steps to your test.

f Save and run the test.

How to Create a Call to a Java Class

The Call Java Class activity lets you to add Java steps to your test script. This feature allows you to incorporate existing Java code into your test.

This task describes how to create the Java call and includes the following steps:

- "Set the global Java settings - optional" on page 91
- "Implement the Service Test Java interface" on page 91
- "Package your custom step - optional" on page 92
- "Add a Call Java Class activity" on page 92
- "Open the Java Class Setting dialog box" on page 92
- "Set the Java Call properties" on page 92
- "Provide Input properties" on page 92

1 Set the global Java settings - optional

To set global VM (Virtual Machine) settings, click in the canvas outside of the Test Flow, and open the **Test Settings** view in the Property Sheet. For details, see the "Test Settings View" on page 52.

2 Implement the Service Test Java interface

Run **javac.exe** to implement the Service Test interface. The interface, **ServiceTestCall.java**, is located in the **<ST_INSTALL_HOME>\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java** folder.

This interface provides the essential information to Service Test: input properties, output properties, and a point of entry.

For example, the following code implements the "Hello World" sample provided with Service Test:

```
javac "C:\Program Files\HP\Service Test\addins\ServiceTest\JavaCall\Java
Interface\src\hp\st\ext\java\*.java" "C:\Program Files\HP\Service Test
\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java\sample\*.java"
```

3 Package your custom step - optional

Package your java classes into a **jar** file. This is optional, since you can also provide a package root for the class.

4 Add a Call Java Class activity

In Service Test, expand the **Java** node in the Toolbox palette, and drag the **Call Java Class** activity into the canvas.

5 Open the Java Class Setting dialog box

Click on the Java step in the canvas, and open the **Input/Checkpoints view** in the Property Sheet. Click the **Select Java File** button to open the Call Java Class Settings Dialog Box.

For general guidelines, see "Java Testing" on page 26.

6 Set the Java Call properties

In the Call Java Class Settings dialog box:

a Provide a classpath. If you packaged your Java step, click the **Jar** button and point to the **jar** file. Alternatively, click the **Package root** button to select a package root folder.

b Click the **Browse** button in the **Selected Java Class** box, to locate the class within the **jar** file or the folder. Make sure it is a class that implements the **ServiceTestCall** interface.

c To provide additional classpaths, click the **Jar** or **Folder** buttons in the **Additional Classpaths** section to Browse to a **jar** file or classpath folder. Click **Add** to move the contents into the list.

d Click **OK** to save the Java Call settings.

For user interface details, see the "Call Java Class Settings Dialog Box" on page 124.

7 Provide Input properties

In the Property Sheet's **Input/Checkpoints** view, provide values for the step's input properties.

References

The Service Test Activities

This section describes the following activity groups:

- REST Service Activities on page 94
- Flow Control Activities on page 95
- Miscellaneous Activities on page 96
- String Manipulation Activities on page 97
- System Activities on page 97
- Math Activities on page 97
- Date and Time Activities on page 98
- File System Activities on page 98
- Database Activities on page 99
- FTP Activities on page 103
- Network Activities on page 106
- Java Activities on page 109
- JMS Activities on page 110
- Load Testing Activities on page 113
- Unified Functional Testing Activities on page 114
- XML Activities on page 114
- Web Service Activities on page 115

REST Service Activities

This group lets you define REST services, and their resources and methods. Using the shortcut menu, you create resources and methods. You define the properties in the Property Sheet.

For REST Services, you can set the method's HTTP settings and use them as a template for that method.

For task details, see "How to Create a Simple Test for a REST Service" on page 158 or "How to Create a Template-Based REST Service" on page 160.

Input Properties - REST HTTP Activities

UI Elements	Description
Endpoint Address	The endpoint URL for the REST service.
HTTP Method	The HTTP method to use for the service call: GET, POST, PUT, DELETE, TRACE, OPTIONS, HEAD, or CONNECT.
HTTP Version	The HTTP version to implement: 1.0 or 1.1
Request Headers	Request header properties, such as keys and their values, and arrays.

Checkpoints - REST HTTP Activities

The following table list the REST output properties for validation. To validate a property, make sure to select its check box.

UI Elements	Description
HTTP Version	The implemented HTTP version.
Status Code	An integer indicating the status code.
Status Description	A textual description of the status.
Response Body	The body returned by the server.

Flow Control Activities

The following activities allow you to control the flow of the test using loops, conditions, breaks, and delays:

- **Condition.** Allows you to use a branching mechanism based on the value of a property.
- **Loop.** A loop frame whose properties can be set independently of the Test Flow. The loop types are **For loop**, **Do While loop**, and **For Each loop**.
- **Break.** Halts the test execution.
- **Continue.** Resumes the test execution after a break.
- **Sleep.** Pauses the test execution for a specified amount of time.

For details, see "Flow Control" on page 19.

Input Properties - Loop

The following iteration settings are available for the Test Flow or other custom loops:

UI Elements	Description
For Loop	Performs the Test Flow or custom loop the number of times that you specify in the **Number of Iterations** box.
Do While Loop - Use condition	Repeats the loop as long as the specified condition is met: ➤ **Variable.** The variable to evaluate. Use the Select Link Source button to enter a data source expression. ➤ **Operator.** A comparison operator relevant to the variable such as =, **!=**, **Contains**, **Starts** and **Regex**. ➤ **Value.** The value with which to compare the result.

UI Elements	Description
Do While Loop - Use event	Performs iterations until the event returns True. This mode is useful for complex conditions that can be defined in an event handler. When you select this option, the Input Properties grid opens. ➤ Click . to define properties. ➤ Click . to open the **Events** view. Click **Create a default handler** in the **Handler** column. ➤ Modify or add code to the event handler method that defines your condition.
For Each Loop	Performs an iteration for each element in the associated array or collection of objects. Data is selected using the **Select Link Source** button . **Note:** When you delete data from a data table, Service Test continues to run an iteration for that row, with empty values. To remove an entire row of a data, click the row and select **Delete** from the shortcut menu (only with Excel installations).

Miscellaneous Activities

This activity group provides access to the following activities:

- **Custom Code.** Allows you to program a step to execute your own activity. Use the **Add** button to define new input or output properties.
- **Set Test Variable.** Defines global variables for your test.
- **Run Program.** Invokes an application the Service Test machine.
- **End Program.** Closes an application that is running.
- **Report Message.** Sends a custom message to the Run Results.

String Manipulation Activities

This activity group provides access to the following string related activities:

- **Concatenate Strings.** Joins two strings.
- **Replace String.** Replaces part of a string with an alternate string.
- **Substring.** Extracts the characters between two specified locations of a string.
- **Trim String.** Removes whitespace characters from the beginning or end of a string.

System Activities

This activity group provides access to the following operating system activities:

- **Set OS Environment Variable.** Sets the value of an operating system variable.
- **Get OS Environment Variable**. Retrieves the value of an operating system variable.

Math Activities

This activity group provides access to the following math related activities:

- **Add.** Adds two numbers.
- **Subtract**. Subtracts one number from another.
- **Multiply.** Multiplies two numbers.
- **Divide**. Divides one number by another.

Date and Time Activities

This activity group provides access to the following date and time related activities:

- **Increment Date.** Adds date or time units to a date/time string.
- **Date/Time Difference**. Calculates the difference between two date/time strings.
- **Format Date/Time.** Formats a date/time string.

File System Activities

This activity group provides access to the following file related activities:

- **File Exists.** Searches for a specific file, or group of files, if you use a wildcard expression. It returns **true** if the file exists.
- **Folder Exists.** Searches for a specific folder. It returns **true** if the folder exists.
- **File Copy.** Copies a file to a new destination.
- **File Move.** Moves a file to another destination.
- **File Create.** Creates a new file.
- **File Delete.** Deletes a file.
- **File Rename.** Renames a file.
- **Write to File.** Writes to an existing or new file.
- **Read from File.** Retrieves text from a file.
- **Get Folder Contents.** Gets the contents of a folder—file names, folder names, or both.

Database Activities

This activity group provides access to the following database related activities:

- **Open Connection.** Opens a connection to a database using the specified connection string.
- **Close Connection.** Closes an open database connection.
- **Select Data.** Executes a SELECT type SQL statement that retrieves data.
- **Execute Command.** Executes an SQL statement that does not retrieve data from the database, such as UPDATE, DELETE, and so forth.
- **Begin Transaction.** Begins a database transaction.
- **Commit Transaction.** Commits a database transaction.
- **Rollback Transaction.** Rolls back a database transaction from a pending state.

Database Activities - Input Properties

To view the Input properties, click the **Input/Checkpoints** view button in the Property Sheet.

Open/ Close Connection

Property	Description
Connection String **(for Open Connection)**	An OLE DB database connection string. You can provide a value in one of the following ways: ➤ Type it in manually. ➤ Open the **Connection Builder** ... . For details, see the "Connection Builder Dialog Box" on page 117. ➤ Use the **Link to a data source** button to create a link to the string.
Connection **(for Close Connection)**	A link to the output of an **Open Connection** step. Use the **Link to a data source** button to create a link.

Select Data

Property	Description
Connection	A link to the output of a previously opened connection string. Use the **Link to a data source** button to create the link.
Query string	An SQL SELECT statement. You can provide a string in one of the following ways: ➤ Type it in manually. ➤ Open the **Query Builder** . For details, see the "Query Builder Dialog Box" on page 118. ➤ Use the **Link to a data source** button to create a link to the string.
Timeout	The maximum time to allow for executing the database command. A value of zero indicates no time limit. **Default:** 30 seconds

Execute Command

Property	Description
Connection	A link to the output of a previously opened connection string. Use the **Link to a data source** button to create the link.
Command	An SQL statement that does not retrieve data, such as UPDATE or DELETE. You can provide a string in one of the following ways: ➤ Type it in manually in the drop-down edit box. ➤ Use the **Link to a data source** button to create a link to the string.
Timeout	The maximum time to allow for executing the database command. A value of zero indicates no time limit. **Default:** 30 seconds

Open Transaction

Property	Description
Connection	A link to the output of a previously opened connection string. Use the **Link to a data source** button to create the link.
Isolation level	The isolation level of the transaction: Default, Chaos, Read Committed, Read Uncommitted, Repeatable Read, Serializable, or Snapshot. For details, see http://msdn.microsoft.com/en-us/library/system.data.isolationlevel.aspx.

Commit/Rollback Transaction

Property	Description
Transaction	A link to an open transaction. Use the **Link to a data source** button to create a link to the output of a **Begin Transaction** step.

Database Activities - Checkpoints

To view the checkpoint properties, click the **Input/Checkpoints** view button in the Property Sheet and click in the lower **Checkpoints** pane.

Open/ Close Connection and Transaction Activities

Property	Description
Result	Indicates whether the connection is active (true) or closed (false).
Result Message	A message indicating success or an informational message from the database.

Select Data

Property	Description
Count	The number of rows returned by the command.
Result	Indicates whether the operation has been successfully completed: true or false.
Result table	An array of returned rows. The row elements are column names. You can set an expected value for each column of each row. **Note:** In order to see the table column, you need to generate the output at least once. You can do this by selecting the **Generate output** option in the Query Builder, or by clicking the **Generate Output** button in the **Checkpoints** pane. **Tip:** Use the **Manage Columns** button to remove or replace columns when: ➤ The columns changed due to parameterization. ➤ A column was removed since the original query. ➤ You don't have access to database when designing the test.

Execute Command

Property	Description
Affected rows	The number of rows affected by the command.
Result	Indicates whether the command executed successfully: true or false.
Result Message	Success or upon failure, a message issued by the database indicating the nature of the failure.

FTP Activities

This activity group provides activities that allow you to perform FTP operations, such as **FTP Upload** and **FTP Download**.

- **FTP Download.** Downloads a file or directory from an FTP server.
- **FTP Upload.** Uploads a file or directory to an FTP server.
- **FTP Rename.** Renames an existing file or directory on an FTP server.
- **FTP File Delete.** Deletes a file on an FTP server.
- **FTP File Get Size.** Gets the size of a file on an FTP server.
- **FTP Directory Delete.** Deletes a empty directory on an FTP server.
- **FTP Directory Create.** Creates a new directory on an FTP server.
- **FTP Directory Get Content.** Lists the content of a directory on an FTP server.

FTP - General Properties

To view the General properties, click the **General** view button in the Property Sheet. The following table describes the FTP-specific properties.

Property	Description
Enable SSL	Indicates whether to establish a secure FTP connection: true or false.
Timeout	The FTP request timeout in milliseconds.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Client certificate	➤ . **Browse Certificates**. Opens the Select Certificate dialog box. To see the icon, click in the top level row of the **Client certificate** section, in the **Value** column. For details, see the "Select Certificate Dialog Box" on page 235. ➤ A Choice element indicating the source of the client certificate for secure FTP connections: ➤ **Use file system certificate**: The expanded node provides the full path of the certificate file. ➤ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. ➤ **Password.** The certificate's password.

FTP - Input Properties

To view the Input properties, click the **Input/Checkpoints** view button in the Property Sheet.

Property	Activity	Description
URL path	All except **FTP Upload**	The URL of the file or directory on the FTP server.
Directory URL path	**FTP Upload**	The URL in which the file or directory will be created.
User ID	All	The username for accessing the FTP server.
Password	All	The password for accessing the FTP server.
Local file path	**FTP Download** and **FTP Upload**	**FTP Download:** The location in which to download the file. **FTP Upload:** The location from which to upload the file.
New file name	**FTP Rename**	A new name for the file.
Directory name	**FTP Directory Create**	A name for the new directory.

FTP - Checkpoints

To view the checkpoint properties, click the **Input/Checkpoints** view button in the Property Sheet and click in the lower **Checkpoints** pane.

Property	Description
Result	An indicator whether the operation succeeded: true or false.
File size	The size of the specified file. (**FTP File Get Size** activity)
Result array	An array of the directory details for all directories under the specified URL path. (**FTP Directory Get Content** activity) The details include: **Name**, **Type**, **Flags**, **Owner**, **Group**, **Size**, and **Date created**.

Network Activities

This activity group provides activities that allow you to configure the transport method, such as **HTTP**.

HTTP - General Properties

To view the General properties, click the **General** view button in the Property Sheet.

Property	Description
Proxy	The settings for the proxy server hosting the service contract: **Server** (URL and port when required), **Username**, and **Password**.
Authentication	The user credentials settings for obtaining a service contract: **Username** and **Password**.
Connection type	The type of connection: **Keep-Alive** (default) or **Close**.
Timeout	The HTTP timeout in milliseconds.
Client certificate	➤ A Choice element indicating the source of the client certificate: ➤ **Use file system certificate**: The expanded node provides the full path of the certificate file. ➤ **Use Windows store certificate**: The expanded node lists the following properties **Store location**, **Store name**, **X509 find type**, and **X509 find value**. ➤ **Password.** The certificate's password.
Maximum automatic redirections	The number of times to attempt accessing a page, including redirection. The default is 3 times.
Allow redirections	Indicates whether to allow the step to redirect to another URL. Default is true.
Reuse cookies	Enables the reusing of cookies for the current step. Default is false.

HTTP - Input Properties

To view the Input properties, click the **Input/Checkpoints** view button in the Property Sheet.

Property	Description
URL	The URL of the HTTP request.
HTTP method	The request's HTTP method: GET, POST, PUT, DELETE, TRACE, OPTIONS, HEAD, and CONNECT.
HTTP version	The HTTP version: 1.0 or 1.1.
RequestHeaders	An array of the HTTP request header with **Key** and **Value** entries.

HTTP - Checkpoints

To view the checkpoint properties, click the **Input/Checkpoints** view button in the Property Sheet and click in the lower **Checkpoints** pane.

Property	Description
HTTP version	The HTTP version of the response.
Status code	The status code of the HTTP response.
Status description	A description of the HTTP status.
Response body as UFT-8 String	The body of the HTTP response in UFT-8 format.

HTTP - Multipart Input Properties

To view the HTTP multipart properties, open the **Multipart** view in the Property Sheet and select **Enable Multipart**. The view displays a hierarchy of the HTTP parts including the header and body, beginning with the second part.

You set the header information for the first part in the **Input/Checkpoints** view and its body in the **HTTP** view.

UI Elements (A-Z)	Description
Type	The multipart subtype to be used in the request's content-type header: Mixed, Digest, Alternative, Related, Report, Signed, Encrypted, FormData, or Parallel.
Headers	Global headers that apply to the entire message—not to a specific part. This field uses an array to display **Name** and **Value** properties.
Parts	An array of the parts to include with the HTTP request, beginning with the second part. Each part consists of: ➤ **Headers.** An array of headers for the part. This field uses an array to display the **Name** and **Value** properties. ➤ **Path.** The path of the part. The **Browse from file** button in the **Value** column allows you to load a non-XML file containing the request body.

HTTP - Multipart Checkpoints

The **Checkpoints** pane shows a hierarchy of the parts of the HTTP response, beginning with the second part. You set checkpoints for the first part in the **Input/Checkpoints** view.

UI Elements	Description
Type	The multipart subtype expected in the response: Mixed, Digest, Alternative, Related, Report, Signed, Encrypted, FormData, or Parallel.
Headers	Global headers that apply to the entire message—not to a specific part. This field uses an array to display **Name** and **Value** properties.
Parts	An array of the parts to include with the HTTP request, beginning with the second part. Each part consists of: ➤ **Headers.** An array of headers for the part. This field uses an array to display **Name** and **Value** properties. ➤ **Body.** The file content's checksum value. The checksum is computed by applying the MD5 hash function to the specified file. The **Calculate file checksum** button ... in the **Value** column, allows you to load a file containing the response body.

Java Activities

This activity group contains the Java-related activity.

Call Java Class. Allows you to set up and configure a call to a Java class.

For details about setting up the Java call, see the "Call Java Class Settings Dialog Box" on page 124.

For task details, see "How to Create a Call to a Java Class" on page 91.

JMS Activities

This activity group allows you to handle JMS messages. For details, see the "JMS Transport Overview" on page 31.

- **Publish Message to Topic.** Publishes a message to multiple subscribers using a JMS topic.
- **Receive Message from Queue.** Receives a message from a JMS queue.
- **Receive Message from Topic.** Receives published messages to a specific JMS topic for a subscription. Place this step after **Subscribe to Topic**.
- **Send Message to Queue.** Sends a message to a JMS queue.
- **Send and Receive Message from Queue.** Sends a message to a specified queue and receives a message from a specified queue.
- **Subscribe to Topic.** Creates a subscription for a JMS topic. Use this step before any **Receive Message from Topic** steps for the subscription.

Input Properties - JMS Activities

The following section describes the input properties for the JMS activities.

Publish Message to Topic

Property	Description
Topic name	The JNDI name of the queue from which to receive the message.
Message	The message to publish.
JMS send properties	JMS properties with the key/value form. You can use the key to modify JMS header or message properties.

Receive Message from Queue

Property	Description
Receive queue name	The JNDI name of the queue from which to receive the message. If the value is empty, Service Test uses a temporary queue.
JMS receive message selector	An SQL expression to filter the received message. For more information on the syntax for defining selectors, see the **Message Properties** section in http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/jms/Message.htm. For example, a key JMSCorrelationID, with a value of 4566636, receives only messages whose correlation ID is 4566636.

Receive Message from Topic

Property	Description
Topic name	The JNDI name of the topic from which to receive the message.
Subscription name	The name of the subscription. Use the same subscriber name value from the previously-called **Subscribe to Topic** step.

Send Message to Queue

Property	Description
Send queue name	The JNDI name of the queue from which to receive the message.
Message	The message to send.
JMS send properties	JMS properties with the key/value form. You can use the key to modify JMS header or message properties.

Send and Receive Message From Queue

Property	Description
Send queue name	The JNDI name of the queue to which to send the message.
Receive queue name	The JNDI name of the queue from which to receive the message. If the value is empty, Service Test uses a temporary queue.
Message	The message to send.
JMS send properties	JMS properties with the key/value form. You can use the key to modify JMS header or message properties.
JMS receive message selector	An SQL expression to filter the received message. For more information on the syntax for defining selectors, see the **Message Properties** section in http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/jms/Message.htm. For example, a key JMSCorrelationID, with a value of 4566636, receives only messages whose correlation ID is 4566636. **Tip:** To automatically generate a selector, use the **Automatically generate selector** option in the Property Sheet's **Test Settings** tab. This selector uses a correlation ID that corresponds to the request's message ID.

Subscribe to Topic

Property	Description
Topic name	The JNDI name of the queue from which to receive the message.
Subscription name	The name of the subscription. To retrieve messages on the subscription created by this step, insert a **Receive Message from Topic** step using the same **Subscription name** value.
JMS receive message selector	An SQL expression to filter the received message. For more information on the syntax for defining selectors, see the **Message Properties** section in http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/jms/Message.htm.

Checkpoints - JMS Activities

The following section describes the checkpoints for the JMS activities. Certain checkpoints may not apply to all activities.

Property	Description
JMSMessageID	The ID of the message.
JMSTimeStamp	The time and date of the message.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSReplyTo	The contents of the **Reply To** field.
JMSDestination	The message's destination.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. False by default.
JMSType	A string describing the JMS type.
JMSExpiration	The message's expiration date.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMS properties	An array of message properties in the structure of keys and values. **Tip:** Use the **Number of elements** drop -down to specify the number of properties, or click the **Add** button to add elements.
Message body	The message content.

Load Testing Activities

This activity group contains activities that allow you prepare the test for load testing:

- **Start Transaction.** Marks the beginning of a LoadRunner transaction.
- **End Transaction.** Marks the end of a LoadRunner transaction.
- **Think Time**. Emulates a time delay, in seconds, between steps.

For details, see "Prepare for load testing - optional" on page 248.

Unified Functional Testing Activities

This activity group contains activities that allow you implement unified testing plan. For task details, see "How to Implement UFT Testing" on page 88.

- **Call Service Test test.** Invokes a Service Test test when the current Test Flow or loop reaches this step.
- **Call QuickTest Professional test.** Invokes a QuickTest Professional test when the current Test Flow or loop reaches this step. Do not insert a call to a QuickTest test that contains a call to a Service Test test, as this can cause unexpected behavior.
- **Call VuGen script.** Runs a VuGen (LoadRunner's Virtual User Generator) script.

General Properties - Unified Functional Testing

The following section describes the general properties for the UFT activities.

Script Path. The path of the Service Test or QuickTest test, or VuGen (LoadRunner's Virtual User Generator) script to run.

XML Activities

This activity group contains activities related to XML files.:

- **Transform XML.** Converts an XML file to another structure based on the specified XLST.
- **Validate XML.** Validates an XML file against an XSD schema.

Web Service Activities

This activity group contains SOAP requests and calls to Web services.

- **SOAP Request.** Sends a manual SOAP request to the server, usually in XML format. You can import a schema and load XML for this activity. For details, see "Input/Checkpoints View" on page 55.
- **<Imported Web service methods>**. For imported Web services, a tree hierarchy showing:
  - **Level 1.** Service name
  - **Level 2.** Port number
  - **Level 3.** Operation name

Input Properties - SOAP Request

To view the General properties, click the **General view** button in the Property Sheet. Service Test derives these properties from the WSDL file.

Property	Description
Standard Schema	A schema for the SOAP request.
	Imports a schema XSD file.
	Loads XML with values for the schema.
	Deletes the current schema.

General Properties - Web Services

To view the General properties, click the **General view** button in the Property Sheet. Service Test derives these properties from the WSDL file.

Property	Description
Endpoint Address	The endpoint location of the service as derived from the WSDL.
SoapAction	An expression indicating the SOAP action, for example, HP.SOAQ.SampleApp/IHPFlights_Service/DeleteFlightOrder.
ContentType	The type of content: For example text/xml; charset=utf-8.
IsOneWay	Indicates whether the service is a one way service: true or false.

Activity-Specific Dialog Boxes

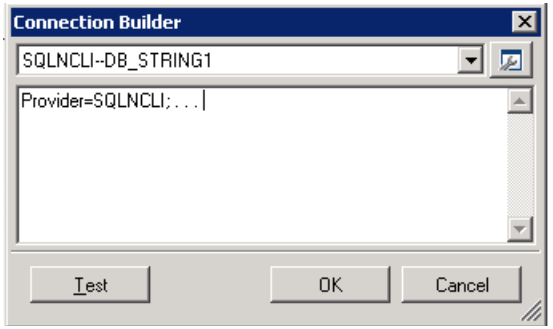
Several activities provide separate dialog boxes to assist you in creating and assigning Input property data.

This section includes:

- "Connection Builder Dialog Box" on page 117
- "Query Builder Dialog Box" on page 118
- "Query Designer Dialog Box" on page 120
- "Call Java Class Settings Dialog Box" on page 124

Connection Builder Dialog Box

This dialog box helps you create a database connection expression for the **Database** > **Open Connection** activity.



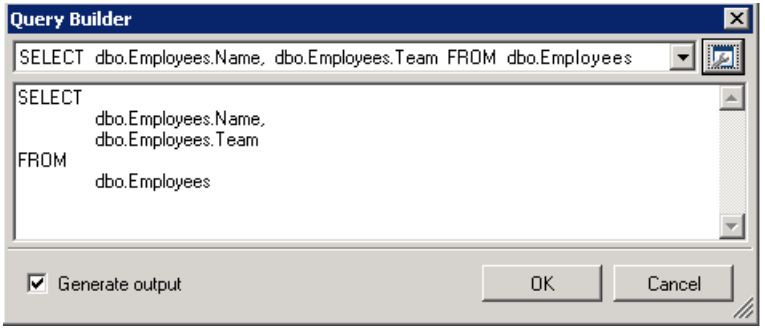
To access	Do the following: **1** Drag a **Database** > **Open Connection** activity into the canvas. **2** Open the **Input/Checkpoints** view in the Property Sheet. **3** Select the Input property—**Connection string**. **4** Click the **Connection Builder** button ... in the right side of the **Value** column.
Relevant tasks	"How to Execute Database Commands or Retrieve Data" on page 81

User interface elements are described below:

UI Elements	Description
Connection string	A drop-down list of previously defined connection strings.
	Connection Builder. Opens the Microsoft Data Link Properties dialog box to help you create a new string or edit an existing one. **Tip:** For details, click **Help** in the Data Link Properties dialog box.
<connection string area>	An editable area for pasting in existing strings, or for editing the connection string selected in the drop-down list.
	Checks the database connection using the string shown in the text area. If it succeeds to connect using the string, Service Test issues a message indicating that the connection opened successfully. After you test a connection, it remains open until after the test is complete or when you close it manually using a **Close Connection** step.

Query Builder Dialog Box

This dialog box helps you create an SQL statement for the **Query string** property of the **Database** > **Select Data** activity.



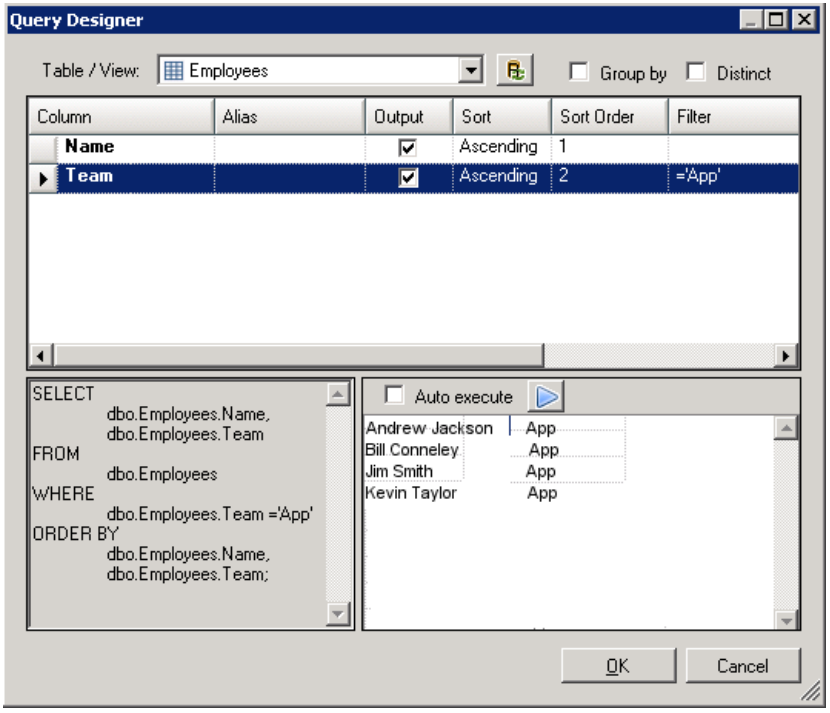
To access	Do the following: **1** Drag a **Database** > **Select Data** activity into the canvas. **2** Open the **Input/Checkpoints** view in the Property Sheet. **3** Select the Input property—**Query string**. **4** Click the **Query Builder** button in the right side of the **Value** column.
Relevant tasks	"How to Execute Database Commands or Retrieve Data" on page 81

User interface elements are described below:

UI Elements	Description
Query string	A drop-down list of previously defined query strings.
	Query Designer. Opens the Query Designer dialog box.
<query string area>	An editable area for pasting in existing strings, or for editing the query string selected in the drop-down list.
Generate output	Automatically retrieves table information from the database when you click **OK,** and sets the structure of the output in the Property Sheet's **Checkpoints** area.

Query Designer Dialog Box

This dialog box helps you design an SQL statement for the **Query string** property of the **Database** > **Select Data** activity.



To access	➤ In the **Input/Checkpoints** view, when setting the Query string property for the **Select Data** step: **1** Open the Query Builder Dialog Box. **2** Click the **Query Designer** button . ➤ In the **Data Window**, when defining a new Database type data source using the "Add New Database Data Source Wizard" on page 313: **1** In the Add New Database Data Source wizard, proceed to the second screen, **Define the query statement**. **2** Click the **Query Designer** button to the right of the **SQL statement** box.
Relevant tasks	"How to Execute Database Commands or Retrieve Data" on page 81 "Add a Database data source" on page 279

User interface elements are described below:

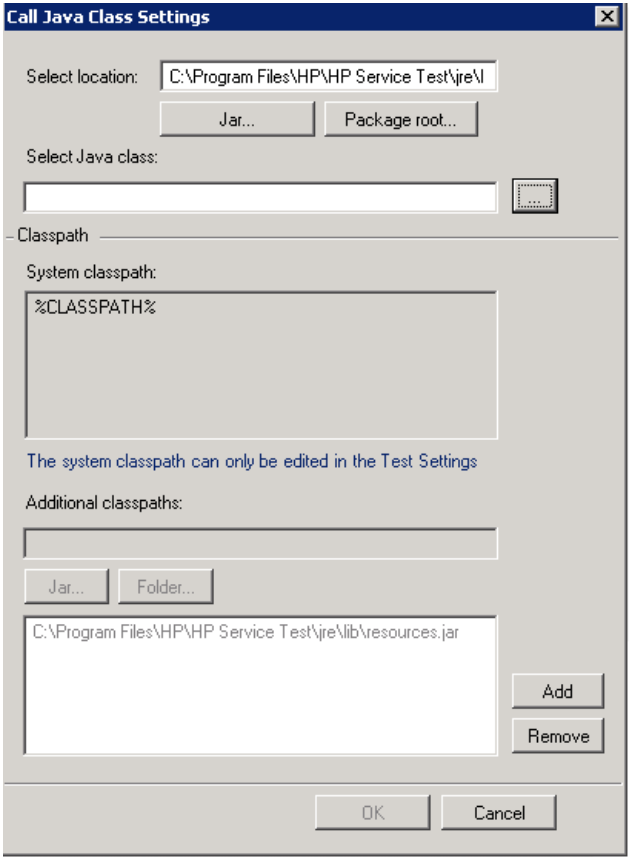
UI Elements	Description
Table/View	A list of the tables and views in the database. The tables and views are distinguishable by different icons.
	Reset Query. Resets the query to its original state and discards all changes.

UI Elements	Description
<table/view columns grid>	A list of the columns in the selected table or view. The grid contains the following columns: ➤ **Column.** The name of the column in the table or view. ➤ **Alias.** An alias name for the column, if applicable. ➤ **Output**. When enabled, it includes the column in the SELECT statement. ➤ **Sort.** Allows you to sort the results in ascending or descending order. ➤ **Sort Order.** When working with multiple sorted columns, this indicates the sorting order beginning with 1. ➤ **Filter.** Allows you to filter the results of the query, by adding a WHERE <column_name> = <filter text> to the SQL statement.
Group by	Adds a **GroupBy** column to the grid, allowing you to select a grouping method such as: Sum, Avg, Min, Max, Count, and so forth. It adds GROUP BY to the SQL statement. If you select a **GroupBy** criteria, it also adds it to the SQL expression.
Distinct	Adds a DISTINCT term to the SQL statement, instructing the query to retrieve only distinct record sets.
<SQL statement preview pane>	A preview of the SQL statement generated by your selections.
<returned rows pane>	The rows returned by the query. **Tip:** If you enable **Autoexecute**, these rows display the current query results.

UI Elements	Description
Auto execute	Automatically executes the SQL statement in the **Preview pane** whenever you make a change in the upper panes of the dialog box. **Tip:** If you notice that the query takes a long time to execute, you can disable this option and click **Execute query** to manually run it.
	Execute query. Executes the query shown in the **Preview pane**. Use this when **Auto execute** is disabled. **Tip:** If the **Returned rows** pane is grayed out, this indicates that the results are not current.

Call Java Class Settings Dialog Box

This dialog box enables you to configure the settings for a call to a Java class.



To access	Do the following: **1** Add a **Call Java Class** step to the canvas. **2** In the Property Sheet, open the Java Call Input Output Input/Checkpoints view . **3** Click the **Select Java File** button.
Relevant tasks	"How to Create a Call to a Java Class" on page 91

User interface elements are described below:

UI Elements	Description
Select location	The location of the Java class. ➤ Jar... Opens the **Select Jar File** dialog box. ➤ Package root... Opens the **Browse for Folder** dialog box to select a root folder.
Select Java class	The Java class to associate with the **Call Java Class** step. When you click **Browse**, Service Test shows the classes in the specified **jar** file.
System classpath	The paths you defined in the Test Settings for system classpaths. To modify this setting, click in the canvas (outside the Test Flow or Loop and open the Property Sheet's **General** view. Modify the Test Setting's **JVM** > **Classpath** node.
Additional classpaths box	Additional classpaths to associate with the **Call Java Class** step. ➤ Jar... Opens the **Select Jar File** dialog box. ➤ Folder... Opens the **Browse for Folder** dialog box to select a folder containing the **.class** file.
<additional classpath list>	A list of **jar** files or folders containing the additional classes. ➤ Add Adds the contents of the **Additional Classpath** box to the additional classpath list. ➤ Remove Removes the selected entry from the list.

Troubleshooting and Limitations - Activities

This section describes troubleshooting and limitations for working with activities.

- **Call Java Class.** Supports only Java primitive types.
- **HTTP.** This activity does not support client certificates.
- **SOAP Request.** Switching between **Text** and **Grid** views, may cause the grid to display an element added below the Body's **Any** node, under the Header's **Any** node.
  Workaround: The Text view shows the correct XML.
- **Call Service Test test.** JMS transport is not supported for tests containing the Call Service Test test activity.
- **Database Transaction activities:** Service Test does not support nested transactions.
  Workaround: Add a new loop activity within an existing transaction. Add the new transactions steps to the newly created loop. Make sure to set the loop iteration to 1.

3

Creating and Saving Tests

This chapter includes:

Concepts

- Test Overview on page 128
- Business Process Testing on page 128

Tasks

- How to Create a New Test on page 131
- How to Add a Test to a Solution on page 132
- How to Create a New Business Component on page 134
- How to Define Test Properties or User/System Variables on page 136

Reference

- Test User Interface on page 139

Troubleshooting and Limitations - Upgrading Tests on page 144

Concepts

Test Overview

To test your applications, you use Service Test to create tests or business components.

You can base new tests on the standard Service Test template, or, if you have LoadRunner installed, on a Load Testing template.

You can create tests on either the file system, or within HP ALM/QC (Application Lifecycle Management/Quality Center). You create business components in the ALM/QC repository.

Business Process Testing

Business Process Testing is a methodology in which several test components are combined to create a complete business process. The BPT user composes a business process by combining a series of test components with data flow between them.

Test components are usually comprised of several steps or service operations. For example, a component's first step may be to read the contents of a file. Its second step could be searching for a string and replacing it. Its third step could be reporting the output to a report.

You can create business process test components from within HP ALM/QC or through Service Test.

In HP ALM/QC, using the BPT model, non-technical SMEs (Subject Matter Experts) can define design steps that are required for the component, using simple textual descriptions of the step's function.

After these are defined, the technical engineer creates a component through a testing application such as Service Test, that performs the desired actions. For further information about creating Business Components from within HP ALM/QC, see the *HP Business Process Testing User Guide*.

A Subject Matter Expert using Business Process Testing in HP ALM/QC combines your saved business components into one or more business process tests. These tests are used to check that the application behaves as expected.

This section also includes:

- "Benefits of Business Process Testing" on page 129
- "Business Process Testing in Service Test" on page 130

Benefits of Business Process Testing

Some of the advantages of working with a BPT module over individual tests are:

- Enables less-technical subject matter experts to create tests
- Enables structured automated testing
- Reduces the duplication of effort when combining manual tests with automatic tests
- Allows component reusability to speed-up the automation process
- Provides the ability to pass parameters from one step to another within your business process. You can save the output of a step to a parameter and use it as an input value for subsequent steps
- Simplifies on-going test maintenance
- Minimizes time-to-test

For more information about creating a BPT components in HP ALM/QC, see the *HP Business Process Testing User Guide*.

Business Process Testing in Service Test

You can create a business component within Service Test or from within ALM/QC. You can also save an existing test as a business component using the **File** menu's **Save as Business Component** option.

When you create a business component within Service Test, the following limitations apply:

- Load Test activities are not supported.
- UFT (Unified Functional Testing) activities are not supported.
- Business components cannot be saved on the file system—only on ALM/QC.
- You cannot save a load-enabled test as a business component.Create a new business component from the **File** menu, or save a non-load-enabled test as a business component.
- Encoded password type properties are not supported. If the component has a property of type password (or encrypted in ALM 11.00 and later), the value will be treated as an ordinary string, without encoding or decoding.
- If you save a Load-Test enabled test as a business component, it will no longer be Load-Test enabled.
- Multiple User Variable profiles are not supported. Remove all but one of the profiles. For details, see "Action Buttons - Test Variables View" on page 62.

You cannot save an existing test containing these activities as a business component. You must first remove these test steps before saving the test as a business component.

For details about creating a business component in Service Test, see "How to Create a New Business Component" on page 134.

Tasks

How to Create a New Test

The following steps describe how to create a new test.

- "Set the path type - optional" on page 131
- "Connect to ALM/QC- optional" on page 131
- "Open the Create a New Test dialog box" on page 131
- "Select a template" on page 132
- "Specify a name" on page 132
- "Select a location" on page 132
- "Generate the test" on page 132
- "Add steps to the test" on page 132

1 Set the path type - optional

By default, the path used for the test is relative. If you plan to transfer the test to other machines, you can use an absolute path. This setting is only relevant for UFT (Unified Functional Testing) steps, such as to a **Call to QuickTest Test** step. Select **Edit** > **Settings** and clear the **Use a Relative Path for UFT Steps** option.

2 Connect to ALM/QC- optional

To create tests in an ALM/QC repository, first connect to the ALM/QC server. For details, see "How to Connect to ALM/QC" on page 370.

3 Open the Create a New Test dialog box

Select **File** > **New** > **Test**. For user interface details, see the "Create New Test Dialog Box" on page 140.

4 Select a template

In the Create a New Test dialog box, select the **Service Test** template. For load testing, select the **Service Test Enabled for Load Testing** template. The latter template is available only when LoadRunner is installed on the machine.

5 Specify a name

Accept the default name or specify your own string in the **Name** box.

6 Select a location

Accept the default location, or click the **Browse** button adjacent to the **Location** box to open the Open Folder dialog box.

For a location on the file system, browse to the folder and click **Open**.

For ALM/QC tests, click the **ALM/QC Test Plan** button in the left pane, and navigate to the folder. Use the New Folder button to create a new branch. Click **Open**.

7 Generate the test

In the Create a New Test dialog box, click **Create**.

8 Add steps to the test

Drag activities from the Toolbox into the canvas. For details, see Chapter 2, "Service Test Activities."

How to Add a Test to a Solution

The following steps describe how to add a new test or an existing one to your solution. Each test has its own canvas upon which you can add a different set of activities.

➤ "Add a new test to the current solution" on page 133

➤ "Add an existing test to the current solution" on page 133

Add a new test to the current solution

1 Select **File** > **Add** > **New Test**. For user interface details, see the "Add New Test Dialog Box" on page 142.

2 In the Add New Test dialog box, select the **Service Test** template. For load testing, select the **Service Test Enabled for Load Testing** template. The latter template is only available when LoadRunner is installed on the machine.

3 Specify a name. Accept the default name or specify your own string in the **Name** box.

4 Select a location. Accept the default location, or click the **Browse** button adjacent to the **Location** box to open the Open Folder dialog box.

- For a location on the file system, browse to the folder and click **Open**.
- For HP ALM/QC tests, click the **ALM/QC Test Plan** button in the left pane, and navigate to the folder. Use the New Folder button to create a new branch. Click **Open**.

5 Click **Create** to generate the test.

Add an existing test to the current solution

1 Select **File** > **Add** > **Existing Test**.

2 In the **Open** dialog box, browse to a **Service Test** test on the file system or in ALM/QC.

3 Click **Open** to add the existing test.

Note: The ability to add multiple tests is limited to solutions stored on the file system—not those stored on QC/ALM.

How to Create a New Business Component

The following steps describe how to create a new business component from within Service Test. Each template provides an empty canvas to which you can add activities to create the business component. For more information, see "Business Process Testing" on page 128.

If you have Service Test installed on your ALM/QC machine, you can launch Service Test directly from the **Test Plan** or **Business Component** module **Automation** tab. For details, see the *HP ALM/QQC User Guide*.

- "Prerequisite - Connect to ALM/QC" on page 134
- "Open the Create a New Test dialog box" on page 134
- "Select a template" on page 134
- "Specify a name" on page 135
- "Select a location" on page 135
- "Generate the business component" on page 135
- "Add component steps" on page 135

1 Prerequisite - Connect to ALM/QC

To create a business component, you must first connect to a project in an ALM/QC with a valid Business Process Testing license. Select **File > ALM/QC Connection**. For details, see "How to Connect to ALM/QC" on page 370.

2 Open the Create a New Test dialog box

Select **File > New > Business Component**. For user interface details, see the "Create New Test Dialog Box" on page 140.

3 Select a template

In the Create New Test dialog box, select the **Components** node in the **Categories** pane. In the **Templates** pane, select the **Business Component** template.

4 Specify a name

Accept the default name or specify your own string in the **Name** box.

5 Select a location

Click the **Browse** button adjacent to the **Location** box and locate a folder for the component in ALM/QC.

6 Generate the business component

In the Create New Test dialog box, click **Create**.

7 Add component steps

Drag the activities into the canvas to create steps in the business component.

Tip: To convert an existing test to a business component, select **File > Save as Business Component** and save the script to the desired location in ALM/QC. If your test contains UFT or Load Testing steps, you cannot convert it until you remove these steps.

How to Define Test Properties or User/System Variables

The following steps describe how to define test properties, user variables, and operating system variables. These settings are optional.

- "Define test properties" on page 136
- "Define user variables" on page 136
- "Define user variable profiles" on page 137
- "Set user variable values" on page 137
- "Set operating system variables" on page 138

Define test properties

This step applies if you want to define custom properties that can be used by all steps in the test, with the ability to assign data from a data source.

1 Click in a blank area of the canvas. In the Property Sheet, open the **Test Input Parameters** view.

2 Click the **Add Property** button to define a new input or output property.

3 In the **Add Input/Output Property** dialog box, specify a property name and type.

4 Click in the **Default Value** column. Specify a value or click the **Link to Source** button to specify a data source. For details, see "Select Link Source Dialog Box" on page 305.

Define user variables

Service Test lets you create user variables and set their values. You can define multiple profiles for the variable values. You select a profile to be active before the test run. For details, see "Define user variable profiles" on page 137.

1 Click in a blank area of the canvas. In the Property Sheet, open the **Test Variables** view.

2 Click the **Add User Variable** button to define a new user variable.

Set user variable values

You can set values for variables in one of the following ways:

- In the Property Sheet's **Test Variables** view, click in the **Profile** column and manually enter values.
- Open the Toolbox (left pane) and drag the **Set Test Variable** activity from the **Miscellaneous** category into the **Test Flow** (or loop). In the **Property Sheet**, define the variable key and value. To obtain values, click the Link to a Data Source button in the row. In the Select Link Source dialog box, select the **Test Variables** option. Select a name and value for the **Variable key** and **Variable value**.
- Open the Property Sheet's **Events** view for the step or define a Custom Code activity. Edit the event handler code and assign a value using the **TestProfile** object. The following example sets the value of the Region user variable to NE.

```
activity.Context.TestProfile.SetVariableValue("Region", "NE");
```

  For details, see Appendix A, "Coding Service Test Events."

Repeat this step for each variable for which you want to set values.

Define user variable profiles

This step applies only if you created user variables as described in the above steps.

1. Define one or more user variables as described above.
2. Click the **Add New Profile** button.
3. In the New Test Profile dialog box, specify a name and indicate the profile (if any) from which to copy the properties. If you do not copy from an existing profile, Service Test copies the user variables from the active profile without its values.
4. To rename or delete a profile, click the **Manage Profiles** button. Click **Remove** or **Rename**.
5. Click the **Compare** button to display the profiles side-by-side.

6 Only the active profile is accessed during the test run. To make a profile active, select it from the **Active Profile** list.

For user interface information, see the "Test Variables View" on page 62.

Set operating system variables

1 Open the Toolbox and drag the **Set OS Environment Variable** activity from the **System** category into the **Test Flow** or another custom loop.

2 In the Property Sheet's **Input/Checkpoints** view, define the variable key and value or click the **Link to Source** button to specify a data source.

Tip: To view a list of the test variables, click in a blank area of the canvas. In the Property Sheet, open the **Test Variables** view. The System variables are listed in the lower pane.

Reference

Test User Interface

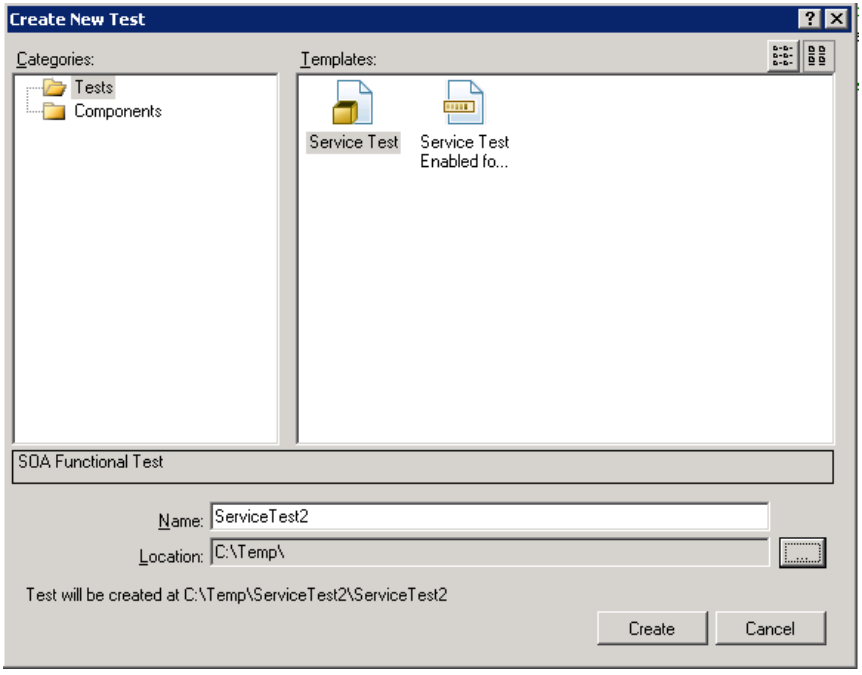
This section includes:

➤ Create New Test Dialog Box on page 140

➤ Add New Test Dialog Box on page 142

Create New Test Dialog Box

This dialog box enables you to specify the location and name of the new test.



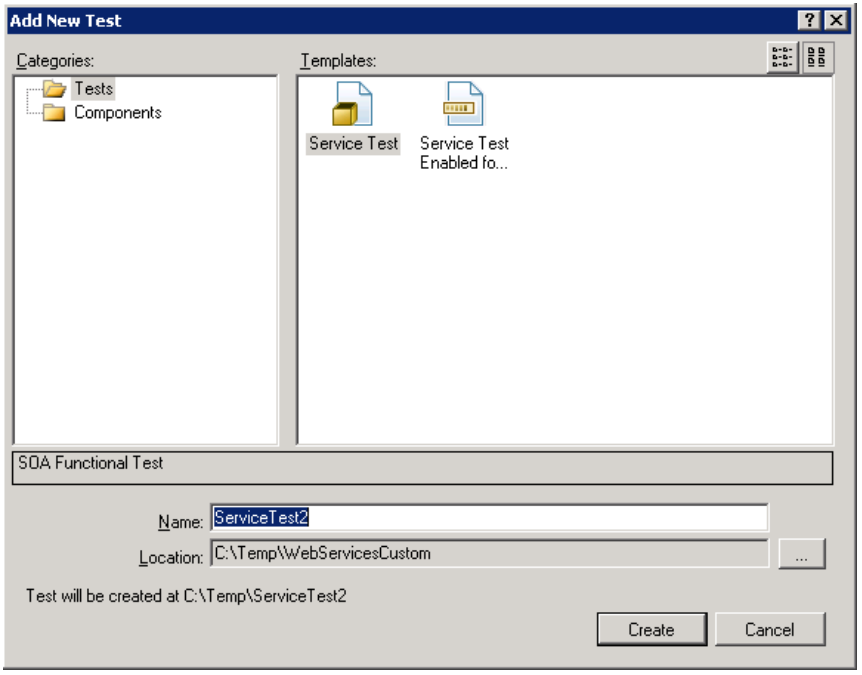
To access	Select **File** > **New** > **Test** (CTRL + SHIFT +N).
Relevant tasks	"How to Create a New Test" on page 131 "How to Create a New Business Component" on page 134

User interface elements are described below:

UI Elements	Description
	Browse. Allows you to select a location for the new test. To select a location on an ALM/QC server, first connect to the server. For details, see "How to Connect to ALM/QC" on page 370.
	Creates a new test or business component in the selected location.
Categories pane	A list of the test categories: ➤ **Tests**: For regular and Load Testing type tests. ➤ **Components**: For Business Process Testing.
Location	The complete path of the test. Use the **Browse** button to select a location. For business components, connect to an ALM/QC project.
Name	The name of the test or business component. It can contain only letters, numbers, underscores, spaces, and dots (".").
Templates pane	A collection of templates: ➤ Tests: **Service Test** and **Service Test Enabled for Load Testing**. ➤ Components: **Business Component** **Note:** The **Service Test Enabled for Load Testing** template is visible only when HP LoadRunner is installed.

Add New Test Dialog Box

This dialog box enables you to specify the location and name of the new test.



To access	Select **File** > **Add** > **New Test.**
Relevant tasks	"How to Add a Test to a Solution" on page 132

User interface elements are described below:

UI Elements	Description
	Browse. Selects a location for the new test. To select a location on an ALM/QC server, first connect to the server. For details, see "How to Connect to ALM/QC" on page 370.
	Adds a new test or business component to the current solution.
Categories pane	A list of the test categories: ➤ **Tests**: For regular and Load Testing type tests. ➤ **Components**: For business process components.
Location	The complete path of the test or the location if connected to ALM/QC. Use the **Browse** button to choose a location. For business components, connect to an ALM/QC project.
Name	A name for the test or business component. May contain only letters, numbers, underscores, spaces, and dots ("..").
Templates pane	A collection of templates: ➤ **Tests**: Service Test and Service Test Enabled for Load Testing. ➤ **Components**: Business Component **Note:** The Service Test Enabled for Load Testing template is only visible when HP LoadRunner is installed.

Troubleshooting and Limitations - Upgrading Tests

This section describes limitations for opening tests created with earlier versions of Service Test.

- REST methods created in Service Test version 11.00 are not supported by Service Test 11.10. This applies when upgrading a test that links a property from the inner **HTTP** step to an outer output property. Service Test resets the operator to '=' and takes the value from the source instead of the original value.
  Workaround: Redefine the REST methods.
- When Service Test upgrades a test from version 11.00 to 11.10, it does not retain the Security settings.
  Workaround: In Service Test 11.00, save the Security Scenario to an **.stss** file. Import this file for your service in the upgraded test. For details, see "Setting Security Overview" on page 182.
- In certain instances, if Service Test was unable to upgrade the test, you may need to modify the code to make it compatible with version 11.10:
  - The user code file is now titled TestUserCode.cs.
  - The namespace at the beginning of the test is Script.
  - The class definition is public class TestUserCode : TestEntities.

```
namespace Script
{
	using System;
	using System.Xml;
	using System.Xml.Schema;
	using HP.ST.Ext.BasicActivities;
	using HP.ST.Fwk.RunTimeFWK;
	using HP.ST.Fwk.RunTimeFWK.ActivityFWK;
	using HP.ST.Fwk.RunTimeFWK.Utilities;
	using HP.ST.Fwk.RunTimeFWK.CompositeActivities;
	using System.Windows.Forms;
	using HP.ST.Ext.FTPActivities;


	[Serializable()]
	public class TestUserCode : TestEntities
	…
```

- In upgraded tests that use non-default checkpoint comparison operators, (operators other then '='), the checkpoint is not generated properly.
  Workaround: Modify the test in some way and save it again.
- The **args** variable is not supported in version 11.10.
  Workaround: Change the **args** variable name to the concrete Activity name as it appears at the top of the Property Sheet. For example, change **args**.Output.outStr = **args**.Input.inStr; to **CodeActivity8**.Output.outStr=**CodeActivity8**.Input.inStr;.

4

Defining Services

This chapter includes:

Concepts

- Preparing Service References on page 148
- Updating Services on page 149
- Negative Testing on page 151

Tasks

- How to Import a WSDL-Based Web Service on page 152
- How to Update a Web Service on page 155
- How to Create a Simple Test for a REST Service on page 158
- How to Create a Template-Based REST Service on page 160

Reference

- Import Service User Interface on page 164
- Update Service Interface on page 169

Troubleshooting and Limitations - Services on page 180

Concepts

Preparing Service References

Service Test lets you import or define services for the purpose of creating high-level tests.

For Web Services, you typically begin creating a test by importing a WSDL file. This file provides a structure for the test, since it describes the service in terms of its elements, argument values and properties. For information on importing a WSDL, see "How to Import a WSDL-Based Web Service" on page 152.

If your service changes during development, Service Test provides a mechanism for updating the service. For details, "Updating Services" on page 149.

Service Test allows you to import both Document/Literal and RPC type Web services.

In **Document/Literal** style Web services, the client sends standard XML documents to the server. The server application is responsible for mapping the server objects (properties, method calls, and so forth) and the values in XML documents. The data is serialized according to a given schema, so it can be validated against the schema. For Document/Literal style Web services, Service Test is able to display the input and output properties in a grid, allowing you to assign values for each property independently.

In **RPC** type Web services, the WSDL file and SOAP body contain the complete operation name, its input and output properties, and their values. There is no schema for this type of service and it is not supported by the WS-I conformance standard. The Service Test interface does not display the input and output properties.

If your service document is unique and cannot be imported in the normal way, you can use the SOAP Request activity to send a manual SOAP request to the server.

For REST services, you define the metadata manually by modelling the service and defining your own methods. For task details, see "How to Create a Template-Based REST Service" on page 160.

Service Test lets you validate imported services by checking their compliance with the WS-I standard. For details, see "Validate the WSDL file - optional" on page 154.

Updating Services

You can update Web services that exist in the toolbox, from their original locations or from alternate locations.

When you update a service, first Service Test compares the service and port names. If the port names match, Service Test transfers all of the data from the updated service, including new security information. If the new version of the service contains a port that was not present in the original service, it adds it to the toolbox, as an additional port for the service.

If the port paths (<service name:port name> combination) do not match, Service Test opens the **Update Service and Port Wizard**. For details, see the "Update Service and Port Wizard" on page 170.

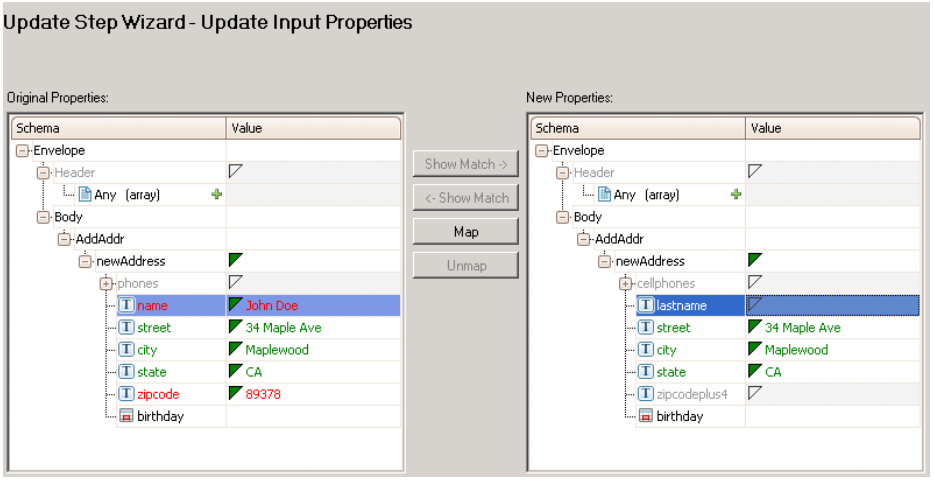
If HTTP transport information such as the endpoint was changed, Service Test will update this automatically, as long as the port path is the same.

In the next step of the update, Service Test compares the operations and properties of the current test steps. Service Test tries to match up the properties that were assigned values between the original service and the updated version.

If the operation names do not match, the Update Step wizard allows you to map the new operation name to the original name.

If the operation names match, but the property names do not, the Update Step wizard allows you to map the new property names to the original ones. Matching properties are ones that have the same XPaths and types.

The wizard screen marks resolved properties in green and unresolved properties in red.



If you originally imported a service from ALM/QC through Service Test Management, you must update the service in ALM/QC—you will not be able to update the service through Service Test.

When you update an RPC Web service, as long as the operation names match, the properties are marked as resolved. If the operation names cannot be resolved, Service Test opens the wizard. If there is no operation available to map to the original, the wizard suggests that you delete the affected step.

For task details, see "How to Update a Web Service" on page 155.

Negative Testing

When performing a functional test for your Web Service, you should approach the testing in a variety of ways. The most common type of testing is called **Positive Testing**—checking that the service does what it was designed to do.

In addition, you should perform **Negative Testing**, to confirm that the application did not perform a task that it was not designed to perform. In those cases, you need to verify that the application issued an appropriate error—a SOAP Fault.

To illustrate this, consider a form accepting input data—you apply positive testing to check that your Web Service has properly accepted the name and other input data. You apply negative testing to make sure that the application detects an invalid character, for example a letter character in a telephone number.

When your service sends requests to the server, the server responds in one of the following ways:

- **SOAP Result.** A SOAP response to the request.
- **SOAP Fault.** A response indicating that the SOAP request was invalid. Negative Testing applies only to SOAP faults.
- **HTTP Error.** An HTTP error, such as Page Not Found, unrelated to Web Services.

Service Test can check for a standard SOAP result or a SOAP fault response. For example, if your Web Service attempts to access a Web page that cannot be found, it will issue a 404 HTTP error. Using negative testing you indicate that you expect a SOAP fault. In this case, the test run will fail if the service accesses a *valid* Web page.

Service Test's Property Sheet lets you provide values for the SOAP fault header and body. You can enter **faultcode**, **faultstring**, and **faultactor** values as well as custom properties using **Any** type parameters. Using the Checkpoint mechanism, you can validate these values and view the results in the Run Results Viewer.

For details on setting up the SOAP fault values, see "SOAP Fault View" on page 61.

Tasks

How to Import a WSDL-Based Web Service

This task describes how to import a Web Service into your project.

This task includes the following steps:

- "Prerequisites" on page 152
- "Open the Import/ Select Service dialog box" on page 152
- "Select a source" on page 153
- "Set connection settings for URL/UDDI imports - optional" on page 153
- "Complete the import" on page 154
- "Validate the WSDL file - optional" on page 154

1 Prerequisites

If you intend to import a WSDL through Service Test Management, the ALM/QC extension for application components, open a connection to the ALM/QC server. For details, see the "HP ALM/QC Connection Dialog Box" on page 373.

2 Open the Import/ Select Service dialog box

- To import a WSDL from the file system or ALM/QC, click Import Service > **Import Service from File or ALM/QC Application Component**. For details, see the "Select Service Dialog Box" on page 165.
- To import a WSDL from a Web site or UDDI repository, click Import Service > **Import Service from URL or UDDI**.
  - For a URL, select **Import from: URL**
  - For a UDDI, select **Import from: UDDI**

For details, see the "Import Service from URL or UDDI Dialog Box" on page 167.

3 Select a source

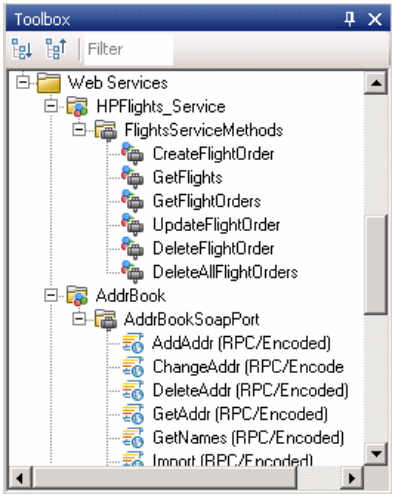
- For **File System** or **ALM/QC Application Components** imports, click the appropriate button in the left pane. Navigate to the WSDL in the file system or the ALM/QC repository.
- For a **URL** import, click the **Browse** button adjacent to the **Address** field. This opens a browser window. Navigate to the WSDL and close the browser. Service Test automatically places the URL in the **Address** field.
- For a **UDDI**, click the **Browse** button adjacent to the **Address** field. This opens the Select Service from UDDI dialog box. Specify a UDDI address and select the service to import. For details, see the "Select Service from UDDI Dialog Box" on page 168.

4 Set connection settings for URL/UDDI imports - optional

For URL or UDDI imports, if your WSDL must be accessed through a secure server or proxy machine, click **Connection Settings** to expand the dialog box. Enter the authentication information or the proxy server details.

5 Complete the import

Click **OK**. If Service Test succeeds in importing the WSDL, it displays the service, its ports, and operations in the Toolbox under the **Web Services** node. In the Toolbox, Service Test differentiates between a Document/Literal and RPC style services with different icons, and an RPC label.



6 Validate the WSDL file - optional

To check the WSDL's compliance with the WS-I standard, select the service and choose **Validate WS-I Compliance** from the shortcut menu. The Output window shows the WS-I validation results. These validations only apply to Document/Literal style services—not RPC.

How to Update a Web Service

This task describes how to update a WSDL-Based Web Service that was already imported and incorporated into a test.

This task includes the following steps:

- "Prerequisites" on page 155
- "Update the service" on page 155
- "Accept the warning" on page 156
- "Run the Update Service and Port wizard" on page 156
- "Run the Update Step wizard" on page 156

1 Prerequisites

If you want to update the WSDL from ALM/QC, make sure you have an open connection to the ALM/QC server. For details, see the "HP ALM/QC Connection Dialog Box" on page 373.

2 Update the service

- To update a service from its original location, select its main node in the **Toolbox**. Choose **Update Service** from the shortcut menu.
- To update the service from a different location, or if the **Update Service** option is not available:
  a. Select the service's main node in the **Toolbox**.
  b. Choose **Update from** > **URL or UDDI** or **Update from** > **File or ALM/QC Application Component** from the shortcut menu.
  c. Navigate to the WSDL and click **OK**.

Note: The **Update Service** option may not be available if the user credentials changed or if the service was imported with an earlier version of Service Test.

---

3 Accept the warning

Service Test issues a warning message that the service was updated. Click **Yes** to accept the update.

4 Run the Update Service and Port wizard

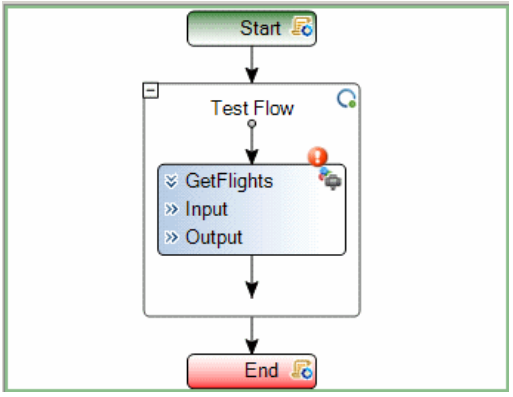
If Service Test detects a change in the port path (<service name:port name> combination) the **Update Service and Port** Wizard opens. Follow the steps of the wizard to resolve all of the Service/Port conflicts. For details, see "Update Service and Port Wizard" on page 170.

Note: The wizard imports all of the services defined in the WSDL, even those deleted manually before the update. To remove them from the toolbox, delete them again manually, after the update.

---

5 Run the Update Step wizard

If a test step became invalid because an operation name changed or properties with values were changed, then the canvas marks the step with a warning marker.

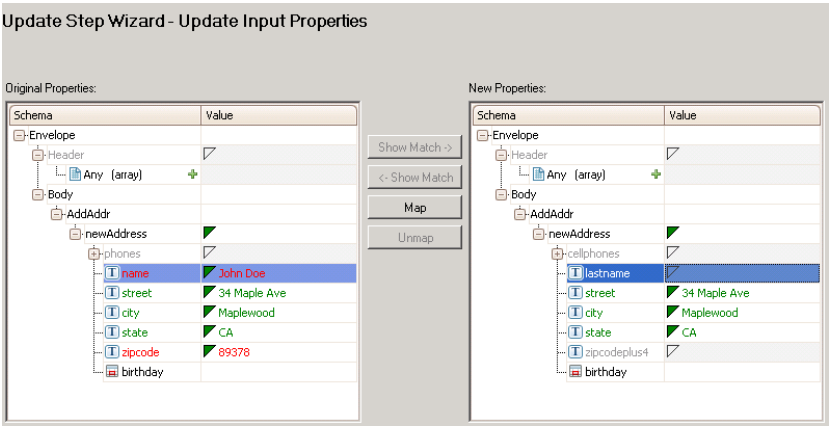


a Click the warning marker to display the message **The step must be resolved. Resolve step…** Click on the message text. The **Update Step** Wizard opens.

Note that the step's properties become read-only, until you resolve them with the help of the wizard.

b In the **Select Operation** page, click in the **New operation** pane and select the operation that corresponds to the original one in the upper pane. Click **Next**. Properties that had been assigned values in the Property Sheet and were affected by the upgrade, are highlighted in red.

c In the **Update Input Properties** page, in the **Original Properties** pane, select a property for which there is a conflict, highlighted in red. In the right pane, **New Properties**, select a property to to map.



Click **Map**. The wizard adds the mapping to the list in the bottom pane. Repeat this for all properties that you want to map. Click **Next**.

d In the **Update Output Properties** page, select a property for which there is a conflict, highlighted in red. In the right pane, **New Properties**, select the property to which you want to map. Click **Map**. The wizard adds the mapping to the list of mappings in the bottom pane. Repeat this for all properties that you want to map. Click **Next**.

e Click **Finish** to save your changes and close the wizard.

For details about the wizard, see the "Update Step Wizard" on page 173.

How to Create a Simple Test for a REST Service

This task describes how to model a REST service and create a simple test. Optionally, you can create a template for each method, allowing you to reuse it within your test without having to provide the configuration settings each time. For details, see "How to Create a Template-Based REST Service" on page 160.

This task includes the following steps:

- "Add a new REST service" on page 159
- "Add a resource" on page 159
- "Add a method" on page 159
- "Create a test step" on page 159
- "Locate the endpoint URL" on page 159
- "Enter the input values" on page 159
- "Locate the request body" on page 160
- "Run the test and view the report" on page 160

1 Prerequisite

Study the structure of the REST Service body and determine which resources and methods you need to define.

2 Add a new REST service

In the Toolbox palette select the **REST Services** node. Choose **Add Service** from the shortcut menu. Provide a meaningful name for the service.

3 Add a resource

Select the service node and choose **Add Resource** from the shortcut menu. Provide a meaningful name for the resource. Repeat as required.

4 Add a method

Select the service or resource node, and choose **Add Method** from the shortcut menu. Provide a meaningful name for the method.

5 Create a test step

Drag the method you created above, into the **Test Flow** of the canvas. Expand the REST service step and click on the **HTTP Activity** box.

6 Locate the endpoint URL

Locate your REST service design document and copy the URL to the clipboard.

7 Enter the input values

a Open the **Input/Checkpoints** view in the Property Sheet.

b Paste the clipboard contents, the endpoint URL, into the **URL** row.

c Select the **HTTP Method**: GET, PUT, POST and so forth.

d Expand the **Request Headers** node. Based on the operation details on the Help page, set the header name and value as follows: Type in the key and its value. For example, in the **Key** row you could type *Content-Type* and in the **Value** row, *text/xml*.

8 Locate the request body

a Return to the design document for the REST service and copy the Request body onto the clipboard.

b In the Property Sheet, open the **HTTP** view and select a **Request Body** type from the drop-down list.

- To paste in plain text, select the **Text** type. Copy the Request body from the design document to the clipboard. Click within the **Body** section and press Ctrl-V to paste in the contents of the request. Modify the element values within the body as required.
- For the **XML** type, click **Import Schema** to load an XSD file. Edit the values for the schema in either the **Grid** or **Text** views.
- For a **File** type, browse to the file with the Request body.
- For a **Post Form**, enter the keys and their values.

9 Run the test and view the report

Click the **Run** button or F5 to compile and run the test. After the test run, the Run Results Viewer opens.

In the left pane of the viewer window, select **Expand All** from the shortcut menu. Select the **HTTP** node. In the **Captured Data** pane, verify that the **Response Body** values are correct.

How to Create a Template-Based REST Service

This task describes how to set up a REST service template. You define the request details and parameters of the service directly from the toolbox, before dragging into a the Test Flow. Once you create the template, you can incorporate the REST activity in your test with minimal configurations.

This task includes the following steps:

- "Add a new REST service" on page 161
- "Add a resource" on page 161
- "Add a method" on page 161

➤ "Locate the endpoint URL" on page 161
➤ "Enter the REST input parameters - optional" on page 161
➤ "Define custom input properties - optional" on page 162
➤ "Enter the Request body directly - optional" on page 162
➤ "Link the body request to a REST Input parameter - optional" on page 162

1 Prerequisite

Study the structure of the REST Service body and determine which resources and methods you need to define.

2 Add a new REST service

In the Toolbox palette select the **REST Services** node. Choose **Add Service** from the shortcut menu. Provide a meaningful name for the service.

3 Add a resource

Select the service node and choose **Add Resource** from the shortcut menu. Provide a meaningful name for the resource. Repeat as required.

4 Add a method

Select the service or resource node, and choose **Add Method** from the shortcut menu. Provide a meaningful name for the method.

5 Locate the endpoint URL

In your REST Service design document, determine the endpoint URL and copy it onto the clipboard.

6 Enter the REST input parameters - optional

a In the Toolbox, click on the method you created above. Open the **HTTP Input/Checkpoints** view in the Property Sheet.

b Paste the clipboard contents, the endpoint URL, into the **URL** row.

c Select the **HTTP method**: **GET**, **PUT**, **POST**, and so forth.

d Expand the **Request Headers** node. Based on the operation details on the Help page, set the header name and value as follows: Type in the key and its value. For example, in the **Key** row you could type *Content-Type* and in the **Value** row, *text/xml*.

7 Define custom input properties - optional

For methods that require input, such as **PUT** or **POST**, add the required input properties. For methods that provide an output such as **GET**, add the required output properties.

a In the **Toolbox**, select the method you created above. Open the **Custom Input/Checkpoints** view in the Property Sheet.

b Expand the plus button and select **Add Input/Output Property**.

c Add custom properties as required. Provide a name and data type for each property.

d Enter the property values in the **Value** column.

8 Enter the Request body directly - optional

You can enter the Request body directly or data drive the values to use a data source expression. To data drive the values, see the next step.

a Return to the design document for the REST service and copy the Request body onto the clipboard.

b In the Property Sheet, open the **HTTP** view and click within the **Body** section. Press Ctrl-V to paste in the contents of the request. Make sure that there no extra spaces before or after the body text.

c Modify the element values within the XML body as required.

9 Link the body request to a REST Input parameter - optional

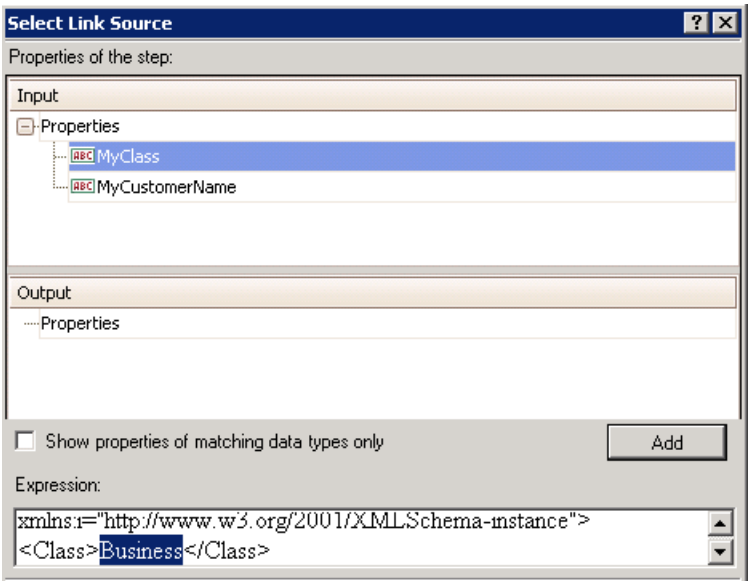
a Return to the design document for the REST service and copy the Request body onto the clipboard.

b In the Property Sheet, open the **HTTP** view and select a **Request Body** of Text. click within the **Body** section.

c Click the **Link to a data source** button to the right of the pane, to open the Select Link Source dialog box.

d In the Select Link Source dialog box, click **More** to expand the box. option. Paste the clipboard contents, the Request body, into the **Expression** box.

e In the test, highlight the value of the element you want to link. In the upper pane, double-click on the custom parameter you defined in step 7. This links the property to a value.



f Repeat the same for all other elements that you want to data drive. Note the expression added in the Expression box in the bottom pane.

```
<Class>{Step.MyClass}</Class>
<CustomerName>{Step.MyCustomerName}</CustomerName>
```

g Click **OK**. Service Test places the data in the HTTP body section.

You have now created a template for your REST method, complete with input parameters and the HTTP information. You can now drag the activity into the Test Flow and run it with minimal intervention.

For an example of creating a REST Service with data driving and data tables, see the *HP Service Test Tutorial* provided with the product.

Reference

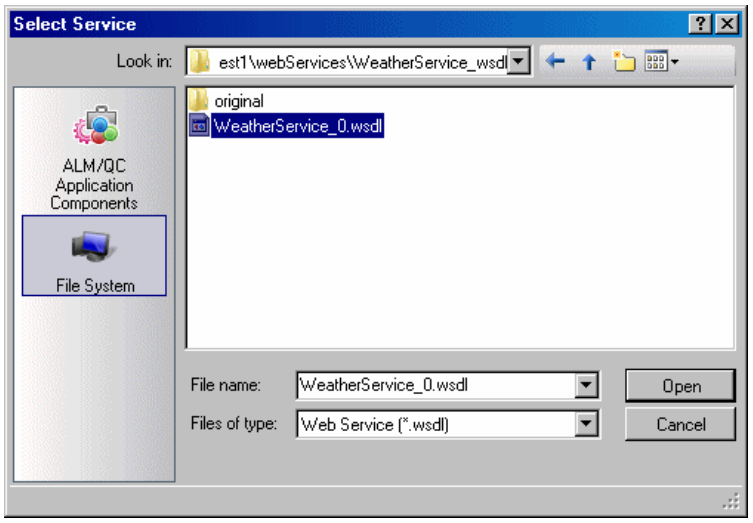
Import Service User Interface

This section includes:

- Select Service Dialog Box on page 165
- Import Service from URL or UDDI Dialog Box on page 167
- Select Service from UDDI Dialog Box on page 168

Select Service Dialog Box

This dialog box enables you to import WSDLs from a file system or Service Test Management, the ALM/QC extension for working with application components.



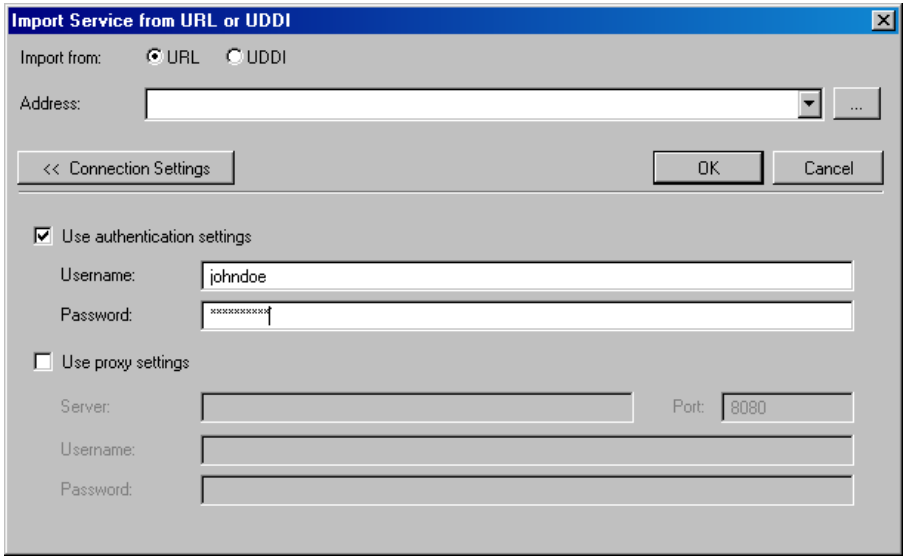
To access	Select **Import Service >Import Service from File or ALM/QC Application Component.**
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 152

The following elements are included (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
ALM/QC Application Components	Show WSDLs in the ALM/QC repository. **Note:** Only visible if there is an open connection to ALM/QC with a Service Test Management extension.
File System	Show WSDLs in the file system.
File name	The contract name: ➤ **File System.** The file name of the WSDL. ➤ **ALM/QC Application Components.** The name of the application component or service.
Files of type	The document type: **WSDL** extension or **All files**.

Import Service from URL or UDDI Dialog Box

This dialog box enables you to import WSDLs through a URL or UDDI.



To access	Select **Import Service** > **Import Service from URL or UDDI**.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 152

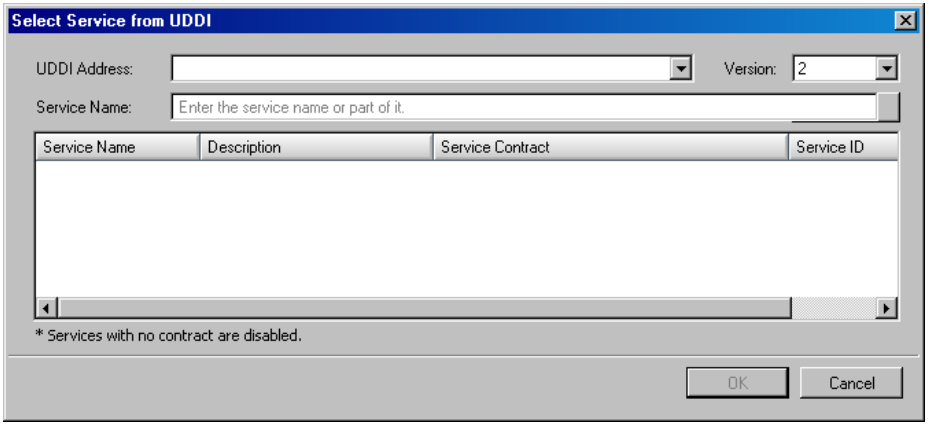
The following elements are included (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
Connection Settings >>	Expands the dialog box to show the authentication and proxy settings.
...	**Browse**. ➤ For **URL** imports: Opens a browser to allow you to navigate to a URL. ➤ For **UDDI** Imports: Opens the Select Service from UDDI Dialog Box.

UI Elements	Description
Address	The URL or UDDI path of the WSDL.
Use authentication settings	The authentication settings by which to access the WSDL: **Username** and **Password**.
Use proxy settings	The authentication settings for the proxy server hosting the WSDL: **Server**, **Port**, **Username**, and **Password**.

Select Service from UDDI Dialog Box

This dialog box enables you to select and import services from a UDDI (Universal Description, Discovery, and Integration) server.



To access	Select **Import Service** > **Import Service from URL or UDDI**. Select **Import from**: **UDDI** and click the ... button.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 152
See also	"Import Service from URL or UDDI Dialog Box" on page 167

The following elements are included (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
Search	Retrieves the list of services that match the string in the **Service name** field. If no string is specified, it retrieves all available services.
UDDI Address	The name or IP address of the UDDI server inquiry API. **For Example:** http://lab1.com:8090/uddi/inquiry.
Version	The UDDI version—2 or 3.
Service Name	A string in the service name by which to filter the list.
<list of services>	A list of the services that match the filter criteria. The grid shows the following information: ➤ **Service Name** ➤ **Description** ➤ **Service Contract** ➤ **Service ID**

Update Service Interface

Service Test provides shortcut menu options for updating Web services that were imported into the Toolbox.

This section includes:

➤ Update Service and Port Wizard on page 170

➤ Update Step Wizard on page 173

For task details, see "How to Update a Web Service" on page 155.

For more information, see "Updating Services" on page 149.

Update Service and Port Wizard

This wizard enables you to update ports that became invalid as a result of an Update Service action. The steps became invalid because the port path (<service name:port name> combination) changed. The wizard allows you to map old port names to new ones.

To access	**1** Import a Web service using the Import Service toolbar button. **2** Select a service's parent node and choose **Update** from the shortcut menu. **Note:** This wizard only opens if there is a discrepancy between the service or port names.
Relevant tasks	"How to Update a Web Service" on page 155.
Wizard map	This wizard contains: <Service Name> Page > Finish Page
See also	"Updating Services" on page 149

<Service Name> Page

This wizard page enables you to map a port or service from the new service, with the original service's service name and port. This page is repeated for each of the services for which there is a conflict.

Important information	General information about this wizard is available here: "Update Step Wizard" on page 173.
Wizard map	This wizard contains: **<Service Name> Page** > Finish Page

User interface elements are described below (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
Show Match ->	Shows the service/port in the **Available Services and Ports** pane, that is mapped to the service/port selected in the **Original Service and Ports** pane.
<- Show Match	Shows the service/port in the **Original Service and Ports** pane, that is mapped to the service/port selected in the **Available Services and Ports** pane.
Map	Maps the selected service/port in the **Original Service and Ports** pane, to a service/port in the **Available Services and Ports** pane. After you map a service/port, the grid displays it in green. **Note:** This button is enabled only if you select a red or black service/port in the **Original Service and Ports** pane, and a service/port in the **Available Services and Ports** pane.
Unmap	Removes the mapping of the selected service/port in the **Original Service and Ports** pane, from a service/port in the **Available Services and Ports** pane. **Note:** This button is only enabled if you select the mapped service/port in both panes. To find the mapped service/port, use the **Show Match** buttons.
Original Service and Ports	A tree hierarchy of the first service or port for which a conflict was found. As you click **Next**, the wizard proceeds to the next conflict. The wizard displays the services and ports in the following colors: **Green.** A service or port that did not change in the update or that was resolved through mapping. **Red.** A services or port that changed during the update and requires mapping. **Black.** A service or port that was resolved automatically during the update, but you chose to unmap it.

UI Elements	Description
Available Services and Ports	A tree hierarchy of all of the available services and ports. The services and ports displayed in green text did not change in the update.
<mapped property list>	A list of all the mapped services and ports—the ones mapped automatically by Service Test and the ones mapped manually. **Tip:** Click on a set of services and ports to highlight them in the upper panes.

Finish Page

This wizard page indicates whether you successfully resolved the conflicts between your original services and ports and those from the updated service.

Important information	General information about this wizard is available here: "Update Service and Port Wizard" on page 170.
Wizard map	This wizard contains: <Service Name> Page > **Finish Page**

User interface elements are described below:

UI Elements	Description
	Closes the wizard and applies your changes to the test. **Note:** Enabled when you successfully map all unresolved services and ports, or when you choose to ignore unresolved conflicts.
Ignore unresolved services/ports	Ignores unresolved conflicts. If you intend to remove the unresolved steps, you can enable this option. **Tip:** To return to the wizard screens to resolve all of the services or ports, click **Back**.

Update Step Wizard

This wizard enables you to update test steps that became invalid as a result of an Update Service action. The steps became invalid because the operation was removed or changed, or property names were modified. The wizard allows you to map old operations and properties to new ones.

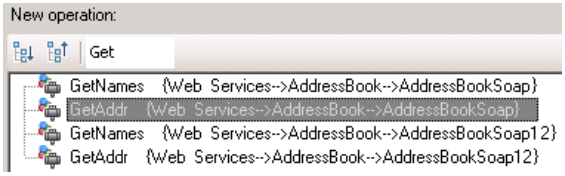
To access	Do the following: **1** Import a Web service using the **Import Service** toolbar button. **2** Drag a Web service from the **Toolbox** to the canvas and set input values. **3** Select a service's parent node and choose **Update** from the shortcut menu. Accept any warning popup messages. **4** Click on the alert adjacent to the step . **5** Click on the text **The step must be resolved. Resolve step...** **Note:** The wizard will open only when there is a change in properties names that were assigned values.
Relevant tasks	"How to Update a Web Service" on page 155.
Wizard map	This wizard contains: Select Operation Page > Update Input Properties Page > Update Output Properties Page > Finish Page
See also	"Updating Services" on page 149

Select Operation Page

This wizard page enables you to map an operation from the new service, with the original service's operation.

Important information	General information about this wizard is available here: "Update Step Wizard" on page 173.
Wizard map	This wizard contains: **Select Operation Page** > Update Input Properties Page > Update Output Properties Page > Finish Page

User interface elements are described below:

UI Elements	Description
Original Operation	A tree hierarchy of the operation in the selected test step.
New Operation	A tree hierarchy of the new service, its ports, and its operations.
	Expand All. Expands all nodes of the new service.
	Collapse All. Collapses all nodes of the new service.
Filter	Allows you to filter the operations in the service. For example to display only those operations containing the term Get, type Get into the **Filter** box. The filter results show the entire path of the operation in the toolbox. 

UI Elements	Description
Update the step	Instructs the wizard to update the step according to your choice in the **New Operation** pane. The **Next** button will proceed to the next conflict. To apply this mapping to all steps using the selected operation in the **Original Operation** pane, select **Apply to all steps of this type**. **Note:** This option is visible only when Service Test detects a conflict in the operation names.
Apply to all steps of this type	Converts all test steps using the original operation, to steps using the new operation name, selected in the **New Operation** pane. If you do not select this option, you will need to repeat the selection of an operation, for conflicts in other test steps. **Note:** This option is visible only when Service Test detects a conflict in the operation names.

Update Input Properties Page

This wizard page enables you to map new input properties with the original input properties.

Important information	General information about this wizard is available here: "Update Step Wizard" on page 173.
Wizard map	This wizard contains: Select Operation Page > **Update Input Properties Page** > Update Output Properties Page > Finish Page

User interface elements are described below (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
Show Match ->	Shows the property in the **New Properties** pane, that is mapped to the property selected in the **Original Properties** pane.
<- Show Match	Shows the property in the **Original Properties** pane, that is mapped to the property selected in the **New Properties** pane.
Map	Maps the selected property in the **Original Properties** pane, to a property in the **New Properties** pane. After you map a property, the grid displays it in green. **Note:** This button is enabled only if you select a red or black property in the **Original Properties** pane, and a property in the **New Properties** pane.
Unmap	Removes the mapping of the selected property in the **Original Properties** pane, from a property in the **New Properties** pane. **Note:** This button is enabled only if you select the mapped properties in both panes. To find the mapped properties, use the **Show Match** buttons.
Original Properties	A tree hierarchy of all of the original step's input properties. The text color indicates the status: **Green.** Properties that did not change in the update or that were already mapped. **Red.** Properties that changed during the update and require mapping. **Black.** Properties that were resolved automatically during the update, but you chose to unmap them.
New Properties	A tree hierarchy of all of the new operation's input properties. The properties displayed in green text, are properties that did not change in the update.

UI Elements	Description
Show only unmapped properties	Hides all non-red properties. The red properties are the ones that Service Test was unable to resolve automatically during the update, and that you did not previously resolve.
<mapped property list>	A list of all the mapped properties—the ones mapped automatically by Service Test and the ones mapped manually. **Tip:** Click on a set of properties to highlight them in the upper panes.

Update Output Properties Page

This wizard page enables you to map new output properties with the original output properties. If the step has no output properties, the wizard skips this step.

Important information	General information about this wizard is available here: "Update Step Wizard" on page 173.
Wizard map	This wizard contains: Select Operation Page > Update Input Properties Page > **Update Output Properties Page** > Finish Page

User interface elements are described below (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
	Shows the property in the **New Properties** pane, that is mapped to the property selected in the **Original Properties** pane.
	Shows the property in the **Original Properties** pane, that is mapped to the property selected in the **New Properties** pane.

UI Elements	Description
Map	Maps the selected property in the **Original Properties** pane, to a property in the **New Properties** pane. After you map a property, the grid displays it in green. **Note:** This button is enabled only if you select a red or black property in the **Original Properties** pane, and a property in the **New Properties** pane.
Unmap	Removes the mapping of the selected property in the **Original Properties** pane, from a property in the **New Properties** pane. **Note:** This button is enabled only if you select the mapped properties in both panes. To find the mapped properties, use the **Show Match** buttons.
Original Properties	A tree hierarchy of all of the original step's output properties. The text color indicates the status: **Green.** Properties that did not change in the update or that were already mapped. **Red.** Properties that changed during the update and require mapping. **Black.** Properties that were resolved automatically during the update, but you chose to unmap them.
New Properties	A tree hierarchy of all of the new operation's output properties. The properties displayed in green text, are properties that did not change in the update.
Show only unmapped properties	Hides all non-red properties. The red properties are the ones that Service Test was unable to resolve automatically during the update.
<mapped property list>	A list of all the mapped properties—the ones mapped automatically by Service Test and the ones mapped manually. **Tip:** Click on a set of properties to highlight them in the upper panes.

Finish Page

This wizard page indicates whether you successfully resolved the conflicts between your original properties and the new ones.

Important information	General information about this wizard is available here: "Update Step Wizard" on page 173.
Wizard map	This wizard contains: Select Operation Page > Update Input Properties Page > Update Output Properties Page > **Finish Page**

User interface elements are described below:

UI Elements	Description
	Closes the wizard and applies your changes to the test. **Note:** Enabled when you successfully map all unresolved properties, or when you choose to ignore unresolved properties.
Ignore unresolved properties	Ignores all unresolved input and output properties. If you intend to remove the step with the unresolved properties from your step, you can enable this option. **Tip:** To return to the wizard screens to resolve all of the properties, click **Back**.

Troubleshooting and Limitations - Services

This section describes troubleshooting and limitations for working with Web and REST services.

Web Services

- WSDL files named con.wsdl, cannot be imported.
  Workaround: Rename the WSDL file before the import.

REST Services

- Importing a schema to a REST, HTTP, or SOAP checkpoint may remove the links of the input properties.
- Copy and Paste is not supported for REST services.
- XPATH checkpoints are not supported for HTTP and REST activities.
- Links that are defined in the toolbox are deleted on the canvas.
- A link defined in the toolbox cannot be deleted.
- Dragging a method onto the canvas after an output property is deleted may cause the test to become unstable.
- When defining a REST method prototype in the toolbox - in order to use the **Trim**, **Ignore**, or **Stop test** options, select the check box in the **Validate** column, in the Checkpoints pane.

5

Web Service Security

This chapter includes:

Concepts

- Setting Security Overview on page 182
- Security Scenarios Overview on page 183
- WCF Scenario Settings on page 187
- Advanced Security Settings on page 191

Tasks

- How to Set Security for a Web Service on the Port Level on page 192
- How to Set Security for a Specific Web Service Step on page 194
- How to Set Security for a Standard Web Service on page 195
- How to Set Security for a WCF Service on page 197
- How to Set up Common Web Services Security Scenarios on page 198
- How to Customize Security for WCF Type Web Services on page 202
- How to Test Web Services that use WS-Security or SSL on page 207
- How to Set up Advanced Standards Testing on page 208

Reference

- Security User Interface on page 211

Troubleshooting and Limitations - Web Service Security on page 237

Concepts

Setting Security Overview

When building Web Service applications, there is a challenge in building scalable applications that are secure. You can secure Web Services by having the message sent over a secure transport, such as Secure Sockets Layer (SSL), or by applying security at the message level, also known as **WS-Security**.

For testing a secured service, answering the following questions will help you define your security scenario.

- Is there transport security, such as SSL? What is the HTTPS URL?
- Is basic authentication required?
- Is mutual authentication required?
- What type of security is required in the SOAP header?

For task details, see "How to Set Security for a Specific Web Service Step" on page 194.

Security Levels

Service Test lets you set the security for a service on two levels—port or step. If you define a service for a port, all of its operations use these settings, by default. When working in the canvas, you can override the default security and customize the security for a particular step.

For User Interface details, see "Security User Interface" on page 211.

Security Scenarios Overview

Service Test provides several built-in scenarios for configuring security in Web Service calls.

A security scenario represents a typical security implementation for a Web Service. It contains information such as authentication, encoding, proxy, certificates, and so forth.

A default **Web Service** scenario can be used for most Web services. It allows you to configure both transport and message-level security. Service Test support for message-level security lets you manually configure the security elements such as tokens, message signatures, and encryption. For details, see "Web Service Security Scenario" on page 184.

WCF scenarios allows you to configure security for HTTP or custom bindings and work with advanced specifications, such as **WS-SecureConversation**.

The built-in security scenarios are:

- Web Service Security Scenario
- WCF Service (CustomBinding) Scenario Overview
- WCF Service (Federation) Scenario Overview
- WCF Services (WSHttpBinding) Scenario Overview

Use the default Web Service scenario for:

- Simple Web Services where no advanced standards are required.
- Web services using HTTP transport level security.
- Web services using message level security (WS-Security) for SOAP 1.1

Use a WCF scenario for:

- WCF Services that utilize advanced security and WS-Specifications.
- Web services using message level security (WS-Security) for SOAP 1.2

Such services can be written in various platforms such as WCF (Windows Communication Foundation), Metro (WSIT), and Axis2. Service Test also supports proprietary standards and transports.

Web Service Security Scenario

The default Web Service scenario is based on the WS-Security specification. This scenario lets you place security credentials in the actual SOAP message.

When a SOAP message sender sends a request, the security credentials, known as **tokens**, are placed in the SOAP message. When the Web server receives the SOAP request, it does not need to send additional requests to verify the integrity of the sender. The server verifies that the credentials are authentic before letting the Web Service execute the application. By not having to go back to the source of the credentials, the application's scalability improves significantly.

To further secure Web Services, it is common to use digital signatures or encryption for the SOAP messages. Digitally signing a SOAP message verifies that the message has not been altered during transmission. Encrypting a SOAP message helps secure a Web Service by making it difficult for anyone other than the intended recipient to read the contents of the message.

The Security Settings dialog box provides the following tabs for the Web Service scenario: **HTTP, WS-Security**, and **WS-Addressing**. The **HTTP** tab lets you configure the transport security, the **WS-Security** tab handles the message-level security, and the **WS-Addressing** tab sets the addressing version.

For user interface details, see "Security Settings View" on page 30.

Transport Level Security

The transport level security includes the authentication and proxy server information. You can also specify Keep Alive preferences and connection timeout. For user interface details, see "Security Settings View" on page 30.

Message Level Security

The **WS-Security** tab lets you set the message level security through tokens, signatures and encryption.

To support WS-Security, Service Test allows you to create security tokens for your script. You can create multiple tokens and set their properties. After creating a token, you use it to sign or encrypt a SOAP message.

The Web Services security mechanism associates security tokens with messages. This mechanism supports several security token formats to accommodate a variety of authentication requirements. For example, a client might need to provide a proof of identity or a security certificate.

The available tokens are: **UserName**, **X509 Certificate**, **Kerberos**, **Kerberos2,** and **SAML**.

- **UserName.** The **User Name** token contains user identification information for the purpose of authentication: **User Name** and **Password**. You can also specify Password Options, indicating how to send the password to the server for authentication: **Text**, **None**, or **Hash**. and indicate whether to include a timestamp.
- **X509 Certificate.** This token is based on an X.509 certificate. To obtain a certificate, you can either purchase it from a certificate authority, such as VeriSign, Inc. or set up your own certificate service to issue a certificate. Most Windows servers support the public key infrastructure (PKI), which enables you to create certificates. You can then have it signed by a certificate authority or use an unsigned certificate.
- **Kerberos /Kerberos2.** (For Windows 2003 or XP SP1 and later). The Kerberos protocol is used to mutually authenticate users and services on an open and unsecured network. Using shared secret keys, it encrypts and signs user credentials. A third party, known as a KDC (Kerberos Key Distribution Center), authenticates the credentials. After authentication, the user may request a service ticket to access one or more services on the network. The ticket includes the encrypted, authenticated identity of the user. The tickets are obtained using the current user's credentials.

  The primary difference between the Kerberos and Kerberos2 tokens, is that Kerberos2 uses the Security Support Provider Interface (SSPI), so it does not require elevated privileges to impersonate the client's identity. In addition, the Kerberos2 security token can be used to secure SOAP messages sent to a Web Service running in a Web farm.
- **SAML Token.** SAML is an XML standard for exchanging security-related information, called assertions, between business partners over the Internet. The assertions can include attribute statements, authentication, decision statements, and authorization decision statements.

SAML uses brokered authentication with a security token issued by STS (Security Token Service). The STS is trusted by the client and the Web Service to provide interoperable security tokens. SAML tokens are important for Web Service security because they provide cross-platform interoperability and a means of exchanging information between clients and services that do not reside within a single security domain.

Message Signatures and Encrypted Data

When you add a security token to a SOAP message, it is added to the SOAP message in the form of an XML element in the WS-Security SOAP header.

The message, however, is exposed and therefore requires additional security. This is especially true when the credentials, including the password, are sent in plain text as it is with role-based security.

The two methods used to secure the data are message signatures and message encryption.

- **Message Signatures.** Message Signatures are used by the recipients to verify that messages were not altered since their signing. The signature is in the form of XML within the SOAP message. The recipient checks the signature to make sure it is valid.
- **Message Encryption.** Although the XML message signature offers a mechanism for verifying that the message has not been altered since it was signed, it does not encrypt the SOAP message—the message is still plain text in XML format. To secure the message in order that it should not be exposed, you encrypt it, making it difficult for an intruder to view and obtain a user's password.

For task details, see "How to Set Security for a Specific Web Service Step" on page 194.

WCF Scenario Settings

This section describes the WCF security scenarios. It includes:

- "WCF Service (CustomBinding) Scenario Overview" on page 187
- "WCF Service (Federation) Scenario Overview" on page 188
- "WCF Services (WSHttpBinding) Scenario Overview" on page 189

WCF Service (CustomBinding) Scenario Overview

The **WCF Service (CustomBinding)** scenario enables the highest degree of customization. Since it is based upon WCF customBinding standard, it allows you to test most WCF services, along with services on other platforms such as Java-based services that use the WS - <*spec_name*> specifications.

Use the **WCF Service (CustomBinding)** scenario to configure a scenario that does not comply with any of the predefined security scenarios. You can customize the transport and encoding settings:

- **Transport.** HTTP, HTTPS, TCP, or NamedPipe
- **Encoding.** Text, MTOM, or WCF Binary

You can also provide additional security information:

- **Authentication mode.** The type of authentication, such as **None**, **AnonymousForCertificate**, and so forth. The options are available from the drop-down list.
- **Bootstrap policy.** For the **SecureConversation** authentication mode, you can specify a bootstrap policy.
- **Net security.** The network security for **TCP** and **Named Pipe** type transports. Typical values are **None**, **Windows stream security**, or **SSL stream security** available from the field's drop-down list. For services with HTTP transport, you should set the value to **None**. To enable SSL for HTTP, choose **HTTPS transport**.

For task details, see "How to Set Security for a WCF Service" on page 197.

For user interface details, see "WCF Service (Federation) Scenario" on page 223.

Note: For WSE3 security configurations, use the **WCF Service (CustomBinding)** Scenario. For details, see "How do I test a WCF service?" on page 203.

WCF Service (Federation) Scenario Overview

In the **WCF Service (Federation)** scenario, the client authenticates against the STS (Security Token Service) to obtain a token. The client uses the token to authenticate against the application server.

Therefore, two bindings are needed, one against the STS and another against the application server.

You define the bindings in two stages:

- Define an STS binding in the **Referenced binding** field.
- Provide security details for the application server's security scenario in the following areas:
  - **Server.** The transport and encoding methods.
  - **Security.** The authentication mode, such as IssuedToken, SecureConversation, and so forth.
  - **Identity.** Information about the server certificate and expected DNS.
  - **STS.** Settings related to the STS, such as the endpoint address and binding.

For task details, see "How to Set Security for a WCF Service" on page 197.

For user interface details, see "WCF Service (Federation) Scenario" on page 223.

WCF Services (WSHttpBinding) Scenario Overview

Note: The **WCF Service (WSHttpBinding)** scenario only supports the testing of WCF services which utilize *wsHttpBinding* and incorporate some level of security. To test WCF services that use *wsHttpBinding* but have no security, use the **WCF Service (CustomBinding)** scenario.

In the **WCF Service (WSHttpBinding)** scenario, you can choose from several types of authentication: None, Windows, Certificate, or Username (message protection).

For all of the authentication types, you can apply advanced settings as described in "Advanced Security Settings" on page 191.

For user interface details, see "WCF Service (WSHttpBinding) Scenario" on page 225.

No Authentication (Anonymous)

In this scenario, the client uses the server's certificate to encrypt a message; there is no client authentication. Use this scenario to test Web Services where the:

- Client uses the server's X.509 certificate for encryption
- Client is not authenticated
- Communication may utilize advanced standards such as secure conversation or MTOM.

Windows Authentication

This scenario uses Windows Authentication. If you are testing a WCF service that has not been customized and uses the default configuration, use this type of scenario.

Use this scenario to test Web Services where the:

- Client and server use Windows authentication
- Security is based on Kerberos or SPNEGO negotiations
- Communication may utilize advanced standards such as secure conversation or MTOM.

Certificate Authentication

In this **WCF WSHttpBinding** scenario, the client uses the server's X.509 certificate to encrypt the message and its own certificate for a signature.

Use this scenario to test Web Services where the:

- Client uses the server's X.509 certificate for encryption.
- Client uses its own X.509 certificate for signatures.
- Communication may utilize advanced standards such as secure conversation or MTOM.

Username Authentication (Message Protection)

In the **WCF WSHttpBinding** scenario, the client uses the server's X.509 certificate to encrypt the message, and sends a user name and password to authenticate itself.

Use this scenario to test Web Services where the:

- Client uses the server's X.509 certificate for encryption.
- Client is authenticated with a username and password.
- Communication may utilize advanced standards such as secure conversation or MTOM.

Advanced Security Settings

This scenario's settings let you customize a **WCF Service (CustomBinding)** scenario in the areas of **Encoding**, **Advanced Standards**, **Security**, or **HTTP and Proxy**.

Not all settings are relevant for all scenarios, so some of them might be disabled or hidden depending on the scenario.

For details, see the "Advanced Settings Dialog Box" on page 229.

Tasks

How to Set Security for a Web Service on the Port Level

This task describes how to configure security settings for a Web service on the port level. You can override this by modifying the settings for a specific test step. For details, see "How to Set Security for a Specific Web Service Step" on page 194.

This task includes the following steps:

- "Prerequisites" on page 193
- "Open the Security Setting dialog box" on page 193
- "Load existing settings - optional" on page 193
- "Create a new security scenario" on page 193
- "Save the settings - optional" on page 193

1 Prerequisites

Import at least one Web Service.

2 Open the Security Setting dialog box

Select a Web service's port node in the toolbox and select **Security Settings** from the shortcut menu.

3 Load existing settings - optional

To load an existing set of Service Test security settings, click **Import** and locate the **.stss** (Service Test Security Scenario) file.

4 Create a new security scenario

Use the Security Settings dialog box to create a new security scenario or modify a loaded one:

a In the Service Details box, select a scenario type: Web Service, WCF Service, and so forth.

b Configure the security settings for the selected scenario. For details, see the "Security Settings for Port <Port_Name> Dialog Box" on page 212.

c For Web Service type scenarios, add tokens, signatures, and encryption. For details, see "Web Service Scenario" on page 214.

5 Save the settings - optional

Click **Save** to save the settings to a an **.stss** file.

How to Set Security for a Specific Web Service Step

This task describes how to configure security settings for Web Services.

This task includes the following steps:

- "Prerequisites" on page 194
- "Open the Security Setting view" on page 194
- "Enable the Service Settings details" on page 194
- "Specify a scenario type" on page 194

1 Prerequisites

Import a Web Service and drag an operation into the canvas.

2 Open the Security Setting view

Click in the Property Sheet to open the **Security Setting** view.

3 Enable the Service Settings details

Clear the **Use the port's security settings** option.

4 Specify a scenario type

In the **Service Details** list, select a scenario type: Web Service, WCF Service, and so forth.

5 Configure the security settings

Configure the security settings for the selected scenario. For task details, see one of the following sections

- "How to Set Security for a Standard Web Service" on page 195
- "How to Set Security for a WCF Service" on page 197

For user interface information, see the "Security Settings for Port <Port_Name> Dialog Box" on page 212.

How to Set Security for a Standard Web Service

This task describes how to configure security settings for a standard Web Service. This mode lets you define the HTTP transport information and security elements such as tokens. For UI details, see "Web Service Scenario" on page 214.

This task includes the following steps:

- "Create a Web Service scenario" on page 196
- "Configure the HTTP settings" on page 196
- "Define security elements (optional)" on page 196
- "Set the WS-Addressing version (optional)" on page 197

1 Create a Web Service scenario

Open the Service Settings dialog box or the **Security Settings** view in the Property Sheet and select **Web Service** In the **Service Details** list (default). For details, see the "Security Settings for Port <Port_Name> Dialog Box" on page 212.

2 Configure the HTTP settings

Open the **HTTP** tab, and set the transport and proxy information. For details, see the "HTTP tab" on page 214.

3 Define security elements (optional)

Click the **WS-Security** tab. Add tokens, message signatures, and encryption.

- To add a token, click and select a token type. Provide the token details in the lower pane. The fields differ based on the token type. For details, see "Message Level Security" on page 184.

Note: When adding a SAML token, if you have the complete SAML token string, you can paste it directly into the **AssertionIDReference** field. If you do not have the complete token string and you want to configure the token manually, make sure that the first row in the schema contains the **Assertion** property—not the **AssertionIDReference**. You can change this using the grid's drop-down menu.

- To add a message signature, (you must first add at least one token), click and select a token in the **Signing token** box, usually an X.509 token. Provide the other required information: For details, see the "WS-Security tab" on page 215
- To add a message encryption, (you must first add at least one token), click and select the token that will do the encryption in the **Encryption token** box. Provide any other required information or accept the defaults.

➤ Organize the security elements in their order of priority. Use the **Up** and **Down** arrows to set the priorities. The basic order is tokens, followed by message signatures, and then encryption. In addition, your service may also require a specific order for the tokens.

4 Set the WS-Addressing version (optional)

Click the **WS-Addressing** tab. Select the relevant version or None if WS-Addressing is not used.

For details about the security elements. see the "Web Service Security Scenario" on page 184.

For user interface details see the "Security Settings for Port <Port_Name> Dialog Box" on page 212.

How to Set Security for a WCF Service

This task describes how to configure security settings for a Web Service using WCF. For guidelines about selecting a WCF service scenario, see "WCF Scenario Settings" on page 187.

This task includes the following steps:

➤ "Create a WCF scenario" on page 197

➤ "Configure the security settings" on page 198

➤ "Configure advanced settings - optional" on page 198

➤ "Save the scenario - optional" on page 198

1 Create a WCF scenario

➤ For port level security, click on the service's port in the Toolbox and and select **Security Settings** from the shortcut menu.

➤ For step level security, open the **Security Settings** view in the Property Sheet. Clear the **Use the port's security settings** option.

Select the desired **WCF Service** in the **Service Details** list.

2 Configure the security settings

Configure the settings as described in the "Security Settings for Port <Port_Name> Dialog Box" on page 212.

3 Configure advanced settings - optional

Click the **Advanced** button to configure advanced security settings. For details, see the "Advanced Settings Dialog Box" on page 229.

4 Save the scenario - optional

Your security scenario is automatically saved with the test. If, however, you also want to use the settings for another test, without having to redefine the scenario, you can save it to an **.stss** file.

To save the scenario, click the **Save** button. Specify a location for the scenario file. To use the file in another test, click **Import**.

How to Set up Common Web Services Security Scenarios

This section illustrates several common security scenarios. The examples apply when using the default Web Service security scenario. For WCF- based services, the **WCF Service (Custom Binding)** scenario is recommended. For additional examples, see "How to Customize Security for WCF Type Web Services" on page 202.

You can set security for all operations in a Web service port or for a specific step in your test.

To set security for a port, see "How to Set Security for a Specific Web Service Step" on page 194.

This section includes the following:

- "Authenticating with a Username Token" on page 199
- "Signing with an X.509 Certificate" on page 199
- "Signing a Specific Element with an X.509 Certificate" on page 200
- "Encrypting with a Certificate" on page 200
- "Authenticating with a Username Token and Encrypting with an X.509 Certificate" on page 201
- "Encrypting and Signing a Message" on page 202

Authenticating with a Username Token

To send a message level username/password token (a UserName token):

1. Select the **Web Service** scenario from the **Service Details** list, and click the **WS-Security** tab.
2. Click the **Add Token** button and add a **Username** token.
3. Customize the token details, such as username and password.

Signing with an X.509 Certificate

To send a message using a X.509 certificate for a digital signature.

1. Select the **Web Service** scenario from the **Service Details** list, and click the **WS-Security** tab.
2. Select **X509 Certificate** token from the **Security Token** drop-down list.
3. Fill in the token details to reference your private certificate. Make sure to enter a value in the **Token name** field.
4. Select a **Reference type**. Since this token is used for a signature, the most common type is BinarySecurityToken.
5. Click the **Add Message Signature** button. In the **Signing Token** drop-down, select the token you created in the previous steps.

Note: The certificate needs to be installed in the Windows certificate store. In the example above, you need to set the actual store name, store location, and subject name of your certificate.

Signing a Specific Element with an X.509 Certificate

It is possible to sign only a specific element in a message. The following example signs a specific element using an XPATH expression.

To send a message using an X.509 certificate for a digital signature:

1 Select the **Web Service** scenario from the **Service Details** list, and click the **WS-Security** tab.

2 Select **X509 Certificate** token from the **Security Token** drop-down list.

3 Fill the token details to reference your private certificate. Make sure to enter a value in the **Token name** field.

4 Select a **Reference type**. Since this token is used for a signature, the most common type is BinarySecurityToken.

5 Click the **Add Message Signature** button. In the **Signing Token** drop-down, select the token you created in the previous steps.

6 Scroll down to the **XPath** field. Enter an XPath expression, for example:, // *[local-name(.)='Body'].

Encrypting with a Certificate

To encrypt a message using the service's X.509 certificate:

1 Select the **Web Service** scenario from the **Service Details** list, and click the **WS-Security** tab.

2 Select **X509 Certificate** token from the **Security Token** drop-down list.

3 Fill in the token details to reference the server's public certificate. Make sure to enter a value in the **Token name** field.

4 Since this token is used for encryption, use Reference as the **Reference type**.

5 Click the **Add Message Encryption** button. In the drop-down list, select the token you created in the previous steps.

6 Scroll down to the **XPath** field. Enter an XPath expression, for example:, // *[local-name(.)='Body'].

Authenticating with a Username Token and Encrypting with an X.509 Certificate

The following section describes how to send a **Username** token to the service and encrypt it with the server's **X.509** certificate:

1 Select the **Web Service** scenario from the **Service Details** list, and click the **WS-Security** tab.

2 Select **Username Token** from the **Security Token** drop-down list. Provide the token details.

3 Select **X509 Certificate Token** from the **Security Token** drop-down list.

4 Fill the token details to reference the server's public certificate. Make sure to enter a value in the **Token name** field. Since this token is used for encryption, use Reference as the **Reference type**.

5 Click the **Add Message Encryption** button. In the drop-down list, select the X.509 token you created in Step 3.

6 To encrypt a specific message, scroll down to the **XPath** field. Enter an XPath expression, for example:, // *[local-name(.)='Body'].

Encrypting and Signing a Message

This example shows how to sign a message using a private key and then encrypt it using the service's public key.

1 Select the **Web Service** scenario from the **Service Details** list, and click the **WS-Security** tab.

2 Select **X509 Certificate Token** from the **Security Token** drop-down list. Fill the token details to reference your private certificate. Make sure to enter a value in the **Token name** field. Select BinarySecurityToken as a **Reference type**, since this token is used for a signature.

3 Select **X509 Certificate Token** from the **Security Token** drop-down list to add another X.509 token. Fill the token details to reference your private certificate. Make sure to enter a value in the **Token name** field. Select Reference as a **Reference type**, since this token is used for a encryption.

4 Click the **Add Message Signature** button. In the drop-down list, select the X.509 token you created in Step 2.

5 Click the **Add Message Encryption** button. In the drop-down list, select the token you created in Step 3.

How to Customize Security for WCF Type Web Services

This section describes how to customize the security settings for Web services using WCF.

This section describes

- "How do I test a WCF service?" on page 203
- "How do I test a WCF service that uses WSHttpBinding?" on page 203
- "How do I test a WCF service that uses CustomBinding?" on page 204
- "How do I test a WCF service that uses netTcp or namedPipe transport?" on page 204
- "How do I test a Federation scenario that uses an STS (Security Token Service)?" on page 204

➤ "How do I test a scenario that uses a WSE3 security configuration with a server certificate?" on page 205

➤ "How do I test a scenario that uses mutual certificate authentication?" on page 206

➤ "How do I configure a WCF binding with TCP transport to require an X.509 client certificate?" on page 206

How do I test a WCF service?

In the **Service Details** list, select a WCF scenario.

If the Federation or WSHttpBinding scenarios are not appropriate, choose the **WCF Service** (C**ustom Binding)** scenario, as it can handle all other bindings.

How do I test a WCF service that uses WSHttpBinding?

WSHttpBinding is one of the most popular bindings in WCF. In order to use this binding, select the **WCF Service (Http Binding)** scenario from the **Scenario Details** list.

In the Client Authentication box, select the client credential type that you use in your binding—Windows, Certificate, or Username. This value corresponds to the **MessageClientCredentialType** property of the WCF's WSHttpBinding.

Windows authentication is the most common value for a WCF services. If you are using the WCF default settings for your service, use this option. Other options are username, certificate, or none.

For some scenarios you should indicate whether to use the WCF proprietary negotiation mechanism to get the service credentials.

Use the Advanced scenario properties to control the usage of a secure session.

For details, see "WCF Services (WSHttpBinding) Scenario Overview" on page 189.

How do I test a WCF service that uses CustomBinding?

Open the **Security View** in the Property Sheet and select the **WCF Service** (C**ustom Binding)** scenario.

You can then customize many binding elements, such as your transport method, encoding, security, and reliable messaging.

For details, see "Configure the security settings" on page 198.

How do I test a WCF service that uses netTcp or namedPipe transport?

Select the **WCF Service (**C**ustom Binding)** scenario from the **Scenario Details** list.

Configure the transport to **TCP** or **NamedPipe**.

For details, see "Configure the security settings" on page 198.

How do I test a Federation scenario that uses an STS (Security Token Service)?

For this scenario, you must define the communication properties for both the STS and the service. Use the built-in Federation scenario.

Select the **WCF Service (Federation)** scenario from the **Scenario Details** list.

For this scenario, you must to define the communication properties for both the STS and the application server.

For details, see "WCF Service (Federation) Scenario" on page 223.

How do I test a scenario that uses a WSE3 security configuration with a server certificate?

The following procedure describes how to set up a security scenario for WSE3.

1. Create a new test, import the WSDL for the W3ES service, and drag the operation into the canvas.
2. Open the **Security View** in the Property Sheet or from the port's shortcut menu. Select the **WCF Service** (C**ustom Binding)** scenario.
3. Set the Transport to **HTTP**, and the Encoding to **Text**.
4. Provide a username and password in the **Identities** section.
5. Click the **Browse** button adjacent to the Server Certificate field and specify the **Store Location**, **Store Name** and **Search text** (optional). Click **Find**, select the certificate, and click **Select**.
6. Provide the **Expected DNS**.
7. Click the **Advanced** button and configure the following settings:
   a. **Encoding** tab— Encoding: Text, WS-Addressing: WSA 04/08 (for example).
   b. **Security** tab:
      - **Enable secure session:** Enabled
      - **Negotiate service credentials:** Enabled
      - **Protection level:** Encrypt and Sign
      - **Message protection order:** Sign Before Encrypt
      - **Message security version:** WSSecurity11WSTrustFebruary2005WSSecureConversationFebruary2005 (first entry)
      - **Require Derived keys**: Enabled

For all other fields, use the default settings.

For details, see "Configure the security settings" on page 198.

How do I test a scenario that uses mutual certificate authentication?

The following procedure describes how to set up a security scenario for mutual certificates and how to comply with a WSE3 security configuration.

1. Select the **WCF Service (CustomBinding)** scenario from the **Scenario Details** list.
2. Set the **Transport** to HTTP, and the **Encoding** to Text.
3. Set the authentication mode to MutualCertificate.
4. In the **Identities** section, select server and client certificates. For details, see "Select Certificate Dialog Box" on page 235.
5. Provide the **Expected DNS**.
6. Click the Advanced button and configure the following settings:
   a. **Encoding** tab— **Encoding**: Text, **WS-Addressing**: WSA 04/08 (for a WSE3 security configuration).
   b. **Security** tab—**Require Derived keys**: Disabled

For all other fields, use the default settings.

For details, see "Configure the security settings" on page 198.

How do I configure a WCF binding with TCP transport to require an X.509 client certificate?

The following procedure describes how to configure a WCF custom scenario to require an X.509 client certificate in **nettcp.**

1. Select the **WCF Service (Custom Binding)** scenario from the **Scenario Details** list.
2. Set the Transport to **TCP** and the Net Security to **SSL stream security**.
3. Open the Property Sheet's Event view.
4. Select the **BeforeApplyProtocolSettings** event. Click in the **Handler** column and select **Create a default handler** from the drop-down. Service Test opens the **TestUserCode.cs** tab.

5 In the **TestUserCode.cs** file, add a reference to the following assembly "<*Installation_Folder*>\addins\ServiceTest\WSImportTechnology\HP.ST.Ext.CommunicationChannels.dll".

6 Add the following definitions to the implementation section of the **TestUserCode.cs** file.

```
var wcf = (HP.ST.Ext.CommunicationChannels.Models.WcfChannelBinding)args[1];
var ssl =
   (HP.ST.Ext.CommunicationChannels.Models.WcfSslStreamSecurityChannel)wcf.
    Protocols.Channels[1];
ssl.RequireClientCertificate = true;
```

For all other fields, use the default settings.

7 Save the test and run it.

How to Test Web Services that use WS-Security or SSL

This section provides a summary of using Service Test for general security testing.

This section inlcudes:

- "How do I test a Web Service that uses SSL?" on page 207
- "How do I test a Web Service that require Windows authentication at the HTTP level?" on page 208
- "How do I test a Web Service that uses WS-Security?" on page 208
- "How do I configure the low-level details of my WS-Security tokens?" on page 208

How do I test a Web Service that uses SSL?

Testing a secure site does not require any special configuration. If your service URL begins with **https**, SSL is automatically used.

If in addition to SSL you are using message-level security (for example a username) then you must configure the security for the message separately.

You can use the basic Web Services security scenario and specify the message-level security such as tokens and signatures.

You can also use the **WCF Service** (C**ustom Binding)** scenario, or the **WCF Service (Http Binding)** scenario with the transport credentials.

How do I test a Web Service that require Windows authentication at the HTTP level?

1 Select the **Web Service** scenario from the **Scenario Details** list.

2 In the **HTTP** tab, specify the credentials.

For details, see "How to Set Security for a Standard Web Service" on page 195.

How do I test a Web Service that uses WS-Security?

Use the basic **Web Services** security scenario and open the **WS-Security** tab. Add the message-level security such as tokens, signatures, and encryption.

How do I configure the low-level details of my WS-Security tokens?

In most cases, you can configure the low-level details as described in "Advanced Settings Dialog Box" on page 229.

How to Set up Advanced Standards Testing

This section provides guidelines for using Service Test in advanced standards testing.

This section inlcudes:

- "How do I test a Web Service that uses MTOM?" on page 209
- "How do I change the WS-Addressing version of a service?" on page 209
- "How do I enable support for a service or activity that uses 256-bit SSL encoding?" on page 209

How do I test a Web Service that uses MTOM?

1 Select the **WCF Service (Custom Binding)** scenario from the **Scenario Details** list.

2 Configure the **Encoding** to **MTOM**.

If your service requires advanced settings, click the **Advanced** button. For details, see the "Advanced Settings Dialog Box" on page 229.

For more information about the scenario, see "Configure the security settings" on page 198.

How do I change the WS-Addressing version of a service?

1 Select the **Web Service** scenario from the **Scenario Details** list.

2 Click the **WS-Addressing** tab and select a version.

For details, see "How to Set Security for a Web Service on the Port Level" on page 192.

If your service uses WCF, use the appropriate scenario and configure the addressing version from the Advanced window's **Encoding** tab. For details, see the "Advanced Settings Dialog Box" on page 229.

How do I enable support for a service or activity that uses 256-bit SSL encoding?

Change the SSL cipher order in Windows Vista so that AES256 precedes AES128 in the cipher list.

Tip: Check with an IT professional before performing the following actions.

To change the cipher order:

1 Type **gpedit.msc** at a command prompt to open your group policy editor.

2 Choose **Computer Configuration** > **Administrative Templates** > **Network** > **SSL Configuration Settings**.

3. Open the only item—**SSL Cipher Suite Order**.
4. Select **Enabled**.
5. The first item in the list is TLS_RSA_WITH_AES_128_CBC_SHA

   And the second item is TLS_RSA_WITH_AES_256_CBC_SHA
6. Change the first 128 to 256. Then move the cursor forward and change the 256 to 128.
7. Move the cursor through the list and change the cipher priorities as in the above step.
8. Close the group policy editor and reboot.

Reference

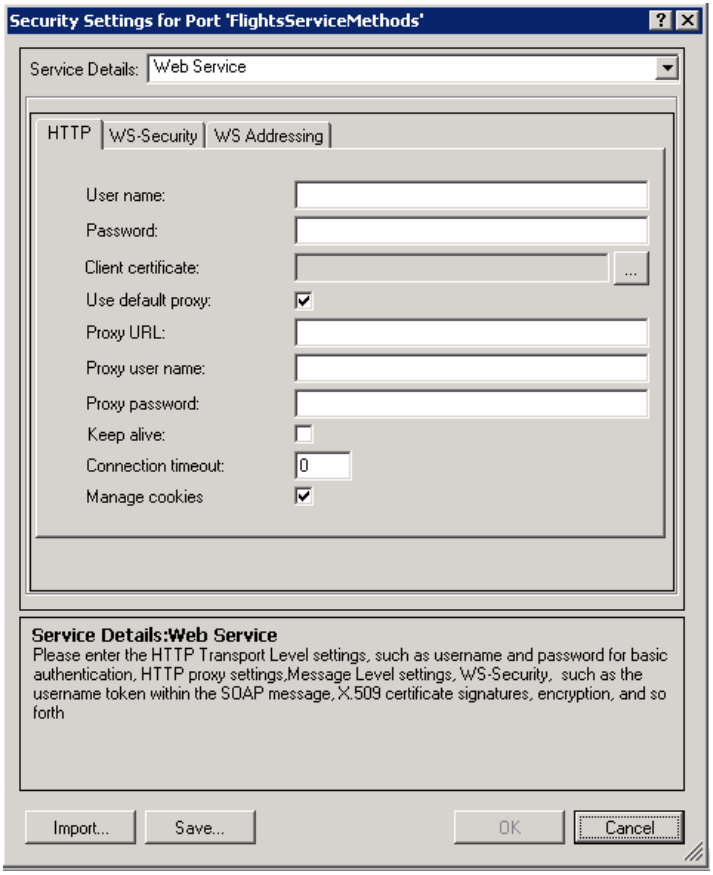
Security User Interface

This section includes:

- Security Settings for Port <Port_Name> Dialog Box on page 212
- Advanced Settings Dialog Box on page 229
- Select Certificate Dialog Box on page 235

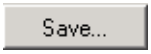
Security Settings for Port <Port_Name> Dialog Box

Using the Security Settings dialog box, you can configure security settings for all operations in a a Web service port. To set the security for a specific step within your test, use the **Security Settings** view in the Property Sheet. For details, see "Security Settings View" on page 67.



To access	Select a Web Services port node in the Toolbox palette and select **Security Setting** from the shortcut menu.
Important information	➤ For details about choosing a security scenario type, see "Security Scenarios Overview" on page 183. ➤ For examples, see "How to Set up Common Web Services Security Scenarios" on page 198.
Relevant tasks	"How to Set Security for a Specific Web Service Step" on page 194

User interface elements are described below:

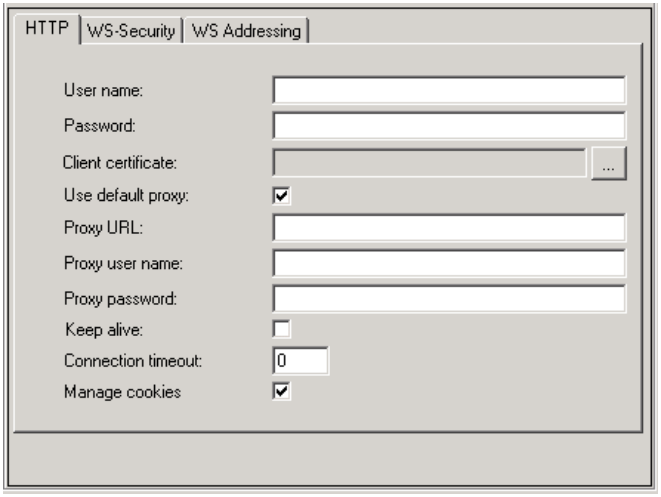
UI Elements	Description
	Saves the security scenario settings to an **.stss** (Service Test Security Scenario) file, for use in other tests. If you are connected to ALM/QC, it saves the file together with the test.
	Loads security settings from a previously saved **.stss** file.
	Opens the Advanced Settings dialog box. For details, see the "Advanced Settings Dialog Box" on page 229. **Note**: Available only for WCF type Web services.
Service Details	The type of Web service. After selecting a type, Service Test provides an interface for modifying the relevant security settings. The service types are: ➤ "Web Service Scenario" on page 214 ➤ "WCF Service (Custom Binding) Scenario" on page 221 ➤ "WCF Service (Federation) Scenario" on page 223 ➤ "WCF Service (WSHttpBinding) Scenario" on page 225

Web Service Scenario

The simple Web Service scenario provides the **HTTP**, **WS-Security**, and **WS-Addressing** tabs.

HTTP tab

The **HTTP** tab lets you provide the HTTP transport level settings such as user credentials for sending a message with basic authentication, proxy settings, message-level settings, encryption, and so forth.



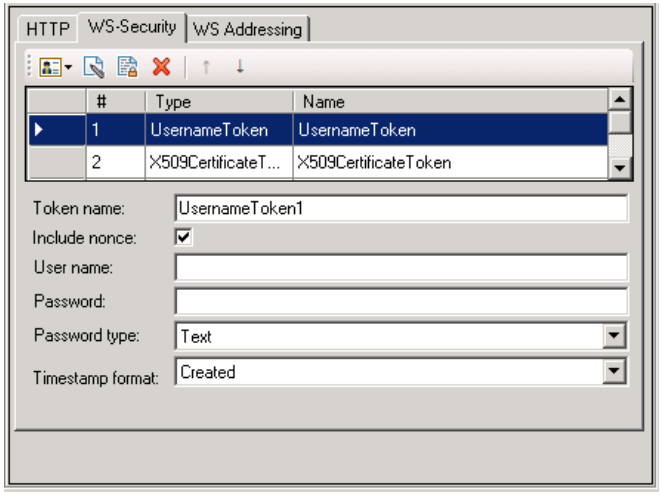
The user interface elements are described below.

UI Elements	Description
User name, Password	The credentials for HTTP authentication such as basic authentication, digest, or NTLM. For example, **User name**: myDomain\myUser **Password**: myPassword
Client certificate	The client credentials required for client certificate authentication when using two-way SSL scenarios. The **Browse** button opens the "Select Certificate Dialog Box" on page 235.

UI Elements	Description
Proxy URL	The URL and port of the proxy server through which the message must pass. For example, http://myProxy:8888/. To use the default, select **Use default proxy**.
Proxy user name, Proxy password	The credentials for the proxy server through which the message must pass.
Keep alive	Keeps the connection persistent.
Connection timeout	The time threshold in which to connect through the proxy server or with authentication.
Manage cookies	Enables the writing of cookie information.

WS-Security tab

The **WS-Security** tab provides an interface to add message level security using tokens, message signatures, and encryption.



The user interface elements are described below. (Unlabeled elements are shown in angle brackets).

UI Elements	Description
	Security Tokens. Allows you to add one of the following tokens: **User Name**, **X509**, **Kerberos**, **Kerberos2**, or **SAML**.
	Add Message Signature. Adds a signature to the message. This requires a token.
	Add Message Encryption. Adds encryption to the message. This requires a token.
	Delete. Removes the security element from the list.
	Up/Down. Positioning tools that allow you to set the priority of the security elements. **Important:** Make sure the security elements are positioned in order of their priority.
<security element list>	A list of the tokens, message signatures, and encryptions.
<Token details pane> - Username tokens	Token details for **Username** tokens. ➤ **Token name.** A meaningful name for the token (you can use the default value). ➤ **Include nonce**. Includes a nonce in the token. ➤ **User name, Password** ➤ **Password type**: Text, Hash, or None. ➤ **Timestamp format:** Full, Created, or None.

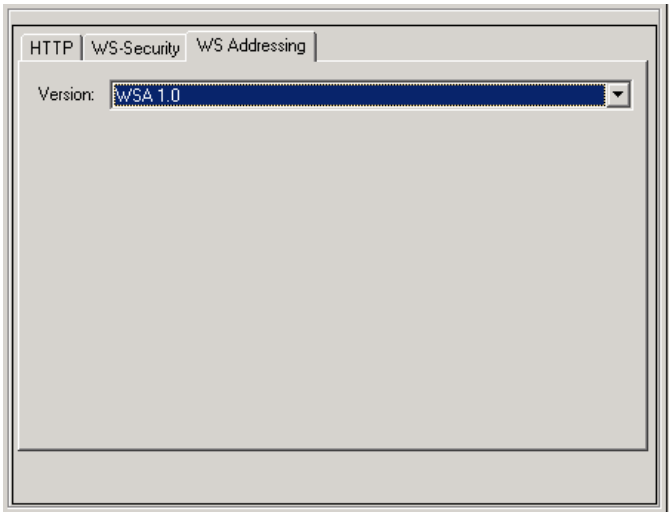
UI Elements	Description
<Token details pane> - X509 Certificate tokens	Token details for **X509 Certificate** tokens: ➤ **Token name.** A meaningful name for the token. ➤ **Certificate**. The path of the server certificate file. The **Browse** button opens the Select Certificate Dialog Box. ➤ **Reference type**: How the token should be referenced: BinarySecurity Token or Reference. When the certificate is used for encryption, for example, a service certificate, use Reference. When using it for a signature (for example, a certificate with your private key) choose BinarySecurity Token.
<Token details pane> - Kerberos tokens	Token details for **Kerberos** tokens: ➤ **Token Name.** A meaningful name for the token. ➤ **Host**. The host name of the server against which you want to authenticate. In most cases, it is the host portion of the service URL. ➤ **Domain.** The Windows domain of the server against which you want to authenticate.
<Token details pane> -SAML tokens	Token details for **SAML** tokens: ➤ **Load from file.** Allows you to browse to a SAML certificate. ➤ **Certificate**. The path of the certificate file. The **Browse** button opens the Select Certificate Dialog Box. ➤ **Certificate reference type.** How the certificate should be referenced— by X509 Data or RSA.

UI Elements	Description
<Signature details pane>	The details of the digital signature used to secure the token. ➤ **Signing token.** The token to use for signing, usually an X.509 type. Select from the list of all added tokens. ➤ **Canonicalization algorithm.** A URL for the algorithm to use for canonicalization. A drop-down list provides common algorithms. If you are unsure which value to use, keep the default. ➤ **Transform algorithm.** A URL for the Transform algorithm to apply to the message signature. A drop-down list provides common algorithms. If you are unsure which value to use, keep the default. ➤ **Inclusive namespaces list.** A list of comma-separated prefixes to be treated as inclusive (optional). ➤ **What to sign.** The SOAP elements to sign: SOAP Body, Timestamp, and WS-Addressing. ➤ **XPath (optional).** An XPath that specifies which parts in the message to sign. If left blank, the elements selected in the **Signature options** field are signed. For example, //*[local-name(.)='Body']. ➤ **Token (optional).** The target token you want to sign. Select from the drop-down list of all added tokens. With most services, this field should be left empty.

UI Elements	Description
<Encryption details pane>	The details of the encryption token. ➤ **Encrypting token.** The token to use for encryption, usually an X.509 type. You can select from a list of all previously created tokens. ➤ **Encrypting type.** Indicates whether to encrypt the whole destination Element or only its Content. ➤ **Key algorithm.** The algorithm to use for the encryption of the session key: RSA15 or RSAOAEP. ➤ **Session algorithm.** The algorithm to use for the encryption of the SOAP message. You can select from a list of common values. ➤ **XPath (optional).** An XPath that indicates the parts of the message to encrypt. If left blank, only the SOAP body is encrypted. ➤ **Token (optional).** The name of the encrypted token. A drop-down box provides a list of all added tokens. With most services, this field should be left empty.

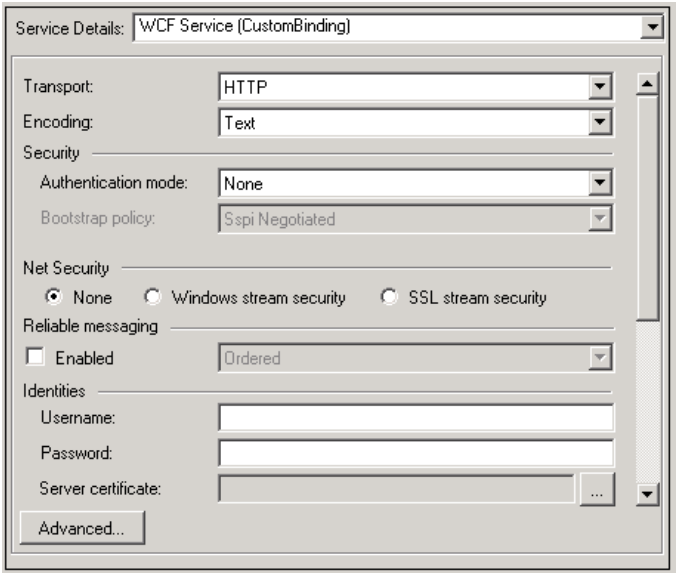
WS-Addressing tab

The **WS-Addressing** tab indicates whether WS-Addressing is used by the service, and if so, its version number.



WCF Service (Custom Binding) Scenario

Use this scenario to test WCF services which require security or transport configurations. For general details, see "WCF Service (CustomBinding) Scenario Overview" on page 187.



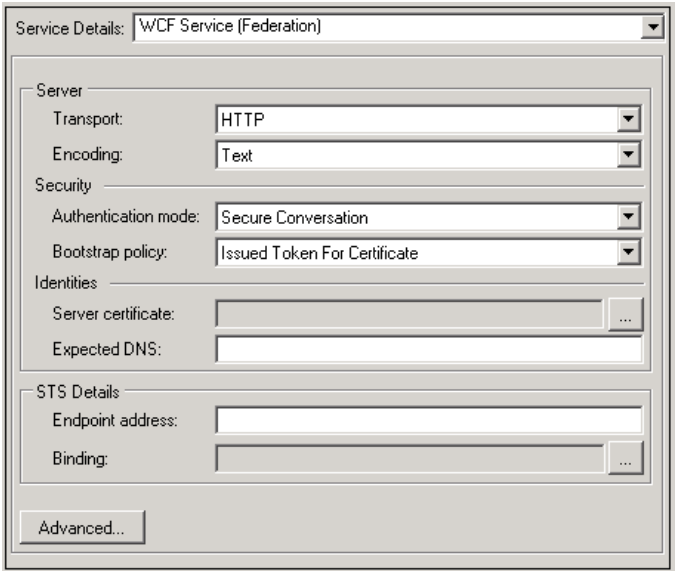
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Transport	**Transport type.** HTTP, HTTPS, TCP, NamedPipe, or AutoSecuredHTTP.
Encoding	**Encoding.** Text, MTOM, or WCF Binary

UI Elements	Description
Security	➤ **Authentication mode.** A drop-down list of possible modes of authentication, such as AnonymousForCertificate, MutualCertificate, and so forth. ➤ **Bootstrap Policy.** A drop-down list of possible bootstrap policies for Secure Conversation authentication., such as SspiNegotiated, UserNameOverTransport, and so forth.
Net Security	The type of stream security: **None, Windows stream security**, or **SSL stream security.**
Reliable Messaging	Enables **Reliable Messaging** in Ordered Or Non-ordered format.
Identities	The identity information for the bindings and certificate: ➤ **Username** and **Password** ➤ **Server /Client certificate.** A certificate that provides identity information for the server or client. Use the **Browse** button to open the Select Certificate Dialog Box. ➤ **Expected DNS**, **SPN**, and **UPN.** The expected identity of the server in terms of its DNS, SPN, or UPN. This can be localhost, an IP address, or a server name.
Client Windows Identity	The identity information for the client windows: ➤ **Current User.** The identity of the user logged onto the machine. ➤ **Custom User.** A user with the following credentials: **Username**, **Password**, and **Domain.**
Advanced...	Opens the Advanced Settings dialog box. For details, see "Advanced Settings Dialog Box" on page 229.

WCF Service (Federation) Scenario

In the **WCF Service (Federation)** scenario, the client authenticates against the STS (Security Token Service) to obtain a token. The client uses the token to authenticate against the application server.

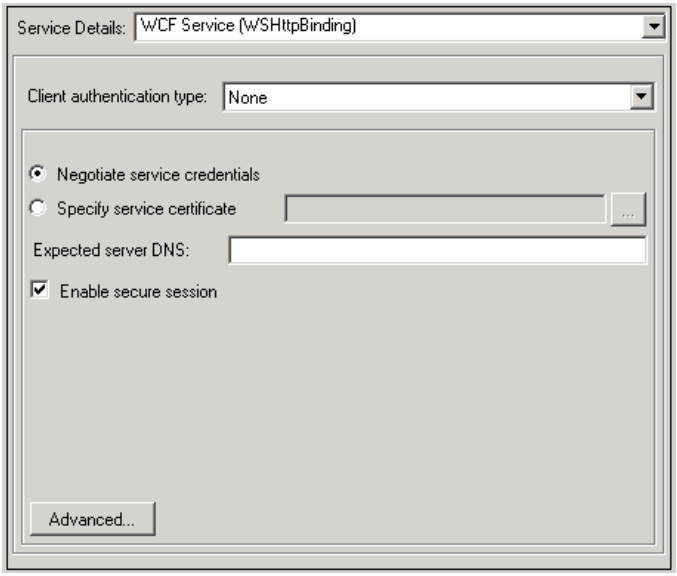


User interface elements are described below (unlabeled elements are shown in angle brackets). For details, see "WCF Service (Federation) Scenario Overview" on page 188.

UI Elements	Description
Server	➤ **Transport.** The transport type: HTTP or HTTPS. ➤ **Encoding.** The server's encoding policy: Text or MTOM.
Security	➤ **Authentication mode.** A drop-down list of possible modes of authentication, such as AnonymousForCertificate, MutualCertificate, and so forth. ➤ **Bootstrap Policy.** A drop-down list of possible bootstrap policies for Secure Conversation authentication., such as SspiNegotiated, UserNameOverTransport, and so forth.
Identities	The identity information for the bindings and certificate: ➤ **Server certificate.** A certificate that provides identity information for the server. Use the **Browse** button to open the Select Certificate Dialog Box. ➤ **Expected DNS.** The expected identity of the server in terms of its DNS. This can be localhost, an IP address, or a server name.
STS (Security Token Service) Details	Information about the STS: ➤ **Endpoint address.** The endpoint address of the STS. This can be localhost, an IP address, or a server name. ➤ **Binding.** The scenario which references the binding that contacts the STS.
Advanced...	Opens the Advanced Settings dialog box. For details, see "Advanced Settings Dialog Box" on page 229.

WCF Service (WSHttpBinding) Scenario

In the **WCF Service (WSHttpBinding)** scenario, you can choose from several types of authentication: None, Windows, Certificate, or Username (message protection). For details, see "WCF Services (WSHttpBinding) Scenario Overview" on page 189.



User interface elements are described below (unlabeled elements are shown in angle brackets) by the client authentication types:

UI Elements	Description
Advanced...	Opens the Advanced Settings dialog box. For details, see "Advanced Settings Dialog Box" on page 229.
Client authentication type	Authentication type: **None**, **Windows**, **Certificate**, or **Username**. For details, see below.

Client Authentication Types:

Client Authentication Type — None

UI Elements	Description
Negotiate server credentials	Negotiates the Web Service's certificate with the server. You can also provide the server's DNS information.
Specify service certificate	The location of the service's certificate. If you select this option, the **Negotiate service credentials** option is not relevant. For more information, see the "Select Certificate Dialog Box" on page 235.
Expected server DNS	The expected identity of the server in terms of its DNS. This can be **localhost**, an IP address, or a server name. It can also be the common name by which the certificate was issued.

Client Authentication Type — Windows

UI Elements	Description
Expected server identity	The expected server identity method: SPN or UPN.
Client Windows identity	The identity information for the client windows: ➤ **Current User.** The identity of the user logged onto the machine ➤ **Custom User.** A user with the following credentials: **Username**, **Password**, and **Domain**
Enable secure session	Allows a secure session using Windows type authentication.

Client Authentication Type — Certificate

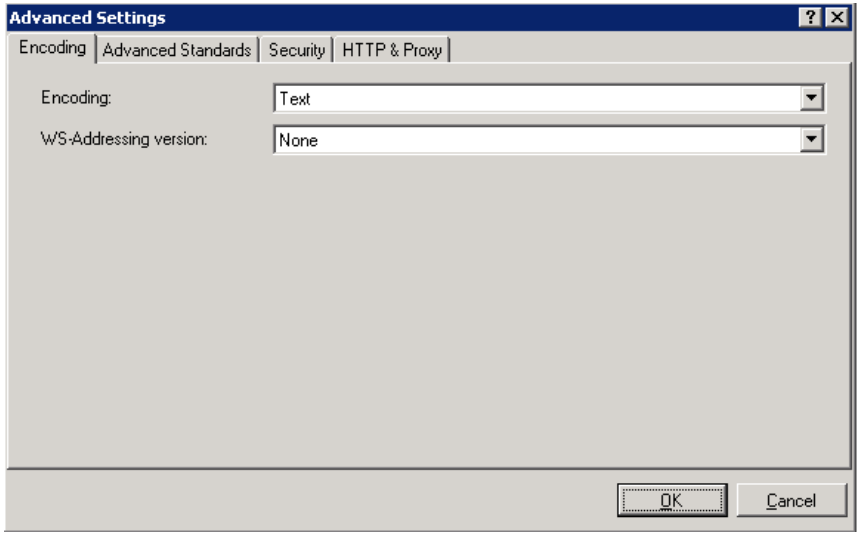
UI Elements	Description
Client certificate	The location of the client certificate. The **Browse** button opens the Select Certificate Dialog Box.
Negotiate server credentials	Negotiates the Web Service's certificate with the server. You can also provide the server's DNS information.
Specify service certificate	The location of the service's certificate. If you select this option, the **Negotiate server credentials** option is disabled. For more information, see the "Select Certificate Dialog Box" on page 235.
Expected server DNS	The expected identity of the server in terms of its DNS. This can be localhost, an IP address, or a server name. It can also be the common name by which the certificate was issued.
Enable secure session	Allows a secure session using Certificate type authentication.

Client Authentication Type — Username (message protection)

UI Elements	Description
Username, Password	The authentication credentials of the client.
Negotiate server credentials	Negotiates the Web Service's certificate with the server. You can also provide the server's DNS information.
Specify service certificate	The location of the service's certificate. If you select this option, the **Negotiate server credentials** option is disabled. For more information, see the "Select Certificate Dialog Box" on page 235.
Expected server DNS	The expected identity of the server in terms of its DNS. This can be **localhost**, an IP address, or a server name. It can also be the common name by which the certificate was issued.
Enable secure session	Allows a secure session using Username type authentication.

Advanced Settings Dialog Box

This dialog box allows you to customize the security settings for your test.



To access	Do the following: **1** Open the Security Settings for Port <Port_Name> Dialog Box. **2** Select a WCF Service scenario type from the **Service Details** list. **3** Click Advanced... .
Important information	➤ For details, see "Advanced Security Settings" on page 191.
Relevant tasks	➤ "How to Set Security for a Web Service on the Port Level" on page 192 ➤ "How to Set Security for a Specific Web Service Step" on page 194 ➤ "How to Set Security for a WCF Service" on page 197

This section includes:

- "Encoding Tab" on page 230
- "Advanced Standards Tab" on page 230
- "Security Tab" on page 231
- "HTTP and Proxy Tab" on page 234

Encoding Tab

The Encoding tab lets you indicate the type of encoding to use for the messages: **Text**, **MTOM**, or **WCF Binary**. The default is **Text** encoding.

The following table describes the UI elements:

UI Elements	Description
Encoding	The encoding type to use for the messages: Text, MTOM, or WCF Binary.
WS-Addressing version	The version of WS-Addressing for the selected encoding: None, WSA 1.0, or WSA 04/08.

Advanced Standards Tab

This tab lets you configure advanced WS- standards, such as Reliable Messaging and the Via address option.

The following table describes the UI elements:

UI Elements	Description
Reliable messaging	Enables reliable messaging for services that implement the **WS-ReliableMessaging** specification. The encoding type to use for the messages: Text, MTOM, or WCF Binary.
Reliable messaging ordered	Indicates whether the reliable session should be ordered.

UI Elements	Description
Reliable messaging version	The version to apply to the messages: WSReliableMessagingFebruary2005 or WSReliableMessaging11.
Specify via address	Sends a message to an intermediate service that submits it to the actual server. This may also apply when you send the message to a debugging proxy. This corresponds to the **WCF clientVia** behavior. This is useful to separate the physical address to which the message is actually sent, from the logical address for which the message is intended.
Via address	The logical address to which to send the message. It may be the physical of the final server or any name. It appears in the SOAP message as follows: <wsa:Action>http://myLogicalAddress<wsa:Action> The logical address is retrieved from the user interface. By default, it is the address specified in the WSDL. You can override this address using this field.

Security Tab

The Advanced security settings correspond to the **WS-Security** specifications.

WCF Service (WSHttpBinding) Scenarios

UI Element	Description
Enable secure session	Establish a security context using the WS-SecureConversation standard.
Negotiate service credentials	Allow WCF proprietary negotiations to negotiate the service's security.

WCF Service (CustomBinding) Scenarios

UI Element	Description
Default algorithm suite	The algorithm to use for symmetric/ asymmetric encryption. The drop-down list of algorithms, gets it values from the SecurityAlgorithmSuite configuration in WCF: **Default:** Basic256
Protection level	Indicates whether the SOAP Body be encrypted/ signed. THe possible values are: None, Sign, and Encrypt And Sign (default) **Default:** Encrypt And Sign
Message protection order	The order for signing and encrypting. Choose from: ➤ Sign Before Encrypt ➤ Sign Before Encrypt-And Encrypt Signature ➤ Encrypt Before Sign
Message security version	The WS-Security security version. You can also indicate whether to **Require derived keys** for the message.
Security header layout	The layout for the message header: Strict, Lax, Lax Timestamp First, or Lax Timestamp Last.
Key entropy mode	The entropy mode for the security key. The possible values are: Client Entropy, Security Entropy, and Combined Entropy. **Default:** Combined Entropy
Require security context cancellation	Indicates whether to require the cancellation of the security context. If you disable this option, stateful security tokens will be used in the **WS-SecureConversation** session, if they are enabled.
Include timestamp	Includes a timestamp in the header.

UI Element	Description
Allow serialized signing token on reply	Enables the reply to send a serialized signing token.
Require signature confirmation	Instructs the server to send a signature confirmation in the response.
X509 Inclusion Mode	When to include the X.509 certificate: ➤ Always to Recipient ➤ Never ➤ Once ➤ Always To Initiator **Note:** This and the next three options only apply when using an X.509 certificate.
X509 Reference Style	How to reference the certificate: ➤ Internal ➤ External
X509 require derived keys	Indicates whether X.509 certificates should require derived keys.
X509 key identifier clause type	The type of clause used to identify the X.509 key. ➤ Any ➤ Thumbprint ➤ Issuer Serial ➤ Subject Key Identifier ➤ Raw Data Key Identifier

HTTP and Proxy Tab

This tab lets you set the HTTP and Proxy information for your test.

HTTP (S) Transport

The following table describes the HTTP(S) Transport options:

UI Element	Description
Transfer mode	The transfer method for requests/responses. The possible values are Buffered, Streamed, Streamed Request, and Streamed Response.
Max response size (KB)	The maximum size of the response before being concatenated. **Default:** 65 KB
Allow cookies	Indicates whether to enable or disable cookies.
Keep-Alive enabled	Indicates whether to enable or disable keep-alive connections.
Authentication scheme	The HTTP authentication method: None, Digest, Negotiate, NTLM, Integrated Windows Authentication, Basic, or Anonymous.
Realm	The realm of the authentication scheme in the form of a URL.
Require client certificate	Indicates whether to require a certificate for SSL transport.
Use default web proxy	Indicates whether to use machine's default proxy settings.
Bypass proxy on local	Indicates whether to ignore the proxy when the service is on the local machine.
Proxy address	The URL of the proxy server.
Proxy authentication scheme	HTTP authentication method on Proxy: Digest, Negotiate, NTLM, Basic, or Anonymous.

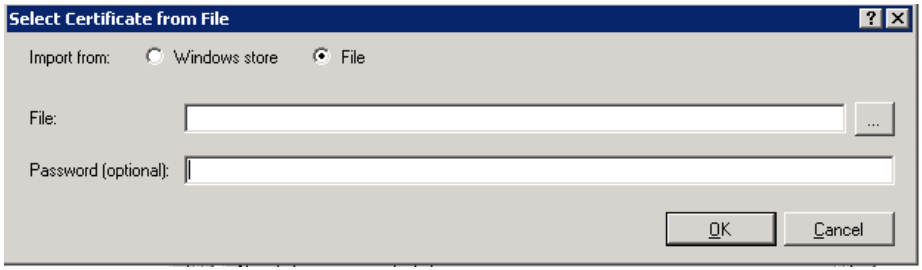
Select Certificate Dialog Box

This dialog box enables you to search and locate a certificate from a file or Windows store.

To access	Do the following: **1** Open the Security Settings for Port <Port_Name> Dialog Box. **2** Select a WCF Service scenario type from the **Service Details** list. **3** Click the **Browse** button adjacent to the **Server Certificate** box.
Relevant tasks	"Configure the security settings" on page 198.

Select Certificate from File

When you choose **Import from**: **File**, the dialog box shows the relevant user elements.

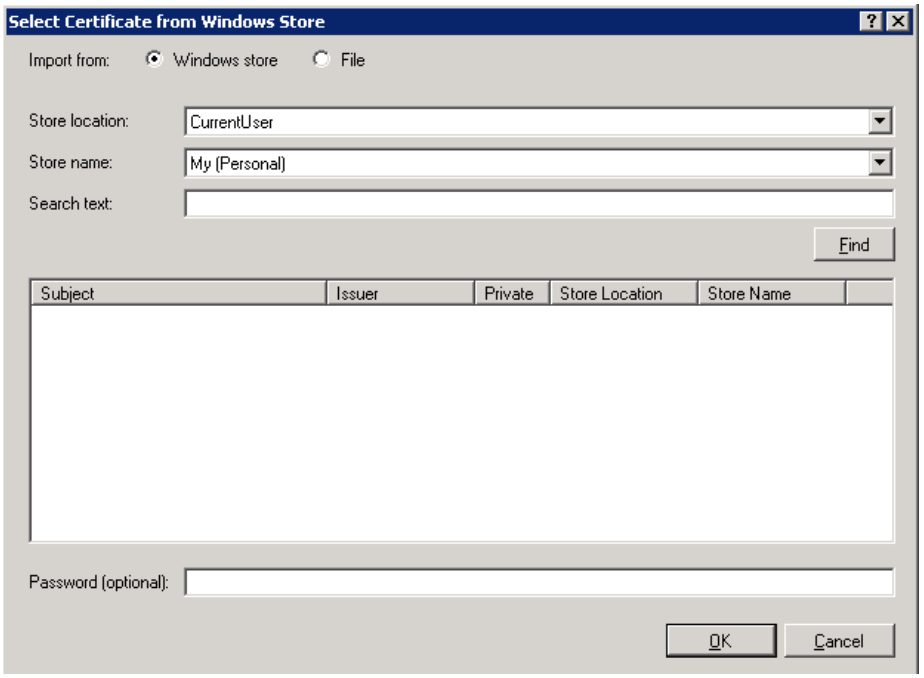


The user interface elements are described below.

UI Elements	Description
...	**Browse.** Allows you to locate the certificate file with a **.cer** or **.pfx** extension.
File	The complete path of the certificate file.
Password (optional)	The password required to access the certificate.

Select Certificate from Windows Store

When you choose **Import from**: **Windows Store**, the dialog box shows the Windows store related user elements:



The user interface elements are described below.

UI Elements	Description
Find	Begins the search for the certificate based on the Search text.
Store location	The store location, for example **Current User.**
Store name	The store name, for example, **AuthRoot**.
Search text	The text to match in the certificate name. If left blank, the **Find** action retrieves all available certificates.

UI Elements	Description
<certificate list>	A list of the certificates in the Windows store sorted by **Subject**, **Issuer**, **Private**, **Store Location**, and **Store Name**.
Password (optional)	The password required to access the certificate.

Troubleshooting and Limitations - Web Service Security

This section describes troubleshooting and limitations for working with Web services security.

- Authentication and proxy security are not supported for Web Services imported from a UDDI.
- When testing Web Services that require message-level security, the Web Service security scenario only supports SOAP version 1.1. For SOAP 1.2 use a WCF type scenario.
- When using a SAML security token for Web services security, user-provided content may contain creation and expiration timestamps. To extend the life of the test, we recommend that you hard-code an expiration date in the distant future. In this is not possible, change the timestamp by implementing the **OnBeforeApplyProtocolSettings** event.
- When using a SAML security token for Web services security, if you edit the values in Grid mode, they may not be updated in Service Test.
  Workaround: To update the values, switch to **Text** mode and save the test.
- Web Service steps are not supported when using a SAML token with a certificate from the file system.
  Workaround: Install the certificate to the Windows store and select the certificate from the store.
- The **AfterProcessRequestSecurity** and the **BeforeProcessResponseSecurity** events are not supported for Web Service activities configured with WCF type scenarios.
- The **OnSendRequest** and the **OnReceiveResponse** events are not supported for Web Service activities configured with WCF type scenarios.

6

Running Tests

This chapter includes:

Concepts

➤ Running Tests Overview on page 240

Tasks

➤ How to Run a Test on page 243

➤ How to Prepare and Run a Load Test on page 247

Reference

➤ Command Line Syntax on page 251

➤ Run Test User Interface on page 252

➤ Debug User Interface on page 255

Troubleshooting and Limitations - Load Testing on page 258

Concepts

Running Tests Overview

You can run tests directly from the Service Test interface or from the command line. The test run allows you to determine if the activities are functional and if they are performing as expected.

Before running the test, you can select a location in which to store the results, and set input parameter values specifically for this test run. For details, see "How to Run a Test" on page 243.

For details about the command line, see "Command Line Syntax" on page 251.

The Run Results Viewer provides a detailed report about each step and its checkpoints. For details, see Chapter 8, "Run Results Viewer."

When you run custom code from the user interface, you can also use breakpoints and other tools to debug your code.

Service Test allows you set the run configuration—**Debug** or **Release**. The **Release** mode is more efficient, and is recommended for Load Testing.

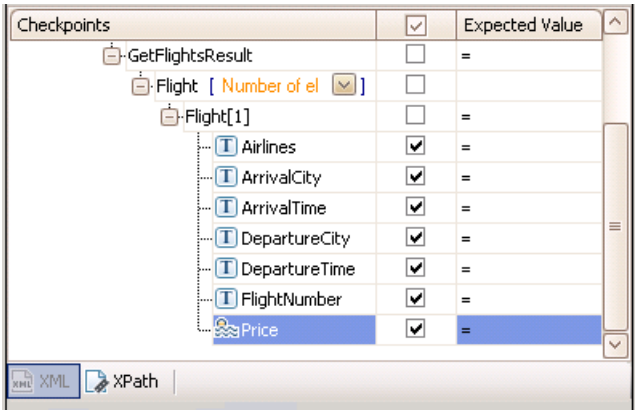
When a test step fails, Service Test continues to run the other test steps. The exception to this is if you enabled the **Stop test if checkpoint fails** option and the checkpoint failed. For details, see "Input/Checkpoints View" on page 55.

For Web Service steps, you can instruct Service Test to validate the SOAP response against a schema or WS-I standards. For details, see "Input/Checkpoints View" on page 55.

Checkpoint Validation

In functional testing, it is necessary to check the response from the server to confirm that your service performed the actions correctly. The response can contain several properties, each containing several data items. The **Checkpoints** section is a central point for defining the required response values for your test.

Before replaying a test, you set expected values for the test properties. You can enter the values manually, link them to a data source, or load values that were captured during replay. This is useful when you have many argument values—instead of manually entering values, you automatically load them.



Service Test automatically displays all of the step's properties with a check box. To include a checkpoint in the test, you select its check box. If you load replay values, you can select only those properties that you want to check.

The **XPath** tab lets you validate results against XPath expressions. For details, see the "Configure checkpoints" on page 243.

Service Test's Checkpoint setting lets you indicate whether or not to stop the test if the checkpoint fails. For details, see the "Input/Checkpoints View" on page 55.

After the test run, you can view the log or the run results and determine if the results were as expected. The Run Results Viewer displays the checkpoint status in its own sub-node. For details, see "HTTP Checkpoint Response" on page 325.

Load Testing

You can add load testing capabilities to your test, that will allow you to measure performance under load.

When you have a full LoadRunner installation on the Service Test machine, Service Test provides a custom template, **Service Test Enabled for Load Test**.

For tests created with the standard **Service Test** template, Service Test provides a conversion utility which converts the script into a LoadRunner compatible script, with a **.usr** extension.

Note: Tests created in Service Test and converted into LoadRunner scripts, cannot be edited in LoadRunner's Virtual User Generator (VuGen). To edit a test, modify it in Service Test.

To measure performance within LoadRunner, you can insert **Start** and **End Transaction** markers. During execution, LoadRunner measures the execution time of all the actions between the transaction markers.

The **Think Time** activity lets you emulate the way a true user operates, pausing between actions. You specify a duration in seconds, for which to wait between test steps.

When working with data tables, you can use the standard data retrieval methods that are available in LoadRunner. For details, see "Data Retrieval Options for Load Test Enabled Tests" on page 308.

For task details, see "How to Prepare and Run a Load Test" on page 247.

For more information, see the *HP LoadRunner Controller User Guide*.

Tasks

How to Run a Test

This task describes how to run a test through the user interface and view its results.

For information on running a test through the command line, see "Command Line Syntax" on page 251.

This task includes the following steps:

- "Configure checkpoints" on page 243
- "Select a build configuration - optional" on page 245
- "Save the test in ALM/QC - optional" on page 245
- "Add external references - optional" on page 245
- "Select a location for the run results - optional" on page 245
- "Set values for the test variables- optional" on page 246
- "Validate the structure of the SOAP response" on page 246
- "Run the test" on page 246
- "Debug the test - optional" on page 246
- "Results" on page 246

1 Configure checkpoints

Checkpoints let you validate the results. For details, see "Checkpoint Validation" on page 241.

a Open the **Input/Checkpoints** view. By default, Service Test displays the **XML** tab. Provide an expected value for each row in one of the following ways:

- Manually

- Link to Source dialog box.

➤ **Load from replay** button.

b Select a comparison operator, such as =, >, or Regex. These may differ depending on the data type.

c Select the **Validate** check boxes in the rows of the properties that you want to validate. Clear the check boxes for the properties you are not validating.

d To validate against an XPath expression:

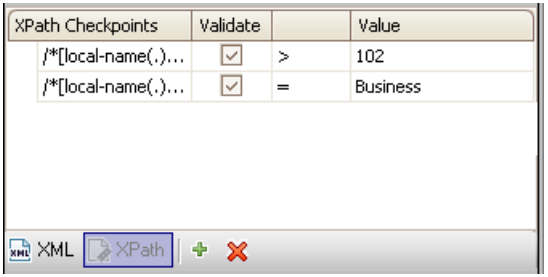
➤ Obtain an XPath expression for your value. To retrieve the XPath of a value already in the **Value** column, select **Copy XPath** from the shortcut menu in the **XML** tab.

➤ Click the **XPath** tab.

➤ Click the Add button to add a new XPath entry.

➤ Paste the expression into the **XPath Checkpoints** column.

➤ Select the check box in the **Validate** column, select a comparison operator, and provide a value.



e To halt the test run if the response does not return the expected value select **Stop test if checkpoint fails**.

Tip: You can customize and override the checkpoint settings using an event handler. For details, see "How to Write Checkpoint Events" on page 389.

For user interface details, see the "Input/Checkpoints View" on page 55.

2 Select a build configuration - optional

Choose a build configuration. Select **Build** > **Set Configuration** to select a configuration: **Debug** or **Release**. During development, it is preferable to use the **Debug** option to allow you access to detailed output information. To conserve resources, such as in Load Testing, use the **Release** option.

3 Save the test in ALM/QC - optional

To save results in ALM/QC, you need to first save the actual test in ALM/QC. Select **File** > **HP ALM/QC Connection**.

4 Add external references - optional

You can add one or more references to external DLL files. To add an external reference, select **Test** > **Add Reference**. Browse for the file and click **OK**.

5 Select a location for the run results - optional

a Click the Run button on the toolbar and select the **Results Location** tab.

To show and hide this dialog box, choose **Edit** > **Settings** > **Show Run Dialog Box.**

b For test saved on the File System, select a location in which to store the run results—**New run results folder** or **Temporary run results folder**. Choose the latter if you do not expect to use and analyze the results. For details, see "Results Location" on page 254.

c For tests saved test in ALM/QC, specify a Run name. If relevant, select a **Test set** and **instance**.

6 Set values for the test variables- optional

Select the test properties or user/system variables. For details, see "How to Define Test Properties or User/System Variables" on page 136.

7 Validate the structure of the SOAP response

For Web Service steps, you can instruct Service Test to validate the SOAP response against the schema or WS-I standards. For details, see the "Input/Checkpoints View" on page 55.

8 Run the test

Click the **OK** button in the Run dialog box.

9 Debug the test - optional

Use the Break mode and other debug tools to solve any runtime issues.

For user interface details, see the "Service Test Main Window" on page 38 or the "Debug Menu" on page 255.

10 Results

View the results in the Run Results Viewer. The viewer opens automatically after a test run.

- A green check mark indicates success.
- A red **X** mark indicates a failure in one of the steps. Expand the nodes of the report to view details about each step and checkpoint.

To view the report at a later stage, choose **Test** > **View Test Results**.

How to Prepare and Run a Load Test

This task describes how to prepare a test for load testing in LoadRunner. For an explanation about load testing, see "Load Testing" on page 242.

This task includes the following steps:

- "Prerequisite" on page 247
- "Create a load test" on page 247
- "Add test steps" on page 248
- "Prepare for load testing - optional" on page 248
- "Assign data to the test - optional" on page 249
- "Set the data retrieval properties - optional" on page 249
- "Set the run configuration to Release" on page 250
- "Run the test in Load Testing mode to validate the test" on page 250
- "Incorporate the test into LoadRunner" on page 250

1 Prerequisite

Make sure that HP LoadRunner or a standalone version of VuGen (HP Virtual User Generator) is installed. Without this installation, the Load Testing template will not be available.

2 Create a load test

Create a new test with the **Service Test Enabled for Load Testing** template.

If you have a test that was created with the standard Service Test template, select **Test** > **Enable Test for Load Testing** or click the **Enable Test for Load Testing** button.

3 Add test steps

Drag activities from the toolbox into the canvas to add steps to the test.

4 Prepare for load testing - optional

To measure the performance of a group of steps, define a transaction.

a Mark the beginning of a transaction.

- ➤ Drag the **Start Transaction** activity from the Toolbox's **Load Testing** category, into the canvas. Place it before the first step of the group of steps that you want to measure.
- ➤ Open the **Input/Checkpoints** view in the Property Sheet. Enter a **Transaction name**. This name will be used in LoadRunner Analysis.

b Mark the end of a transaction.

- ➤ Drag the **End Transaction** activity to the end of the group of steps you want to measure.
- ➤ In the Property Sheet's **Input/Checkpoints** view, type a transaction name. The name must be one that was already used for a prior **Start Transaction** step.
- ➤ In the End Transaction's **Input** properties, select a **Status** for reporting: PASS, FAIL, AUTO, or STOP.

Note: The End Transaction status is only the LoadRunner transaction's status—not the status of step in Service Test. For example, if you assign a **Failed** status to the transaction, Service Test can still issue a **Passed** status for the test step.

---

c Set the think time.

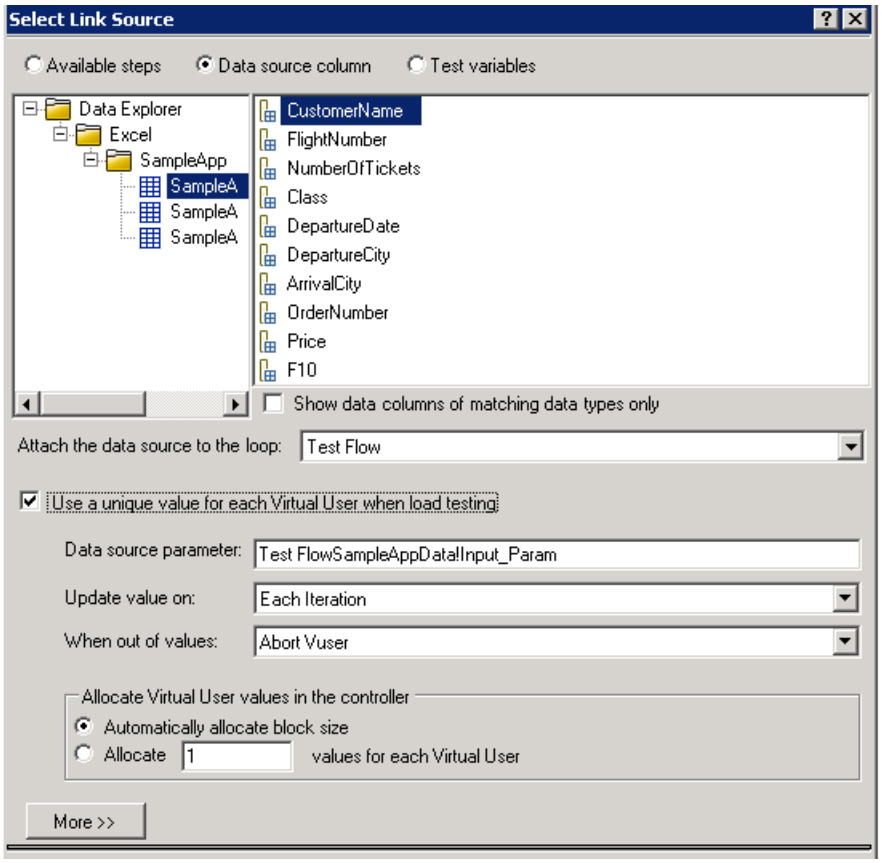
- ➤ If you want to emulate think time, drag the **Load Testing** > **Think Time** activity between the relevant steps.
- ➤ In the Property Sheet's **Input/Checkpoints** view, click in the **Duration (sec)** row and specify a think time in seconds.

5 Assign data to the test - optional

Import or create data tables for the input properties. For details, see "How to Assign Data to Test Steps" on page 283.

6 Set the data retrieval properties - optional

If you have data in the **Data Window** pane, set the data retrieval properties. Click the **Link to a data source** button. For details, see the "Data Retrieval Options for Load Test Enabled Tests" on page 308.

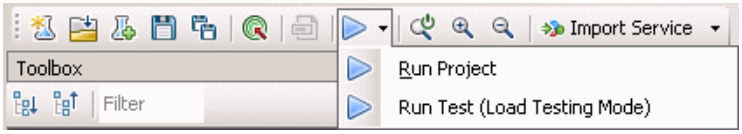


7 Set the run configuration to Release

Select **Build** > **Set Configuration** and select **Release**. The **Release** mode conserves resources, thus enhancing the load testing capabilities.

8 Run the test in Load Testing mode to validate the test

Expand the **Run** button and select **Run test (Load Testing mode)**. This run is only for debugging purposes, to verify that the test is functional.



Note: When you run a test in Load Testing mode, the **Output** tab does not contain data and the Run Results Viewer does not open. To view the results, select **Run Project** to run the test in functional mode.

---

9 Incorporate the test into LoadRunner

Add the test to the LoadRunner Controller console to include it in a load test. For more information, see the *HP LoadRunner Controller User Guide*.

Reference

Command Line Syntax

The Service Test executable, **ServiceTestExecuter.exe**, is located in the product's bin directory.

You can run **ServiceTestExecuter.exe** from the command line using the following parameters:

UI Elements	Description
-test	The full path of the test (required). Specify the test directory—not the solution directory.
-inParams	The full path of an XML file containing the input property values (optional).
-outParams	The full path of an XML file containing the output property values (optional).
-profile	The name of an Test profile (optional). For details, see "How to Define Test Properties or User/System Variables" on page 136.
-report	The folder in which to store the report.

Tip: To retrieve a list of all of available parameters, run **ServiceTestExecuter.exe** without any parameters.

The following example runs Test1 using input properties from inParams.xml and output properties from outParams.xml:

```
C:\Program Files\HP\HP Service Test\bin> ServiceTestExecuter.exe -test
"c:\MyTests\Test1\Test1" -inParams "c:\MyData\inParams.xml" -outParams
"c:\MyData\outParams.xml"
-profile Profile1
```

where the input property file, InParams.xml, has, for example, this structure:

```
<Arguments>
    <a>1</a>
     <b>2</b>
</Arguments>
```

Note: To run a test from the command line, you must save and run the test at least once.

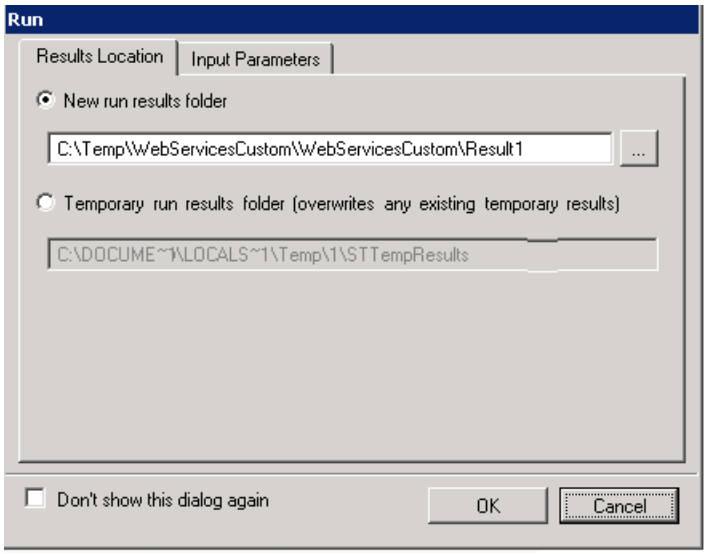
Run Test User Interface

This section includes:

➤ Run Dialog Box on page 253

Run Dialog Box

Enables you to view the properties of the application.



To access	Select **Debug** > **Run** or click .
Relevant tasks	"How to Run a Test" on page 243

The Run dialog box has the following tabs:

- "Results Location" on page 254
- "Input Parameters" on page 255

Results Location

This tab lets you specify a location for the Run Results. The fields differ for a test saved on the file system and one saved in an ALM/QC project. The user elements are described below.

GUI Element (A-Z)	Description
New run results in HP ALM/QC Project	Stores the run results in the current ALM/QC project. You cannot change the destination **Project name** (Available when there is a connection to an ALM/QC server). The following fields are editable: ➤ **Run name.** A custom name for the test run. ➤ **Test set.** The name of the test set in ALM/QC. ➤ **Instance.** The instance number of the run.
New run results folder	Stores the run results in a specific folder (Available when there is no connection to an ALM/QC server). Click the Browse button to select a location. The default path is **C:\Documents and Settings\%USERPROFILE%\My Documents\ServiceTestProjects\<Test Name Folder\<TestName>.**
Temporary run results folder	Stores the run results in a temporary folder, overwriting any existing results. This folder uses the **TEMP** variable defined for the current user, **%USERPROFILE%\Local Settings\Temp\STTempResults.**
Don't show this dialog again	Bypasses the Run dialog box in future test runs, using your last selection. **Tip**: To show this dialog box again, select **Edit** > **Settings** > **Show Run Dialog Box**.

Input Parameters

This tab lets you view and specify values for Input parameters that you defined earlier.

GUI Element (A-Z)	Description
Schema	The name of the parameter.
Value	An editable column containing the parameter values.

For details about defining user variables, see "Action Buttons - Test Variables View" on page 62.

Debug User Interface

This section includes:

- Debug Menu on page 255
- Debug Output Window on page 257

Debug Menu

Enables you to view the properties of the application.

To access	Expand the Service Test **Debug** menu.
Relevant tasks	"Debug the test - optional" on page 246

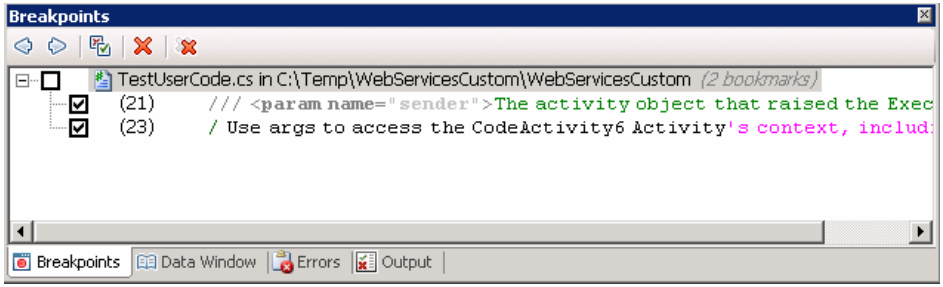
The following tabs are included:

UI Element	Description
	Run. Runs and compile the test.
	Stop Process. Stops the test execution.

UI Element	Description
	Break. Pauses the test run and enters Break mode in order to allow you to edit the code.
	Continue Debugging. Resumes the test run from the point of the break.
	Step Over. Executes the current statement and enters Break mode in order to allow you to edit the code. If the statement calls another procedure, Service Test will not step into that procedure. It will run the procedure, and advance to the next statement in the current procedure. **Step Into.** Executes the current statement and enters Break mode in order to allow you to edit the code. If the statement calls another procedure, Service Test will step into that procedure. **Step Out.** If you used **Step Into** and the control moved to a called procedure, Step Out automatically runs the current procedure, and returns to the procedure from which it was called.
	Step Into. Executes the current statement and enters Break mode in order to allow you to edit the code. If the statement calls another procedure, Service Test will step into that procedure.
	Step Out. If you used **Step Into** and the control moved to a called procedure, Step Out automatically runs the current procedure, and returns to the procedure from which it was called.
Toggle Breakpoint	Inserts or removes a breakpoint at the location of the cursor. **Note:** This applies when working in a code window, such as an event, but not in the canvas. **Tip:** The shortcut key for toggling a breakpoint is **F7**.

Debug Output Window

Enables you to view the run's debug information and breakpoints.



To access	Select **View** > **Debug** > **<sub-item>.**
Relevant tasks	"Debug the test - optional" on page 246

The following tabs are included:

UI Elements	Description
Breakpoints tab	A list of the breakpoints in the test, for each open buffer. This tab provides the following controls: ➤ . **Go to Previous Bookmark.** Jumps to previous bookmark in the list. ➤ . **Go to Next.** Jumps to next bookmark in the list. ➤ . **Enable/Disable.** Enable or disable all bookmarks. ➤ . **Delete**. Delete the selected bookmark. ➤ . **Delete All.** Delete all bookmarks in all buffers. **Tip:** To create a bookmark, click on a line within a buffer and press **F7**.
Callstacks tab	A list of the calls in the stack sorted by **Name** or **Language.**
Console tab	A console allowing you to manually enter debug commands.

UI Elements	Description
Loaded modules tab	A list of the currently loaded modules by **Name, Address**, **Path**, **Order**, and **Symbols.**
Local variables tab	A list of the local variables by **Name, Value**, and **Type.**
Threads tab	A list of the currently loaded modules by **ID**, **Name, Location**, **Priority**, and **Frozen.**
Watch tab	A list of watched expressions. The shortcut menu provides the following controls: ➤ **Add watch.** Allows you to add an expression to be watched. ➤ **Remove Watch.** Removes the selected watched item. ➤ **Remove all.** Removes all items from the watch list.

Troubleshooting and Limitations - Load Testing

This section describes troubleshooting and limitations for working with load testing.

Running Tests

➤ When you manually add a reference to an external DLL, Service Test prompts you to save it locally. To change your preference about a specific referenced file, remove the reference and add it again manually.

➤ The Validate Structure checkpoint fails if the expected value is a SOAP Fault and the Web Service call returns an **UnsupportedMediaType** status.

Load Testing

➤ Related data mapping in a load-enabled test is not supported for the LoadRunner parameter advance policy of **Each Occurrence**.

➤ Load Enabled tests that were compiled or run within VuGen, cannot be run in the LoadRunner Controller. Run the tests in Service Test only—do not replay them in VuGen.

- Tests created as Business Process Testing (BPT) components, cannot be used in load testing.
- If your tests use IBM's MQ client, make sure to install the MQ client on all machines running these tests.
- When running a test in Load Testing mode, the structural checkpoints, **Validate Structure** and **Validate WS-I**, are validated, even when the options are disabled in the user interface.

7

Data Handling

This chapter includes:

Concepts

- Data Handling Overview on page 262
- Defining Data on page 262
- Unique Data on page 271
- Data Relations and Navigation on page 272

Tasks

- How to Create Data Sources on page 276
- How to Assign Data to Test Steps on page 283
- How to Access Data Through Event Handler Coding on page 287
- How to Set the Navigation Properties on page 290
- How to Data Drive a Test Step on page 291

Reference

- Data Handling User Interface on page 293
- Database Data Source User Interface on page 313

Troubleshooting and Limitations - Data on page 316

Concepts

Data Handling Overview

Before running a test, you may assign input and checkpoint data and specify how the data should be used.

Service Test's Data window lets you view and maintain data for your test steps. You can import data from Excel files, create local data tables, or define XML files as sources for your data. For details, see "How to Assign Data to Test Steps" on page 283.

You can data-drive the properties of a step or the SOAP request/response body. Data-driving is the mechanism by which Service Test automatically creates data expressions that refer to data in a new, editable table. For details, see the "Data Driving Dialog Box" on page 296.

Defining Data

Service Test allows you to assign data to your steps in several ways. You can enter values manually, link to existing data, or data drive the values.

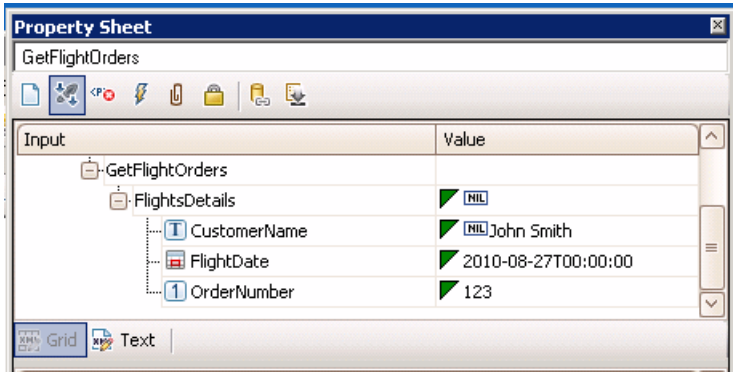
This section also includes:

- "Populating Data Tables Manually" on page 263
- "Linking to a Data Source" on page 264
- "Retrieving Data from a Database" on page 266
- "Outgoing Links" on page 268
- "Data Driving" on page 269

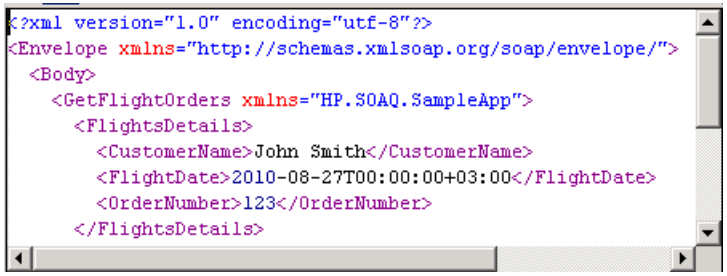
Note: For machines with Excel, the data tables follow standard Excel behavior—when you delete data from a table, the row remains without data. To remove a data row, select **Delete** from the row's shortcut menu. Otherwise, Service Test will run an iteration for that row, with empty values.

Populating Data Tables Manually

The Input/Checkpoints View grid allows you to view and edit the values of each input property. You can manually insert values into the grid for both simple properties and arrays.



You can also edit the values in the XML text mode. The editor also allows you to discard changes and return to the original XML using the **Revert** button.

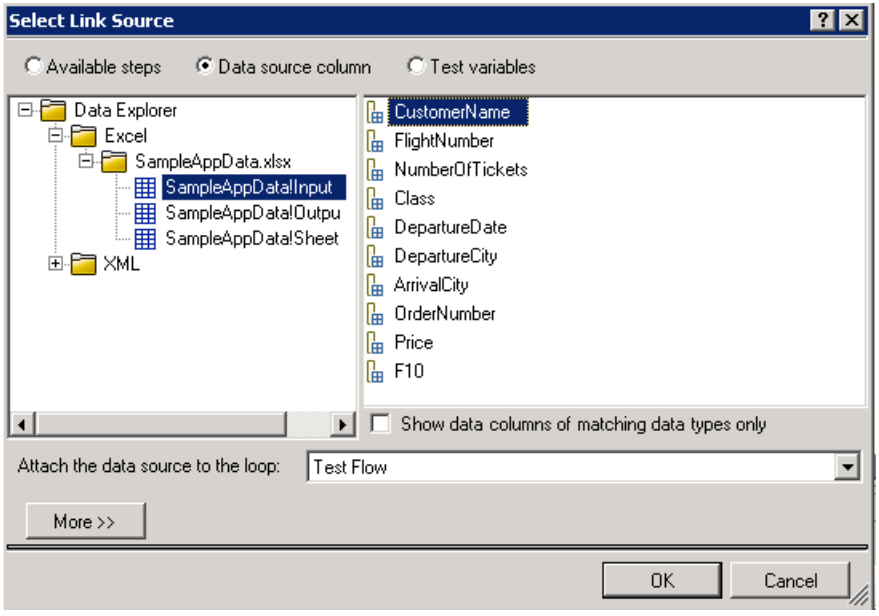


You can also populate the checkpoint values manually. In addition, after running the test at least once, you can load the replay values directly into the checkpoint using the **Load from Replay** button.

Linking to a Data Source

The recommended way to define data for a step is by linking it to a data source. The Select Link Source dialog box allows you to link to data in several ways:

- Specify a constant value
- Reference the output of a another step
- Use data from a data source, such as an Excel file or database table
- Use a test variable



You can also use a combination of these types: for example, **Data source column** and **Test variable** expressions combined to make a single expression.

To use data source columns, you import an existing data file, such as an Excel spreadsheet or XML file. Alternatively, you can use a wizard to create a custom local table. For details, see the relevant section:

- "Add an Excel data source" on page 276
- "Add a local data table" on page 277
- "Add an XML data source" on page 278.
- "Add a Database data source" on page 279

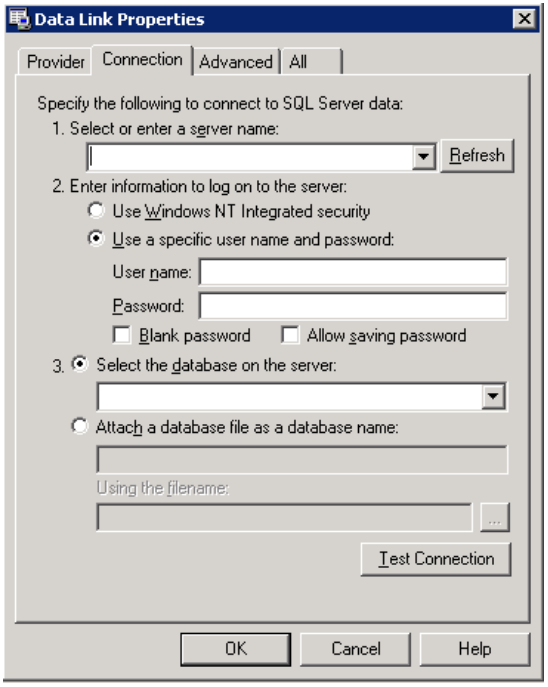
When preparing scripts for load testing, the Select Link Source dialog box also lets you select the way in which to retrieve the data and what to do when the data has run out. For details, see "Data Retrieval Options for Load Test Enabled Tests" on page 308.

For task details, see "How to Assign Data to Test Steps" on page 283.

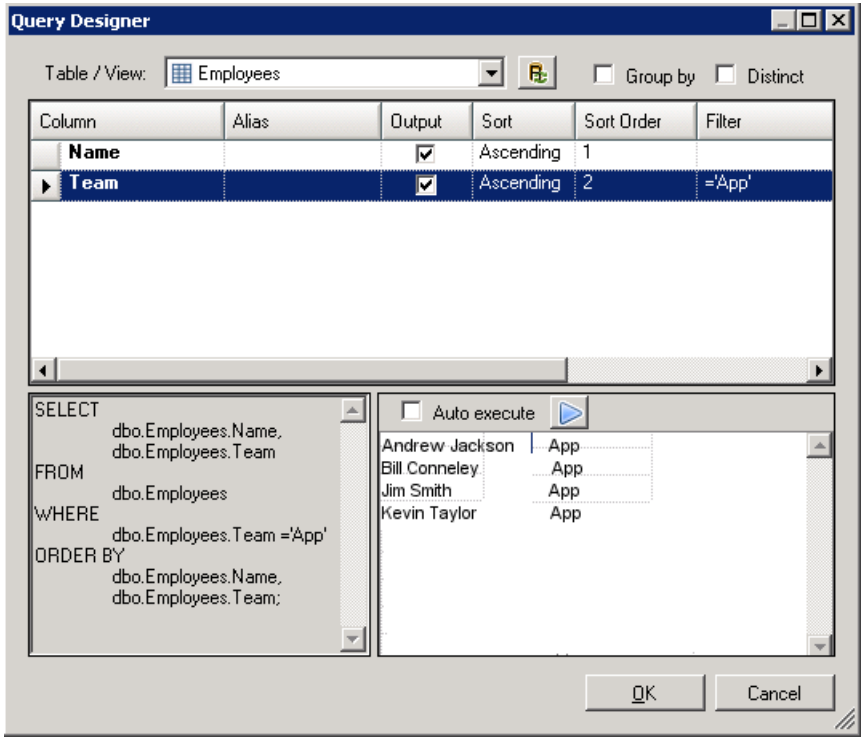
Retrieving Data from a Database

Service Test lets you retrieve data from a standard database. A wizard guides you through connecting to a database and creating an SQL query statement. After you complete the wizard, the Data Window tab displays the database's data tables.

You provide a connection string or create one through Microsoft's Link Data Properties dialog box.



Once you are connected to the database, you can use Service Test's Query Designer to create a custom SQL statement, based on the tables and views in your database.



For details on how to use the Query Designer, see the "Query Designer Dialog Box" on page 120.

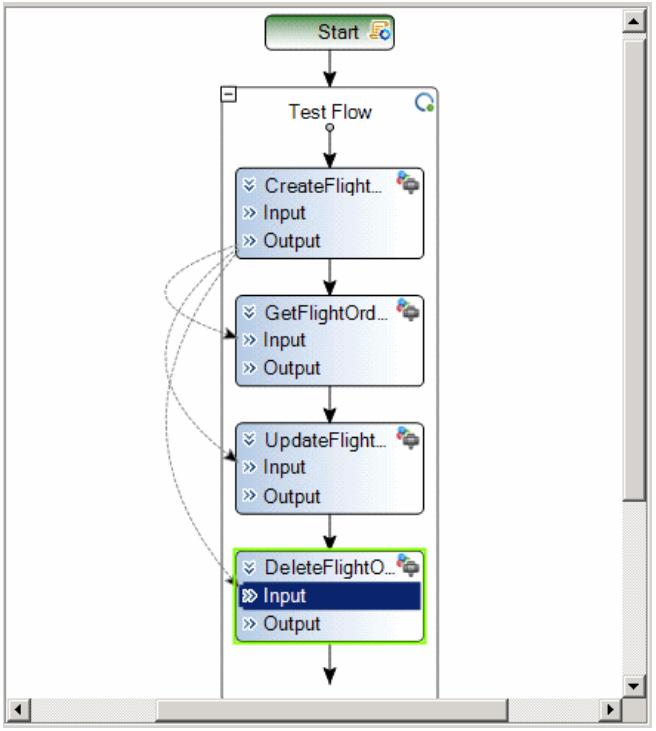
To learn more about using the wizard, see "Add a Database data source" on page 279.

Outgoing Links

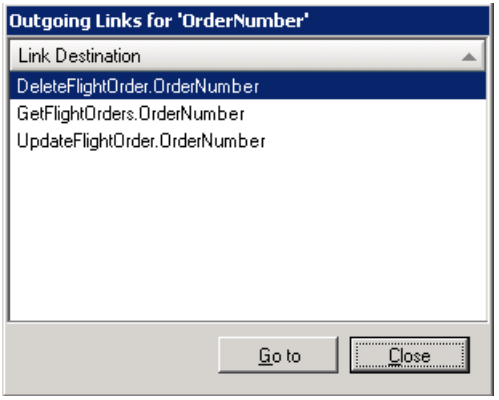
When you link a property to another step's output property, Service Test indicates this with a **Outgoing Links** button in the Property Sheet's **Input/Checkpoints** view.

By clicking on the button, you can open a list of the properties that link to this output property. This is especially helpful when linking to an output property multiple times.

The following example shows three steps linking to a single output property of the CreateFlightOrder step.

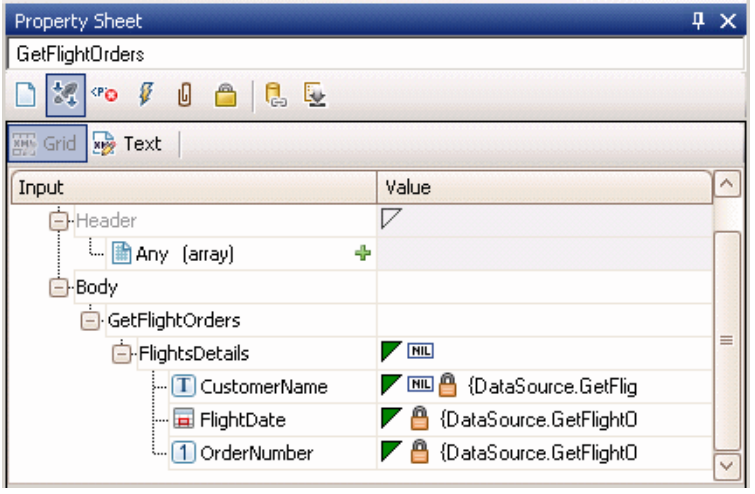


When you click the **Outgoing Links** button, Service Test opens a list of the target properties that link to this value. Using the **Go to** button, you can navigate directly to the linked property.



Data Driving

Data-driving is the mechanism by which Service Test automatically creates data expressions for properties. When you data drive a step, Service Test adds data expressions to the step's value fields.

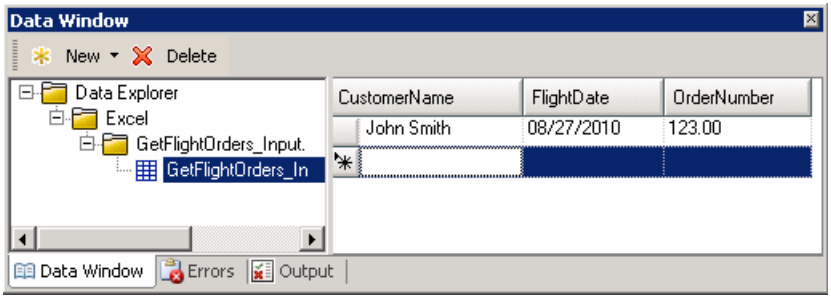


These expressions refer to data in a new editable Excel table or XML structure, depending on the type of provider you specified during the data driving.

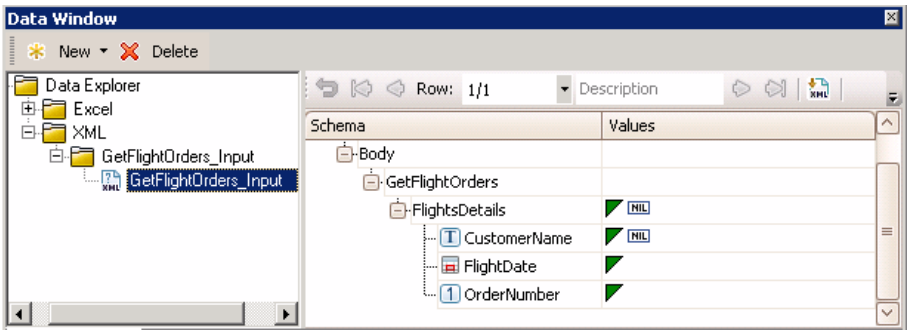
You can customize data-driving in the following ways:

- Specify the type of data source to create—Excel or XML
- Indicate which properties to data drive:
  - For standard input and output (checkpoint) properties: **Input**, **Checkpoints**, or both.
  - For XSD files, loaded for the HTTP or SOAP Request steps: **Request Body**, **Response Body**, or both.
  - For additional types such as HTTP multipart and SOAP Fault properties, see the "Data Driving Dialog Box" on page 296.
- Set the loop type. This overrides the regular Input Properties - Loop property.

For Excel type tables, you can manually add to and edit the data in the Data Window.



For XML data sources, you can manually edit the XML root.



For task details, see "How to Data Drive a Test Step" on page 291.

For user interface details, see "Data Driving Dialog Box" on page 296.

Unique Data

When you add data tables in the Data Window, this does not automatically associate the data with a step property.

You associate a step's properties with data in the following ways:

- Use the Link to Source dialog box. For details see "How to Assign Data to Test Steps" on page 283.
- Data Driving. For details, see "How to Data Drive a Test Step" on page 291.

Service Test lets you add data sources to your Test Flow or loop, without attaching it directly to a specific step property. This is especially useful if you want to use event handler coding to access your data.

When you define a new data source, Service Test assigns it a unique ID, called the **Query ID**. Each **Query ID** indicates a separate instance of the data. Two data sources may use the same data from the same table, but since they have different **Query IDs**, they do no affect one another.

For details, see "Add data as a data source" on page 282.

Data Relations and Navigation

Service Test lets you indicate how to use the data source, how many times to loop through the data.

You can also define relations between data tables and data sources.

This section also includes:

➤ "Navigating within the Data" on page 272

➤ "Child Relations" on page 274

➤ "Data Keywords" on page 274

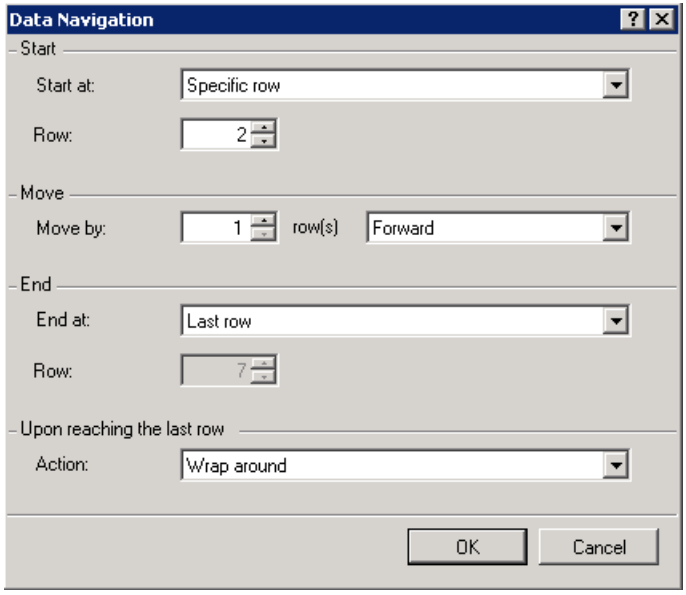
Navigating within the Data

Using the Data Navigation dialog box, you control the way you use the values in the data tables.

The Data Navigation settings work together with the loop settings described in "Input Properties - Loop" on page 95.

If the loop is configured as a **ForEach** type, and you designate a specific data source as the loop's collection, then the Data Navigation settings affect the number of iterations of the loop.

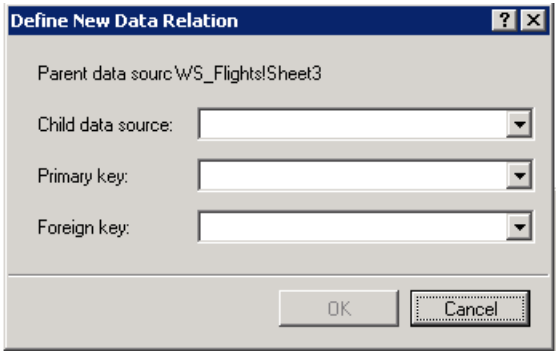
For data sources that are not designated as the loop collection, but whose values are fetched by steps within the loop, the Data Navigation policy affects the data differently. It indicates the order in which the values are fetched from the data source, and assigned to steps within the loop.



For user interface details, see the "Data Navigation Dialog Box" on page 309.

Child Relations

You can define data relations for all of your data tables. You can then use this relation as data for test properties. This is useful in cases where some property values are in one table, and other property values are in another table.



For task details, see "Create a new child relation" on page 282.

For user interface details, see the "Define/Edit New Data Relation Dialog Box" on page 312.

Data Keywords

You can use keywords to customize the test run and validation. To use a keyword, type it into the **Value** column or table row. Enclose the keyword with the # character. The keyword names are not case-sensitive.

Service Test provides keywords for both Input properties and checkpoints.

Input Keywords

For input properties, you can enter a keyword into one or more data provider sources, such as a data table or XML code.

Keyword	Description
#SKIP#	Omits this element from the XML of the request.
#NIL#	Adds a **nil=true** attribute to the property's XML in the request. **Example:** <name John Doe nil="true"></name> If the XML element is not nillable, Service Test reports this to the log.

Checkpoint Keywords

You can use keywords for checkpoints, by entering the keyword into the data table or XML associated with the checkpoint. Service Test provides the following keywords for checkpoints.

Keyword	Description
#SKIP#	The checkpoint with the #SKIP# value will not be validated.
#NIL#	During the test run, Service Test checks that the element has a **nil="true"** attribute in its response.
#NOT_FOUND#	Indicates that the element should not be found in the XML of the response. This setting ignores the comparison operator—it only checks for the absence of a value.
#EXISTS#	Indicates that the element should be found in the XML of the response. This setting ignores the comparison operator—it only checks for the presence of a value.

Tasks

How to Create Data Sources

This task describes how to define data sources before selecting a test step. After creating a test step, you link it to an existing data source. The following steps provide you with four options for creating a data source. To create a data source for your properties, you only need to add data using one of these options.

This task includes the following steps.

- "Add an Excel data source" on page 276
- "Add a local data table" on page 277
- "Add an XML data source" on page 278
- "Add a Database data source" on page 279
- "Create a new child relation" on page 282
- "Add data as a data source" on page 282

Add an Excel data source

1. Make sure the Data Window is visible. If not, select **View** > **Data Window**.
2. In the Data Window pane, select **New** > **Excel File**.
3. In the Add New Excel File Data Source dialog box, Browse to the file and indicate if the first row is a header row.
4. Specify a data source name.
5. Select an import mode: **Referenced** or **Embedded**. The Data Explorer shows the table in the right pane.
6. Click **OK**.
7. In the Data Explorer tree, select the data source. In the right pane, use the keyboard arrows to navigate within the table and set values.

For user interface details, see the "Add New Excel File Data Source Dialog Box" on page 299.

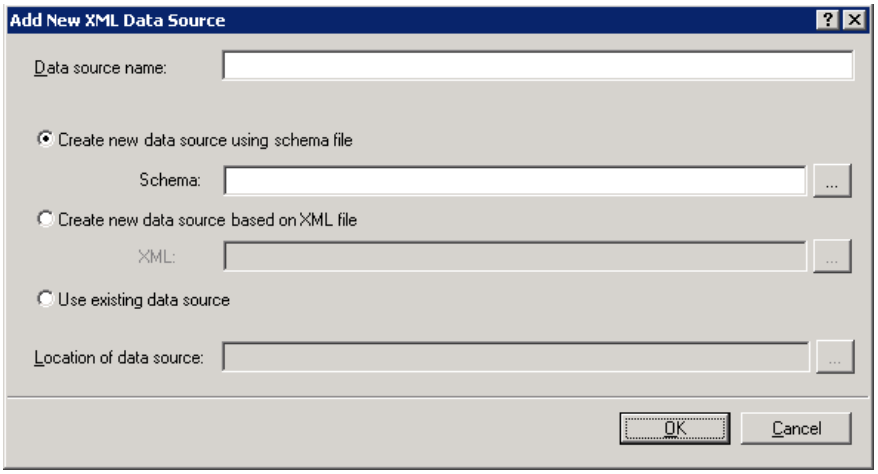
Add a local data table

1. Show the Data Window. Select **View** > **Data Window**.
2. In the Data Window pane, select **New** > **Local Table**. The **Add New Local Table Data Source Wizard** opens.
3. Click **Add** to create a column for the data table.
4. In the Add Columns dialog box, specify a column name and description. Select a data type from the drop-down list. Click **OK**.
5. Repeat steps **3** and **4** for each column you want to create.
6. Use the arrows to reorder the columns as required.
7. Click **Next** to specify a name for the data source. Click **Finish**.
8. In the Data Explorer, select the table's node and move within the table using the keyboard arrows. Click the grid cells to manually set values.

Add an XML data source

1 Show the Data Window. Select **View** > **Data Window**.

2 In the Data Window pane, select **New** > **XML**. The Add New XML Data Source dialog box opens. For user interface details, see the "Add New XML Data Source Dialog Box" on page 302.



3 Provide a **Data source name**.

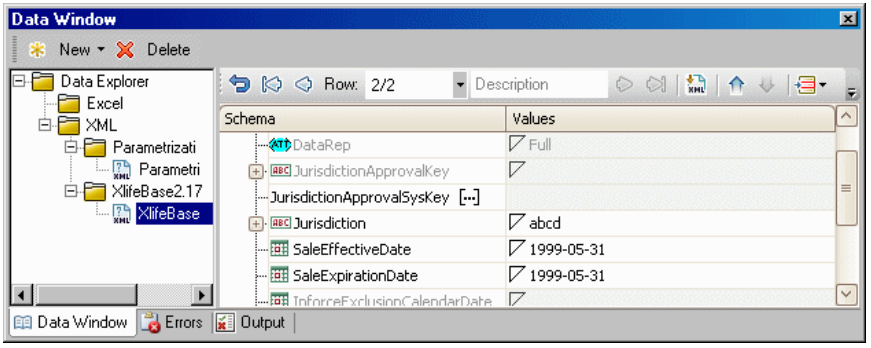
4 Select a source or schema option:

- **Create new data source using schema file.** Browse to the **Schema File Path**.
- **Create new data source based on XML file.** Browse to the **XML File Path** representing the structure of the data. You can add data at a later time.
- **Use existing data source.** Browse to the **Location of data source**. Each data source is stored as a subfolder of the test's **DataQueries** folder. Make sure to use the data source name as it appears in the folder, without the suffix. For example, for the file **MyXSD_0.xsd** file, enter *MyXSD* in the **Data source name** field.

Tip: Choose **Test** > **Open Script Directory** to access the test folder.

5 Click **OK**.

6 Expand the data source's node in the Data Window.



7 Edit the XML data in the grid. Add rows as described in the "XML Data Source - Toolbar" on page 294.

8 To load XML values, click the **Load data from an XML file** button.

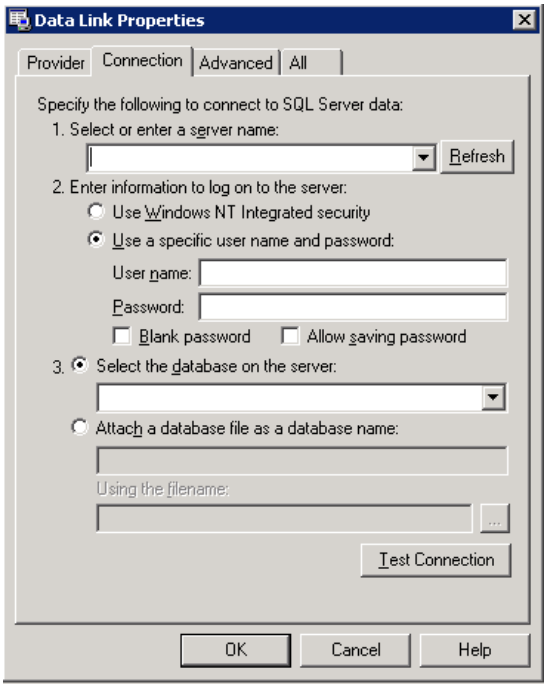
Add a Database data source

1 Show the Data Window. Select **View** > **Data Window**.

2 In the Data Window pane, select **New** > **Database**. The **Add New Database Data Source Wizard** opens. For user interface details, see the "Add New Database Data Source Wizard" on page 313.

3 In the **Define the connection string** screen:

a Paste an OLE DB connection string into the text area, or click the **Browse** button to use Microsoft's Data Link Properties dialog box. For details, click the **Help** button in the Data Link Properties dialog box.



b Click **Check Connection** to verify that the connection string is valid.

c Click **Next**. This button will be available only for valid connections.

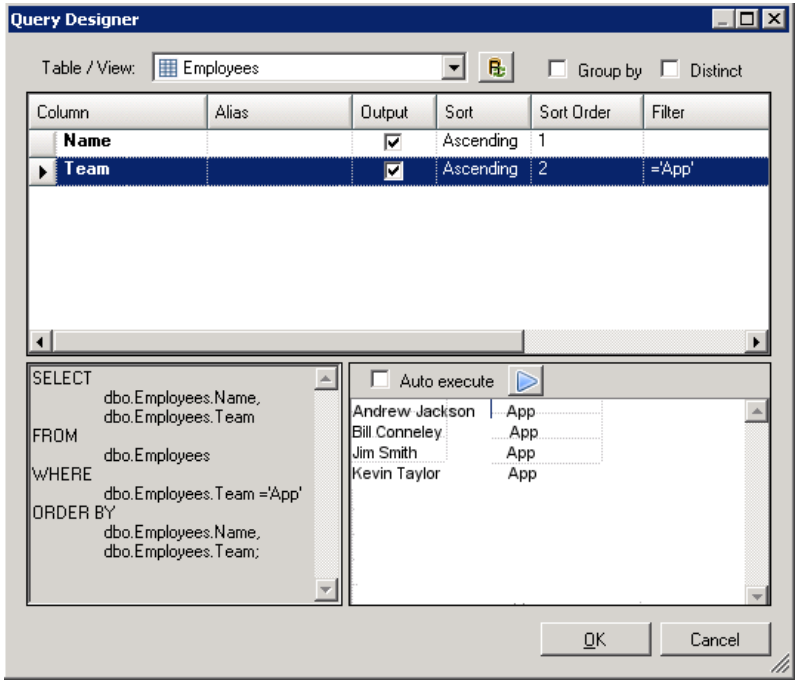
4 In the **Define the query statement** page:

a Provide a unique name for the data source, not used by another data source.

b Create an SQL statement. You can paste an existing string into the text area or click the **Browse** button to open the **Query Designer**. For details, see the "Query Designer Dialog Box" on page 120.

c If you are using the **Query Designer**:

- Select a table or view. The **Query Designer** lists all of the rows in the pane below and displays the corresponding SQL statement in the preview pane.
- Enable **Auto execute** to view the results of your query as you make changes in the upper pane.
- By default, the **Query Designer** selects all columns in the table. To remove a column from the query, clear the **Output** check box for that column.
- To set a sorting order, select Ascend or Descend from the **Sort** column drop-down. If you have multiple rows that you are sorting, provide the **Sort Order** beginning with 1. The **Query Designer** adds an ORDER BY clause to the preview pane.
- To filter the returned results, enter the text to match in the **Filter** column.
- To add a GROUP BY or DISTINCT clause to your statements select the appropriate check box.



➤ Click **OK** to accept the query and return to the wizard.

d In the wizard's **Define the query statement** page, click the **Check SQL Statement** button. A **Query Preview** dialog box opens. Click **Close** to return to the wizard.

e Click **Finish**. Service Test shows the query details in the Property Sheet and the data in the **Data Window** tab. For details, see the "Database Data Source Properties View" on page 69.

f To update the data at a later stage, click **Refresh** in the **Data Window** tab. For details, see the "Data Window Tab" on page 294.

Create a new child relation

This step lets you add child relations using primary and foreign keys. This applies to **Excel**, **Local Table**, and **Database** type data sources.

1 Add at least two Excel files or one file with two worksheets, containing the foreign and primary keys.

2 In the Data Window, select the parent table in the Data Explorer tree.

3 Click in the Property Sheet to open the **Data Source Properties** view.

4 Click the **Add** button. The Define New Data Relation dialog box opens.

5 Specify the details of the new relation. You can specify a relation to a different data source type, provided that is either an **Excel**, **Local Table**, or **Database** type data source. For user interface details, see the "Define/Edit New Data Relation Dialog Box" on page 312.

6 To edit a data relation, double-click the entry or click **Edit**.

7 To delete a data relation, select the entry and click **Remove**.

Add data as a data source

This step lets you define data as a data source, even without linking to a property. This will allow you to access the data source easily and use it in event handler coding. For details, see "Unique Data" on page 271

1 In the Data Window, select the parent table in the Data Explorer tree.

2 In the canvas, select the Test Flow or loop frame.

3 Click in the Property Sheet to open the **Data Sources** view.

4 Click **Add**. The Attach Data Source to Loop dialog box opens.

5 Select a data source from the list and click **OK**.

6 To set a navigation policy, double-click the data source or click **Edit** in the Property Sheet.

Tip: The data source is available only to steps within the current loop. To allow access to the data source from all steps, add it to the parent loop, such as Test Flow.

How to Assign Data to Test Steps

This task describes how to link the test step properties to data sources through the following steps:

- "Open the Select Link Source dialog box" on page 283
- "Select a linking option" on page 283
- "Assign array data from a data source - no leaf nodes" on page 285
- "Assign data from an array output" on page 285

1 Open the Select Link Source dialog box

a Display the step's properties. Select **View** > **Property Sheet** and open the **Input/Checkpoints** view.

b Click the **Link Source** button in the right corner of the row.

2 Select a linking option

a In the Select Link Source dialog box, select a data source option:

- **Available steps**. Select a step in the left pane and its property in the right pane.

- **Data source column.** For properties with simple data structures or with both leaf nodes and arrays—in the left pane select a data source. In the right pane, double-click on the relevant node. For properties that only have arrays, see step 3 on page 285.
- **Test variables**. Select a user or system variable. For information on defining user variables and profiles, see "Define user variables" on page 136.

b To use a combination of several data sources for one parameter, click **More** to expand the Select Link Source dialog box. Select the data sources one-by-one and click **Add**. If necessary, edit the expression manually.

c Click **OK** to complete the linking of the data. Service Test copies the values or expressions into the **Value** column of the grid.

Note:

- Linking data sources to properties only applies to steps within a loop.
- When linking data sources to an array with multiple elements. you only link the data to the first array element. Service Test will apply the data to the next elements according to the availability of the data. For example, if the Excel table contains three rows of data for the current parent row, it will populate three array elements.
- Linking a leaf node to a complex XML node is only supported for string type data.

For user interface details, see the "Select Link Source Dialog Box" on page 305.

3 Assign array data from a data source - no leaf nodes

Follow these steps if the property only contains an array, and there are there are no leaf nodes outside the array.

a Create a data relation (if required) as described in "Create a new child relation" on page 282.

b Click within the frame of Test Flow, or the relevant loop, and open the **Data Sources** view in the Property Sheet.

c Click the **Add** button. The Attach Data Source to Loop dialog box opens.

d Select the parent data source from the list. Click **OK**.

e To edit the data navigation properties, click **Edit**. Set the preferences and click **OK**.

f Select the step in the canvas. In the Property Sheet's **Input/Parameter's** view, click within the row of the array element you want to assign.

g Click the **Link Source** button in the right corner of the row. Select the **Data source column** option and select the desired node.

4 Assign data from an array output

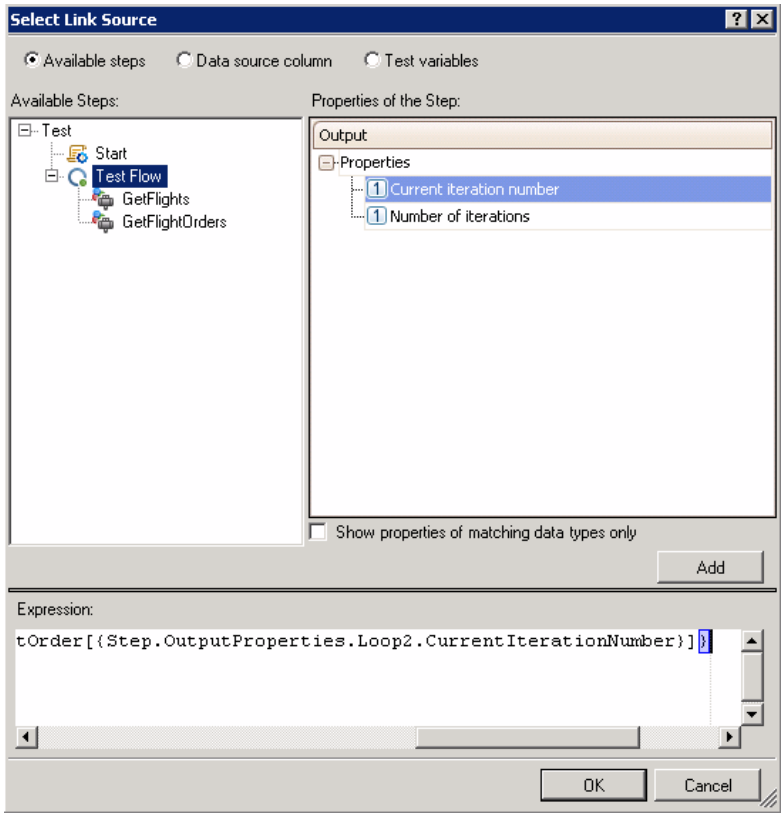
Follow these steps if you want to assign data to a simple variable from an output array element.

a Open the Select Link Source dialog box for the simple property. Click **More** to expand the dialog box.

b Select the **Available steps** option. In the left pane, select the step whose output value you want to use. In right pane, select an array element.

c Click **Add** to transfer the data source string to the **Expression** area.

d To set the index of the array to a variable, select the index number, in the **Expression** area, for example, 1, and select a variable from the upper pane. Click **Add**. The following example uses the **Current iteration number** from the **Test Flow** node, as a variable for the array's index.



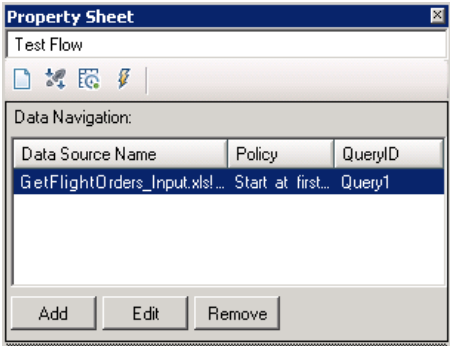
How to Access Data Through Event Handler Coding

This task describes how to access data sources through event handler coding through the following steps:

- "Prerequisite" on page 287
- "Open an event handler window for the test step" on page 288
- "Edit the code" on page 288
- "Example - retrieving a specific row" on page 288
- "Example - retrieving the current row" on page 288
- "Example - setting input properties for a Web Service call" on page 289

1 Prerequisite

Make sure you defined at least one data source for the loop containing the step. Determine your data source's Query ID, for example Query1, Query2 and so forth. For details, see "Add data as a data source" on page 282.



2 Open an event handler window for the test step

Select a test step and open the **Events** view in the Property Sheet. Double-click an event to open the **TestUserCode.cs** tab. For details, see Appendix A, "Coding Service Test Events."

3 Edit the code

Locate the **Todo** section. Use the Query ID for obtaining a value from a data cell. Use the following syntax:

```
this.<activity_instance>.<property_name>=this.<QueryID>.
                                GetValue(row_number>"<col_name>").ToString();
```

Note: Make sure to place the code in the relevant event handler. For example, code that sets input properties should be placed in the **BeforeExecuteStepEvent** handler, while code that sets output properties should be placed in the **AfterExecuteStepEvent** handler.

---

4 Example - retrieving a specific row

The following example using the **Concatenate Strings** activity, retrieves the **Prefix** and **Suffix** properties from the data sources with a Query ID of Query1, using the first and third rows. The **ToString** function converts the text to a string.

```
this.ConcatenateStringsActivity5.Prefix=this.Query1.GetValue(1,"Field1").ToString();
this.ConcatenateStringsActivity5.Suffix=this.Query1.GetValue(3,"Feld2").ToString();
```

This code overrides the navigation policy, since you specifically instructed Service Test to take the get data from a specific row.

5 Example - retrieving the current row

To retrieve values from the current row, using the active data navigation policy, leave out the row number.

The following example using the **Add** activity, retrieves the **A** and **B** properties from the data source with an ID of Query2, using the current row. Since the values are numbers, there is no need to convert them to a string.

```
this.AddActivity4.A=this.Query2.GetValue("Field1");
this.AddActivity4.B=this.Query2.GetValue("Field2");
```

6 Example - setting input properties for a Web Service call

Web Service calls store their property information in separate XML documents. Therefore, when working with Web Service calls, you cannot access the input or output properties directly using auto-complete. Instead, you need to parse the XML using an XPath expression.

To retrieve a property's XPath expression, click on the property in the Property Sheet and select **Copy XPath** from its shortcut menu. The **InputEnvelope** object contains all of the input properties of the Web service call. The **OutputEnvelope** object contains all of the Web service's output properties.

The following example using the sample application's **GetFlights** activity, sets the DepartureCity property to Denver. The XPath is copied from the Property Sheet.

```
XmlNode DepartureCity =
this.StServiceCallActivity5.InputEnvelope.SelectSingleNode("/
*[local-name(.)='Envelope'][1]/*[local-name(.)='Body'][1]/
*[local-name(.)='GetFlights'][1]/*[local-name(.)='DepartureCity'][1]");
if (DepartureCity!=null)
{
        DepartureCity.InnerText="Denver";
}
```

How to Set the Navigation Properties

This task describes how to set the navigation preferences for data stored in tables. You can set different navigation properties for each data source. For details, see "Navigating within the Data" on page 272.

This task includes the following steps:

- "Open the Data Sources view" on page 290
- "Select a data source" on page 290
- "Open the Data Navigation dialog box" on page 290
- "Set the navigation properties" on page 290

1 Open the Data Sources view

In the canvas, click within the loop frame, but not within a test step. In the Property Sheet, click the **Data Sources** view button (visible only for steps within a loop).

2 Select a data source

Click **Add** to select a data source. The Attach Data Source to Loop dialog box opens. Double-click a data source.

3 Open the Data Navigation dialog box

in the **Data Sources** view, select the data source with your data. Click **Edit** to open the Data Navigation dialog box.

4 Set the navigation properties

a In the Data Navigation dialog box, specify the **Start** and **End** rows.

b Indicate the direction in which to move when retrieving data from the table: **Forward** or **Backward**, and the number of rows by which to advance for each iteration.

c Specify the action to do when reaching the last row- **Wrap around** or **Keep using that row**. Click **OK**.

For user interface details, see the "Data Navigation Dialog Box" on page 309.

How to Data Drive a Test Step

This task describes how to data drive a test step. Data driving is Service Test's mechanism for automatically creating data expressions for step properties.

For further details, see "Data Driving" on page 269.

This task includes the following steps:

- "Prerequisites" on page 291
- "Open the Input Properties view" on page 291
- "Include the data" on page 291
- "Data drive the test step" on page 291
- "Enter values" on page 292

1 Prerequisites

Create a step in the canvas. For details, see "Create the test flow" on page 33.

2 Open the Input Properties view

Display the step's properties grid. Select **View** > **Property Sheet** and open the **Input/Checkpoints** view.

3 Include the data

To include optional properties in the data driving, toggle the triangle icon adjacent to the property. A filled-in triangle indicates an included property.

4 Data drive the test step

a Click the **Data Drive** button. The Data Driving dialog box opens.

b Choose a Data provider: **Excel** or **XML.**

c Select the properties upon which you want to apply data driving: input properties, checkpoints, or both.

d Indicate whether you want to **Configure 'Test Flow' as a For Each loop using the new data source**. This option repeats the test steps according to the number of data rows in the Excel or XML data source. If you disable this option, you can manually set the number of iterations through the loop properties. This setting overrides the general loop properties. For details, see "Input Properties - Loop" on page 95.

e Click **OK**.

f Service Test prompts you to confirm the action. Click **OK**.

Service Test populates the value column with data expressions. The expressions are preceded by informational icons, indicating read-only status or unmatching data types.

Note: If your checkpoints are within an array, you must create at least one array element before data driving. To add an array element, select the array's parent node and click the **Add** button.

For user interface details, see the "Data Driving Dialog Box" on page 296.

5 Enter values

In the Data Window (**View** > **Data Window**), edit the columns of the newly created data table(s) or XML source.

Reference

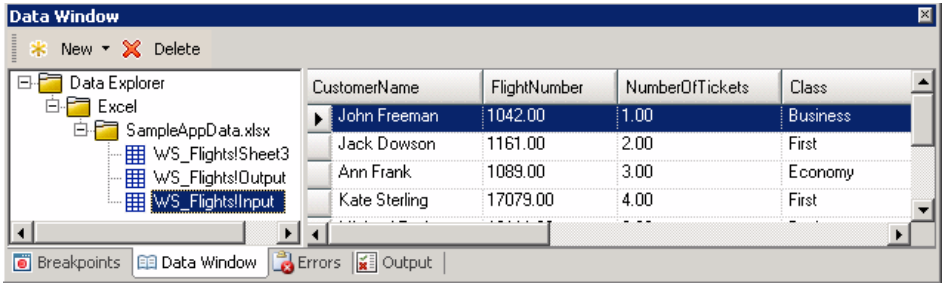
Data Handling User Interface

This section includes:

- Data Window Tab on page 294
- Data Driving Dialog Box on page 296
- Add New Excel File Data Source Dialog Box on page 299
- Add New Local Table Data Source Wizard on page 300
- Add New XML Data Source Dialog Box on page 302
- Attach Data Source to Loop Dialog Box on page 304
- Select Link Source Dialog Box on page 305
- Data Navigation Dialog Box on page 309
- Define/Edit New Data Relation Dialog Box on page 312
- Add New Database Data Source Wizard on page 313

Data Window Tab

Enables you to view the output messages that were generated while compiling and running the test.



To access	Select **View** > **Data Window** (CTRL+ALT+D).
Important information	Click on a data source to see its contents.
Relevant tasks	➤ How to Create Data Sources on page 276 ➤ How to Assign Data to Test Steps on page 283

XML Data Source - Toolbar

The following controls are available when selecting an XML data source in the **Data Window** tab.

UI Elements (from left to right)	Description
	Go back. Go back to previous view.
	Go to start. Go to the first row.
	Go to previous row. Go to the previous row (one row up).
Row	The current row number out of the total rows, such as **1/3** and so forth.

UI Elements (from left to right)	Description
Description	A title for the row currently displayed.
	Go to next. Go to the next row (one row down).
	Go to last. Go to the last row.
	Import data from an XML file. Loads data from an XML file.
	Export data from an XML file. Exports the current data to an XML file.
	Move up. Moves the current row up one position.
	Move down. Moves current row down one position.
	Adds a new empty row at one of the following locations. ➤ **Add after current** ➤ **Add at start** ➤ **Add at end**
	Duplicates the current row at one of the following locations. ➤ **Duplicate after current** ➤ **Duplicate at start** ➤ **Duplicate at end**
	Deletes the current row.

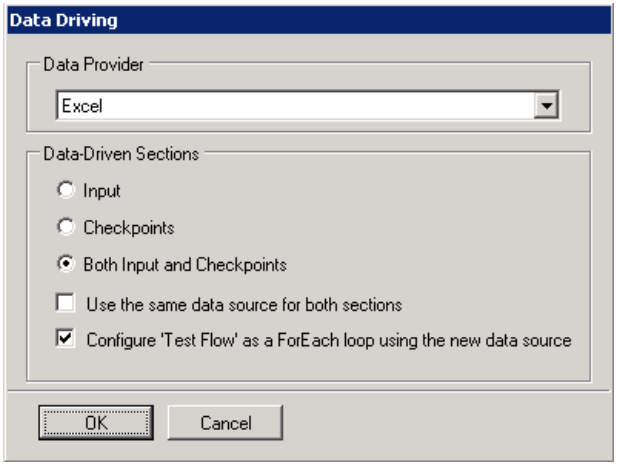
Database Data Source Nodes

The following controls are available when selecting a database type data source in the **Data Window** tab.

UI Element	Description
Refresh	**Refresh.** Updates the data from the database.
Count	**Count.** Opens a message box with a row count.

Data Driving Dialog Box

Enables you to populate the input property and checkpoint values for a step, with data expressions.



To access	Click in the Property Sheet's toolbar.
Relevant tasks	"How to Data Drive a Test Step" on page 291
See also	"Add New Excel File Data Source Dialog Box" on page 299

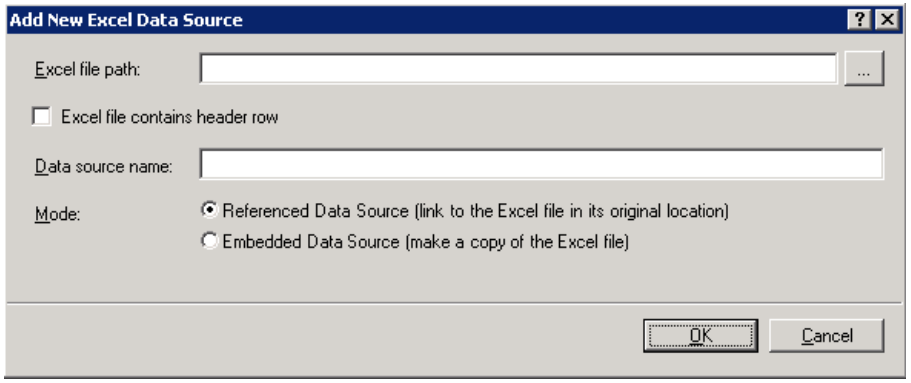
The following elements are included:

UI Elements	Description
Data Provider	The source type of the data: **Excel** or **XML**.
Data Driven Sections - for properties	The section upon which to apply the data driving. These options are available when data driving standard input and output checkpoints properties. ➤ **Input**. All input properties listed in the grid. ➤ **Checkpoints**. All checkpoints listed in the grid. For array type Output properties, you must create at least one array element in order to enable data driving. ➤ **Both Input and Checkpoints**. All properties. **Note:** Activities that do not have output parameters, such as **Report Message** and **System** type activities, will only show the **Input** option.
Data Driven Sections - for schemas	The schema to data drive. These options are available when data driving XSD schema files for an **HTTP**, **SOAP Request**, or **REST** step. ➤ **Request Body**. All input properties associated with the request body. ➤ **Response Body**. All output properties associated with the response body. For array type properties, you must create at least one array element in order to enable data driving. ➤ **Both Request Body and Response Body**. All entities in both schemas. **Note:** For HTTP steps, these options are only available after importing a schema into the **Input** and/or **Checkpoints** sections.
Data Driven Sections - for SOAP Fault properties	**Fault Properties.** The SOAP Fault properties to data drive. This option is available when invoking data driving in the Property Sheet's **SOAP Fault** view.

UI Elements	Description
Data Driven Sections - for SOAP Fault properties	The multipart properties to data drive. These options are available for an HTTP step with a multipart message. ➤ **Request MultipartInfo**. Only Request multipart properties. ➤ **Response MultipartInfo**. Only Response multipart properties. ➤ **Both Request MultipartInfo and Response MultipartInfo**. All multipart properties.
Use the same data source for both sections	Uses the same data source for the Input/Checkpoint properties or Request/Response body. When selected, Service Test creates a single Excel worksheet with columns for both the Input and Output properties. For example, for the sample application's **CreateFlightOrder** step, Service Test creates a single worksheet with columns for the Input properties: Class, CustomerName, DepartureDate, FlightNumber, NumberofTickets, and the Output properties: OrderNumber, and TotalPrice. **Note**: This option is available only when selecting the **Both** option, for an Excel Data Provider type.
Configure 'Test Flow/ Loop' as a ForEach loop using the new data source	Runs the steps in the Test Flow or Loop using a **For Each** type loop, with the values from the newly created data source. The number of iterations is determined by the number of rows in the data source—it performs one iteration for each row. **Note:** This option overrides the configuration that you may have set for the Test Flow/Loop settings. For details, see "Input Properties - Loop" on page 95.

Add New Excel File Data Source Dialog Box

Enables you to add new Excel data source to the Data Explorer within the Data tab.



To access	Select **View** > **Data Window** and select **New** > **Excel File**.
Relevant tasks	"Add an Excel data source" on page 276

User interface elements are described below:

UI Elements	Description
...	**Browse.** Enables you to locate the Excel file containing the data.
Excel file path	The complete path of the Excel file containing the data.
Excel file contains header row	Indicates whether the data in the Excel table contains a header row. Header rows are ignored when running the test.

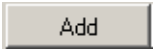
UI Elements	Description
Data source name	The name of data source as it will appear in the Data Window's Excel file node. **Default:** The name of the file.
Mode	The mode of storage for the data source: ➤ **Referenced Data Source.** Links to the Excel file at its original location. If the data changes at its source, it will affect your test. In addition, if you modify the data in Service Test, it will affect the data at its source. ➤ **Embedded Data Source.** Copies the Excel file locally. If the data changes at its source, it will not affect your test.

Add New Local Table Data Source Wizard

Enables you to create local tables in the Data window for use with your test.

To access	Select **View** > **Data Window** and select **New** > **Local Table**.
Relevant tasks	"Add a local data table" on page 277
See also	➤ "Add New Excel File Data Source Dialog Box" on page 299 ➤ "Add New XML Data Source Dialog Box" on page 302

User interface elements are described below:

UI Elements	Description
	Opens the **Add Columns** dialog box for providing a new column's properties.
	Opens the **Edit Columns** dialog box for modifying the properties of the selected column.
	Deletes the selected data table column.

UI Elements	Description
	Raises/lowers the selection in the column list.
Columns	A list of the columns in the local data table.

Add/Edit Columns Dialog Box

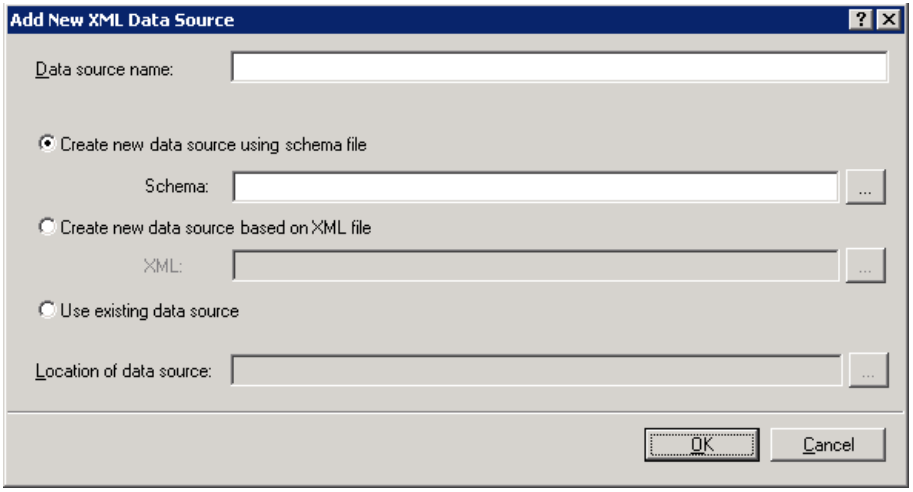
The Add/Edit Columns dialog box lets you define the properties for a new column or edit the properties of an existing one.

User interface elements are described below:

UI Elements (A-Z)	Description
Column Name	An arbitrary name for the column.
Data Type	The type of data in the column: ➤ String ➤ Integer ➤ Double ➤ Date
Description	A description of the data in the column.

Add New XML Data Source Dialog Box

Enables you to add new XSD or XML data sources to the Data Explorer within the **Data Window** tab.



To access	Select **View** > **Data Window** and select **New** > **XML**.
Relevant tasks	"Add an XML data source" on page 278
See also	➤ "Add New Excel File Data Source Dialog Box" on page 299 ➤ "Add New Local Table Data Source Wizard" on page 300

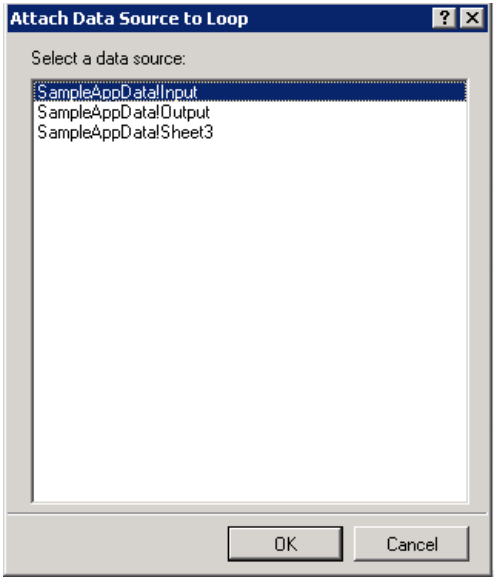
User interface elements are described below:

UI Elements	Description
Data source name	The display name of the data source.
Create new data source using Schema file	Loads the specified schema into the Data Explorer window as a basis for the new data source.

UI Elements	Description
Create new data source based on XML file	Creates a new schema based on the XML file specified in the **XML file path** box.
Use existing data source	Uses an existing data source specified in the **Location of data source** box. Since this data source is referenced, changes made to it will be reflected in all tests using the data source.

Attach Data Source to Loop Dialog Box

This dialog box enables you to add a data source to the current loop.



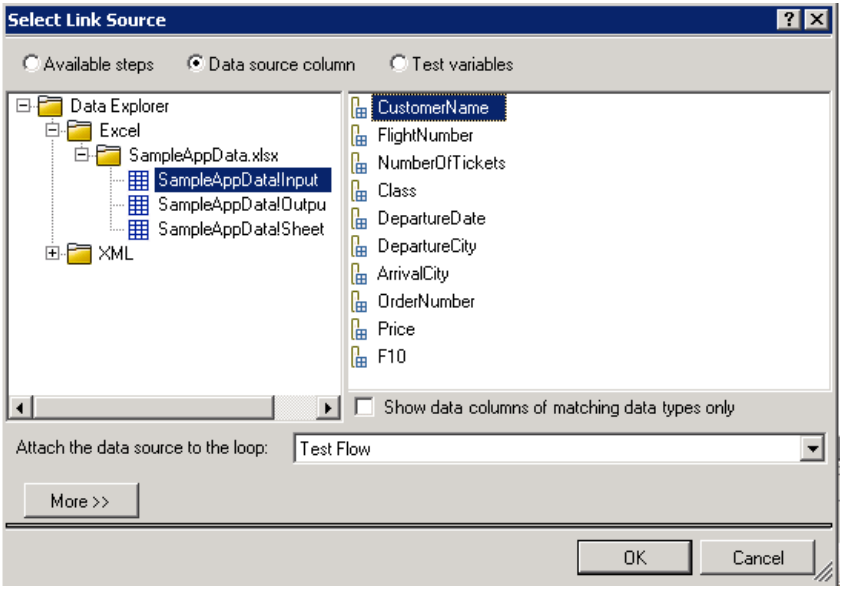
To access	Do the following: **1** In the canvas, click in a loop, but not within a step. **2** In the Property Sheet, select the **Data Sources** view. . **3** Click **Add**.
Important	You must first define data in the Data Window before you generate a list of data sources. When you link any step in the loop to the data, it automatically adds a data source to the loop. For details, see "How to Assign Data to Test Steps" on page 283.
Relevant tasks	"How to Set the Navigation Properties" on page 290
See also	"Data Navigation Dialog Box" on page 309

User interface elements are described below:

UI Element	Description
Select a data source	A list of the available data sources.

Select Link Source Dialog Box

This dialog box enables you to select a data source for the step's properties.



To access	Click in the **Value** column of the property for which you want to select a data source.
Important information	➤ To see the icon, click in the property row and move the cursor to the right of the cell. ➤ A step property not contained within a loop, cannot be linked to a data source. For example, the For Loop's **Number of Iterations** property, cannot be linked to a data source, since it is not contained within the Test Flow loop.
Relevant tasks	"How to Assign Data to Test Steps" on page 283
See also	➤ "How to Create Data Sources" on page 276 ➤ "How to Set the Navigation Properties" on page 290

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
	Expands the dialog box to allow you to use multiple data sources for an expression, and edit the expression manually.
<data source>	The source for the property value: ➤ **Available steps**. A property value from another step. ➤ **Data source column.** A value from a data source—Excel, XML, local table, or database. ➤ **Test variables.** A value of an test variable—either user or system.
<available step list>	A list of the steps whose properties are available as data for the current step. This list could include: ➤ Previous steps. ➤ An input value of the current step, when selecting data for a property in the **Checkpoints** section. ➤ For loop type steps, previous steps or other steps within loop. **Note:** Visible only when choosing **Available steps**.

UI Elements	Description
<step properties grid>	A schema view of the selected step's properties. **Note:** Visible only when choosing **Available steps**.
<data sources and their columns>	A list of the Excel, XML, local table, or database data sources in the left pane, and their columns or schema (for XML) in the right pane. **Note:** Visible only when choosing the **Data source column** option.
<user variable grid>	A list of the user-defined test variables and their values in the current profile. **Note:** Visible only when choosing the **Test variables** option.
<system variable grid>	A list of the system environment variables and their values: ControllerHostName, IsLoadTest, ScenarioID, SystemTempDir, GroupName, TestDir, TestName, LocalHostName, OS, OSVersion, ProductDir, ProductName, ProductVer, and UserName. **Note:** Visible only when choosing the **Test variables** option.
Show data columns of matching data types only	Hides all columns (or schema rows for XML data sources whose data types do not match the input property whose value you want to set. **Note:** Visible only when choosing the **Data source column** option.
Show properties of matching data types only	Hides all properties whose data types do not match the one whose value you want to set. **Note:** Visible only when choosing the **Available steps**.
Attach the data source to the loop	A drop-down list of the loops in the test containing the activity. The data source is only associated with the selected loop. **Note:** Visible only when choosing the **Data source column** option.

Data Retrieval Options for Load Test Enabled Tests

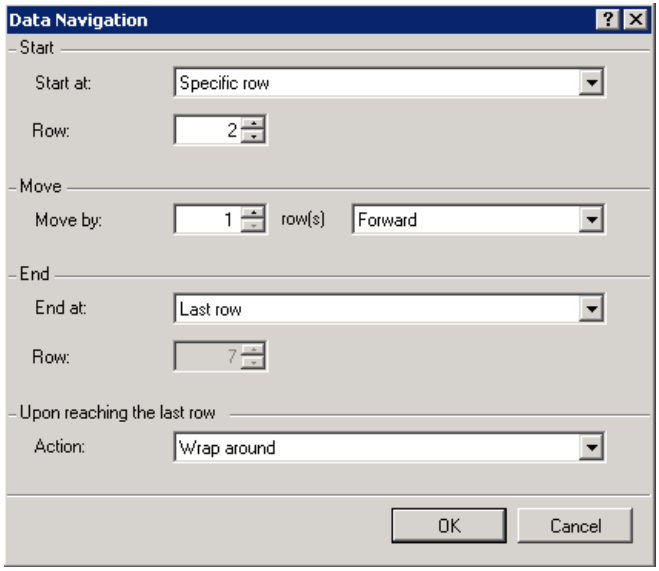
For Load Test Enabled tests, the Select Link Source dialog box provides additional controls for the Input properties, when choosing the **Data source column** option.

User interface elements are described below:

UI Elements	Description
Attach the data source to the loop	The loop with which to associate the data source (for future implementations).
Use a unique value for each Virtual User when load testing	Assigns a unique value to each Virtual User, from the data table, with the following properties: ➤ **Data source parameter**. The parameter to use as a source of values for the selected input property. ➤ **Update value on**. The frequency by which LoadRunner will update the values: **Each Iteration**, **Each Occurrence** (of the parameter), or **Once**. ➤ **When out of values**.The action to do when reaching the end of the data table: ➤ **Abort Vuser.** End the test run for the Virtual User. ➤ **Continue Cyclic**. Return to the top of the data table and use the data in a cyclic manner. ➤ **Continue with Last.** Use the last value retrieved from the data table for all subsequent occurrences.
Allocate Virtual User values in the Controller	Allocates a specific amount of values from the data table for the Virtual User: ➤ **Automatically allocate block size**. Automatically allocates a block size from the data based on the number of iterations and Virtual Users (only for Update value on **Each Iteration).** ➤ **Allocate** <*amount*> **values for each Virtual User.** Allocates a specific number of values for each Virtual User. **Note:** Allocation is only possible when choosing Update value on **Each Occurrence** or **Each Iteration.** For details, see the *HP LoadRunner Controller User Guide*.

Data Navigation Dialog Box

This dialog box enables you to set the data navigation properties for your test loop. You can set the direction in which to use the data, from where to begin, and the condition for selecting data.



To access	Do the following: **1** In the canvas, click inside the loop frame—not on a test step. **2** In the Property Sheet, select the **Data Source** view . **3** Click **Add**. The Attach Data Source to Loop dialog box opens. **4** Select a data source to attach and click **OK**. The data source is added to the grid. **5** Click **Edit**.
Relevant tasks	"How to Set the Navigation Properties" on page 290
See also	"Navigating within the Data" on page 272

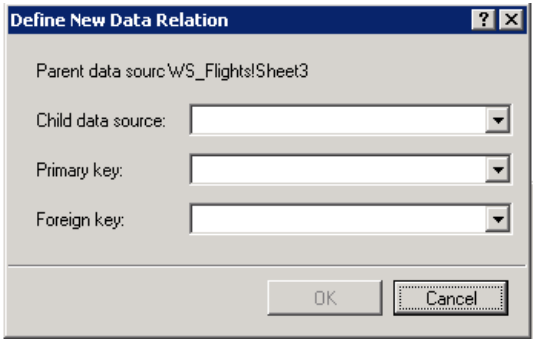
User interface elements are described below:

UI Elements	Description
Start at	The location from where to begin taking data from the data source. ➤ **Start at.** The row of data from which to begin: **First row**, **Last row**, **Specific row**, **Random row,** or **First row matching.** ➤ **Row.** The row number from which to begin (only available when selecting **Specific row**). ➤ **Condition**. The condition the data row must meet in order to start the looping (only available when selecting the **First row matching** option): ➤ **Column.** One of the data source's columns. ➤ **Comparison operator.** =, !=,>, >=, <, <=, Starts, Ends, Contains, or Regex (depending on the row's data type). ➤ **Value.** The value to be matched in order to use this row of data.
Move	The direction and amount of rows by which to advance within the data. ➤ **Move by.** The number of rows to advance. ➤ **Direction.** The direction in which to move: **Forward**, **Backward,** or **Random.**

UI Elements	Description
End at	When to stop the looping through the data: **First row**, **Last row**, **Specific row**, or **First row matching.** ➤ **Row.** The row number in which to end (only available when selecting **Specific row**). ➤ **Condition**. The condition the data row must meet in order to end the looping (only available when selecting the **First row matching** option): ➤ **Column.** One of the data source's columns. ➤ **Comparison operator.** =, !=,>, >=, <, <=, Starts, Ends, Contains, or Regex (depending on row's data type). ➤ **Value.** The value to be matched in order to use this row of data. **Note:** This setting stops the looping only when the loop type is set to '**For Each**' (see "Input Properties - Loop" on page 95) and the current data source whose navigation settings you are editing, is the same data source that is linked to the loop. **Tip:** This setting is only relevant when the **Move** direction is set to **Forward** or **Backward**. For a **Random** direction, each row is visited once in random order. The navigation ends after all of the rows were visited.
Upon reaching the last row - Action: (Forward direction)	The action to take when reaching the last row of data. ➤ **Keep using that row**. Keep using the last row of data for all subsequent loops. ➤ **Wrap around.** Start again from the first row of data.
Upon reaching the first row - Action: (Backward direction)	The action to take when reaching the first row of data. ➤ **Keep using that row**. Keep using the first row of data for all subsequent loops. ➤ **Wrap around.** Start again from the last row of data.

Define/Edit New Data Relation Dialog Box

This dialog box enables you to define or edit a new data relation for a data source. The following is an example of the Define New Data Relation dialog box:



To access	Do the following: **1** Make sure you have at least one data source in the **Data Window** tab. **2** Select a data source in the **Data Window's** left pane. **3** Click in the Property Sheet pane. The **Data Source Properties** view opens. **4** Click **Add** or **Edit**.
Relevant tasks	"Create a new child relation" on page 282

User interface elements are described below:

UI Elements	Description
Child data source	The name of the parent data source, selected in the Property Sheet (read-only).
Child data source	The sheet or table to use for the data source.

UI Elements	Description
Primary key	A column in the parent data source to use as a primary key for the child relation.
Foreign key	The column in the child data source to use as a foreign key for the child relation.

Database Data Source User Interface

Service Test allows you to work with standard database providers to retrieve data for your step's properties.

This section includes the following:

➤ "Add New Database Data Source Wizard" on page 313

Add New Database Data Source Wizard

This wizard enables you to retrieve data rows from a standard database provider. You specify a connection string and an SQL statement to retrieve the data.

To access	Do the following: **1** Make sure the Data Window is visible. If not, select **View** > **Data Window**. **2** In the Data Window pane, select **New** > **Database**.
Relevant tasks	"Add a Database data source" on page 279
Wizard map	This wizard contains: Define the Connection String Page > Define the Query Statement Page
See also	"Query Designer Dialog Box" on page 120

Define the Connection String Page

This wizard page enables you to define a connection string for connecting to the database.

Important information	General information about this wizard is available here: "Add New Database Data Source Wizard" on page 313.
Wizard map	This wizard contains: **Define the Connection String Page** > Define the Query Statement Page

User interface elements are described below:

UI Elements	Description
Connection string	A drop-down list of previously defined connection strings.
	Create New String. Opens the Microsoft Data Link Properties dialog box.
<connection string area>	An editable area for pasting in existing strings, or for editing the connection string selected in the drop-down list. **Tip:** To add a string to the drop-down list, click **Check Connection**.
	Checks the database connection for the string shown in the text area. If it is a valid string, the wizard adds it to the list of Connection strings and changes the status to Connected .

Define the Query Statement Page

This wizard page enables you to provide a query statement for retrieving your data.

Important information	General information about this wizard is available here: "Add New Database Data Source Wizard" on page 313.
Wizard map	This wizard contains: Define the Connection String Page > **Define the Query Statement Page**

User interface elements are described below:

UI Elements	Description
Data source name	The name of the data source as it will be referenced by your test.
SQL Statement	A drop-down list of the available SQL statements.
	Opens the Query Designer. For details, see the "Query Designer Dialog Box" on page 120.
<SQL statement area>	An editable area for pasting existing SQL strings or for editing the SQL string selected in the drop-down list. **Tip:** To add an SQL statement pasted into the text area to the drop-down list, click **Check SQL Statement**.
	Executes the SQL statement shown in the text area and opens the results in a **Query Preview** window. If the statement is valid, the wizard adds it to the drop-down list of SQL statements and changes the status to Query executed successfully .

Troubleshooting and Limitations - Data

This section describes troubleshooting and limitations for working with data.

- Linking to or from an **Any** type XML element is not supported.
- The copying and pasting of data source links is not supported. When providing a link for a property, use the Select Data Source dialog box—do not copy the expression from the **Value** column.
- Keywords such as **#NIL#**, **#Skip#**, and so forth, are not supported in the checkpoint grid. They are supported in the Data Window's data sources.
- Data driving for arrays within checkpoints, using Excel data sources, is not supported.
- Working with an imported Excel file as a data source, may temporarily disable an open Excel document on the same machine.
- Data sources with parent-child relationships, should not be accessed together in the same loop, unless the child data source is used for data-driving array elements. Accessing parent-child data sources in the same loop, may corrupt the test.

8

Run Results Viewer

This chapter includes:

Concepts

- Run Results Viewer Overview on page 318
- Run Results File Location on page 320
- Captured Data on page 321
- Custom Fields on page 325

Tasks

- How to Open Run Results on page 326
- How to Navigate the Run Results Tree on page 327
- How to Manually Submit Defects to ALM/QC on page 329
- How to Export Run Results on page 330

Reference

- Run Results Viewer User Interface on page 332
- Run Results Viewer Commands on page 334
- Run Results Viewer Panes on page 337
- Run Results Viewer Dialog Boxes on page 350

Concepts

Run Results Viewer Overview

After running a test, the Run Results Viewer opens automatically at the end of a test run. You can also open it manually at any time.

The Run Results Viewer contains a description of the steps performed during the run session. It displays details for each iteration of the test run. The results are grouped by the steps in the test.

For an overview of the various panes, see "Run Results Viewer User Interface" on page 332.

Standalone Installation

The Service Test installation includes the Run Results Viewer application to view run results. You can also install the Run Results viewer independently of Service Test on any machine.

To install the Run Results Viewer, run the HP_Run_Results_Viewer.msi file in the DVD's STSetup\MSI\ThirdPartyInstallations folder on any machine.

QuickTest Professional Integration

Service Test lets you perform (UFT) Unified Functional Testing with Quick Test Professional tests. If you run a Service Test test that contains a call to a QuickTest Professional test, or vice versa, you can view the results for all steps performed in the main test and in the called test.

Run Results XML File

The results of each Service Test run session are saved in a single **.xml** file (called **results.xml**). This **.xml** file stores information on each of the run result nodes in the display. The information in these nodes is used to dynamically create **.htm** files that are shown in the top-right pane in the Run Results Viewer.

Each node in the run results tree is an element in the **results.xml** file. In addition, there are different elements that represent different types of information displayed in the run results. You can take run result information from the **.xml** file and use XSL to display the information you require in a customized format (either when printing from within the Service Test Run Results Viewer, when displaying run results in your own customized results viewer, or when exporting the run results to an HTML file).

XSL provides you with the tools to describe exactly which run result information to display and exactly where and how to display, print or export it. You can also modify the **.css** file referenced by the **.xsl** file, to change the appearance of the report (for example, fonts, colors, and so forth).

For example, in the **results.xml** file, one element tag contains the name of a step, and another element tag contains information on the time at which the run session is performed. Using XSL, you could tell your customized run results viewer that the step name should be displayed in a specific place on the page and in a bold green font, and that the time information should not be displayed at all.

You may find it easier to modify the existing **.xsl** and **.css** files provided with Service Test, instead of creating your own customized files from scratch. The files are located in **C:\Program Files\HP\UFT Result Viewer\dat**, and are named as follows:

- **PShort.xsl.** Specifies the content of the run results report printed, or exported to an HTML file, when you select the **Short** option in the Print or Export to HTML File dialog boxes.
- **PDetails.xsl.** Specifies the content of the run results report printed, or exported to an HTML file, when you select the **Detailed** option in the Print or Export to HTML File dialog boxes.
- **PResults.css.** Specifies the appearance of the run results print preview. This file is referenced by all three **.xsl** files.

For more information on printing run results using a customized **.xsl** file, see "Print Dialog Box (Run Results Viewer)" on page 356.

For more information on exporting the run results to a file using a customized **.xsl** file, see "Export Run Results Dialog Box" on page 351.

Run Results File Location

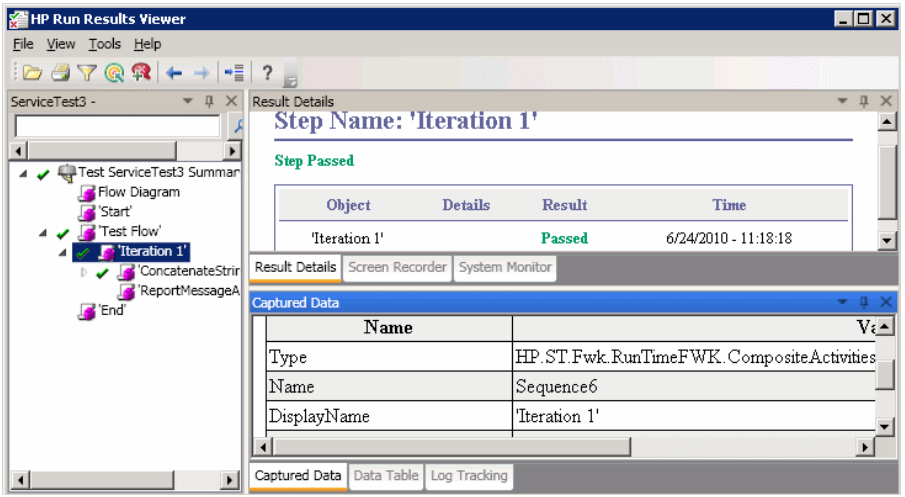
When you run a test, Service Test opens the Run dialog box. You can select a path for the results or use the default location—the test's folder. Alternatively, you can specify a temporary folder, that gets overwritten by each subsequent test run. This is ideal if you are developing a test and have no need for the results.

If your test is stored in ALM/QC (Application Management Lifecycle/ALM/QC), the Run dialog box prompts you to store the results in your ALM/QC project. You cannot modify the project, but you can specify a run name, a test folder, and an instance.

For details, see "How to Run a Test" on page 243.

Captured Data

The captured data tab in the bottom pane shows the data generated during a test run. The contents of the tab differs depending on the level you select in the Results tree in the left pane.



These user interface elements are described below:

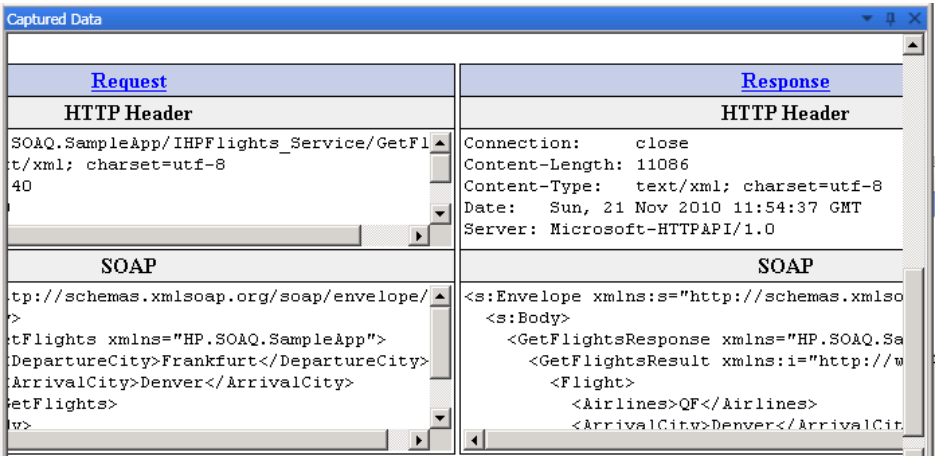
UI Elements	Description
Flow Diagram	A snapshot of the test's canvas.
Start, End	General information about the Start and End activities.
Test Flow / Loop	Information about the loop such as loop name, loop type, display name and so forth.
Iteration	Information about the iteration to which the activities belong.

UI Elements	Description
Activity	Captured data from the activity: ➤ For Service type activities, this level shows the Request and Response data for the operation or method. ➤ For the **Report Message** activity, this level displays the custom message defined in the activity's properties. For details, see "Sending Messages to the Output and Results Viewer" on page 22.
Checkpoint	Data about the checkpoints such as the Expected and Actual values, the method of evaluation (Equals, Does Not Equal, and so forth) and status.

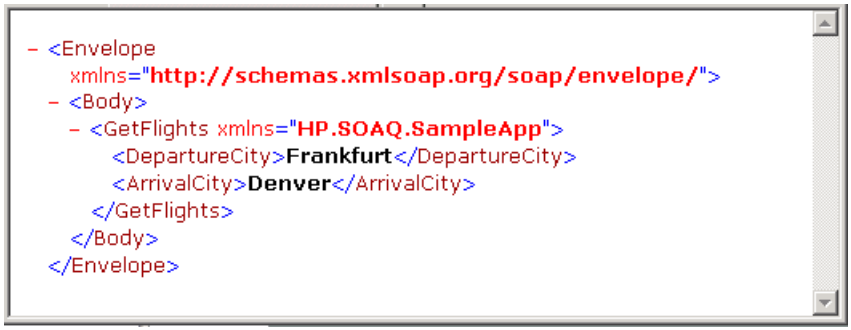
Request and Response

The Activity level captured data contains a table showing the request and response messages.

The following example shows the Request and Response data for the **GetFlights** operation from the sample Web Service.

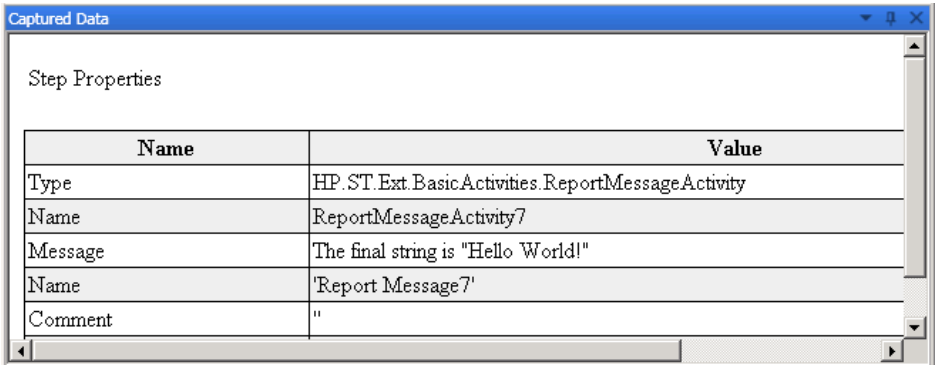


To view the request or response in greater detail, click the **Request** or **Response** link to open the XML in a separate browser.



Messages

For the **Report Message** activity, the Activity level displays a custom message defined in the activity's properties.

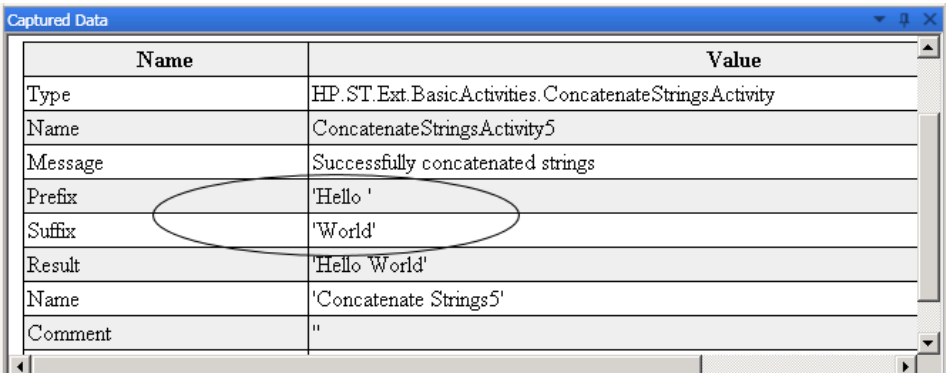


For details, see "Sending Messages to the Output and Results Viewer" on page 22.

Data Driven and Parameterized Properties

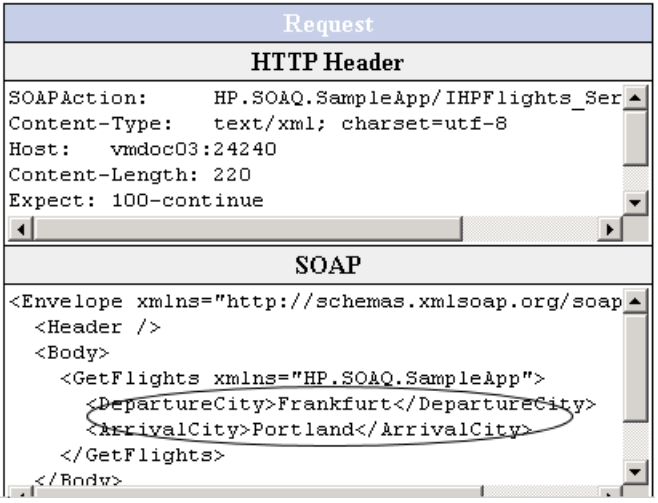
If you parameterized or applied data driving to an activity's properties, the viewer lists the actual values used during the test run, per iteration.

For built-in activities, the **Step Properties** table shows the values used during the test run.



Name	Value
Type	HP.ST.Ext.BasicActivities.ConcatenateStringsActivity
Name	ConcatenateStringsActivity5
Message	Successfully concatenated strings
Prefix	'Hello '
Suffix	'World'
Result	'Hello World'
Name	'Concatenate Strings5'
Comment	''

For service requests, you can observe the actual values in the **Request/ Response** table.



HTTP Checkpoint Response

To view the actual result for HTTP activity checkpoints, select the checkpoint's parent node in the **Captured Data** tab, and scroll to the Response body. Locate the actual result string in the Response body table. The child node of the checkpoint does not provide this information.

Custom Fields

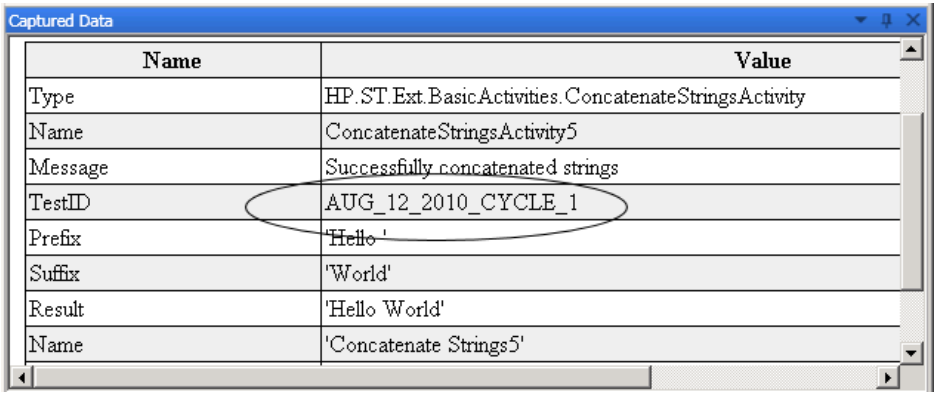
You can use the **Report** function to show custom information in the Results report. You can specify strings or existing arguments and display them in the report.

You add the **Report** function to the test's events. For details, see the section on Custom Code and Events.

The following example prints **AUG-12-2010_CYCLE_1** in the **TestID** property's **Value** column.

```
args.Activity.Report("TestID","AUG-12-2010_CYCLE_1");
```

The report displays the keyword and its value at the Activity level of the results. For details, see the "Captured Data" on page 321.



Tasks

How to Open Run Results

After a test run, Service Test automatically opens the Run Results Window with the results of the current run. This task describes how to open specific run results in the Run Results Viewer.

This task includes the following steps:

- "Open the Run Results Viewer" on page 327
- "Connect to your ALM/QC project - optional" on page 327
- "View saved results" on page 327

Open the Run Results Viewer

If the Run Results Viewer is not open, select **Test** > **View Test Results** in the Service Test window.

Connect to your ALM/QC project - optional

If your run results are saved in ALM/QC, connect to your ALM/QC project before opening the results file. For details, see "HP ALM Connection - Server Connection Dialog Box" on page 359.

View saved results

Click the **Open** button or select **File** > **Open**. The Open Run Results dialog box opens. Browse to the relevant results folder or file. For details, see "Open Run Results Dialog Box" on page 355.

How to Navigate the Run Results Tree

This task describes how to collapse or expand a branch in the run results tree to select the level of detail that the tree displays.

When you open the Run Results Viewer for the first time, the tree expands one level at a time. If the child branches under a parent branch were previously expanded, that state is maintained when you expand or collapse the parent branch.

To expand a specific branch:

Do any of the following:

- Select the branch and click the expand (+) sign to the left of the branch icon.
- Press the plus key (+) on your keyboard number pad.

The tree displays the details for the branch, and the expand sign changes to collapse.

To expand a branch and all branches below it:

➤ Select the branch and press the asterisk (*) key on your keyboard number pad.

To expand all of the branches in the run results tree:

Do any of the following:

➤ Right-click a branch and select **Expand All**.

➤ Select the top level of the tree and press the asterisk (*) key on your keyboard number pad.

To collapse a specific node:

Do any of the following:

➤ Select it and click the collapse (–) sign to the left of the node icon.

➤ Press the minus key (–) on your keyboard number pad.

The node's child nodes disappear from the tree, and the collapse sign changes to expand (+).

To collapse all of the nodes in the tree:

Right-click a node and select **Collapse All**.

To move between previously selected nodes within the run results tree:

Click the **Go to Previous Node** or **Go to Next Node** buttons.

To find specific steps within the Run Results:

Use the **Search** box (shown in the example below).

You can search for text and/or types of nodes. For details, see "Run Results Tree Pane and Search Box" on page 337.

To filter the tree to display only nodes that match certain criteria:

Use the **Filter dialog box (View > Filters).** For details, see "Filter Dialog Box" on page 353.

How to Manually Submit Defects to ALM/QC

This task describes how to manually add defects to a ALM/QC project.

This task includes the following steps:

- "Prerequisites" on page 329
- "Connect to a ALM/QC project" on page 329
- "Open the New Defect dialog box" on page 330
- "Modify the defect information if needed and submit it" on page 330
- "Results" on page 330

1 Prerequisites

Ensure that the ALM/QC client is installed on your computer. (Enter the ALM/QC Server URL in a browser and ensure that the Login screen is displayed.)

2 Connect to a ALM/QC project

Select **Tools > ALM/QC Connection** or click the **ALM/QC Connection** button and connect to a ALM/QC project. "HP ALM Connection - Server Connection Dialog Box" on page 359.

Note: If you do not connect to a ALM/QC project before proceeding to the next step, Service Test prompts you to connect before continuing.

---

3 Open the New Defect dialog box

Select **Tools** > **Add Defect** or click the **Add Defect** button to open the New Defect dialog box in the specified ALM/QC project. The New Defect dialog box opens.

4 Modify the defect information if needed and submit it

Basic information is included in the description, but you can modify the defect if needed.

5 Results

The defect is added to the ALM/QC project's defect database.

How to Export Run Results

This task describes how to export run results to a file. For details on what is included when you export run results, see "Export Run Results Dialog Box" on page 351.

This task includes the following steps:

- "Open the results in the Run Results Viewer" on page 331
- "Specify the export settings" on page 331
- "Save the file" on page 331
- "Results" on page 331

1 Open the results in the Run Results Viewer

For details, see "Open Run Results Dialog Box" on page 355.

2 Specify the export settings

Select **File** > **Export To File**. The Export Run Results dialog box opens. For details on the various settings, see "Export Run Results Dialog Box" on page 351.

3 Save the file

Click **Export**. The Save As dialog box opens. Specify the file name and path, and select the required file type.

Report type	Save as type
Step details	➤ HTML (*.htm, *.htm) (default) ➤ PDF (*.pdf) ➤ DOC (*.doc)
Data Table	Excel (.xls)
Log Tracking	XML (.xml)
Screen Recorder	FlashBack (*.fbr)
System Monitor	➤ Text (*.csv, *.txt) (default) ➤ Excel (.xls) ➤ XML (.xml) ➤ HTML (*.htm, *.htm)

4 Results

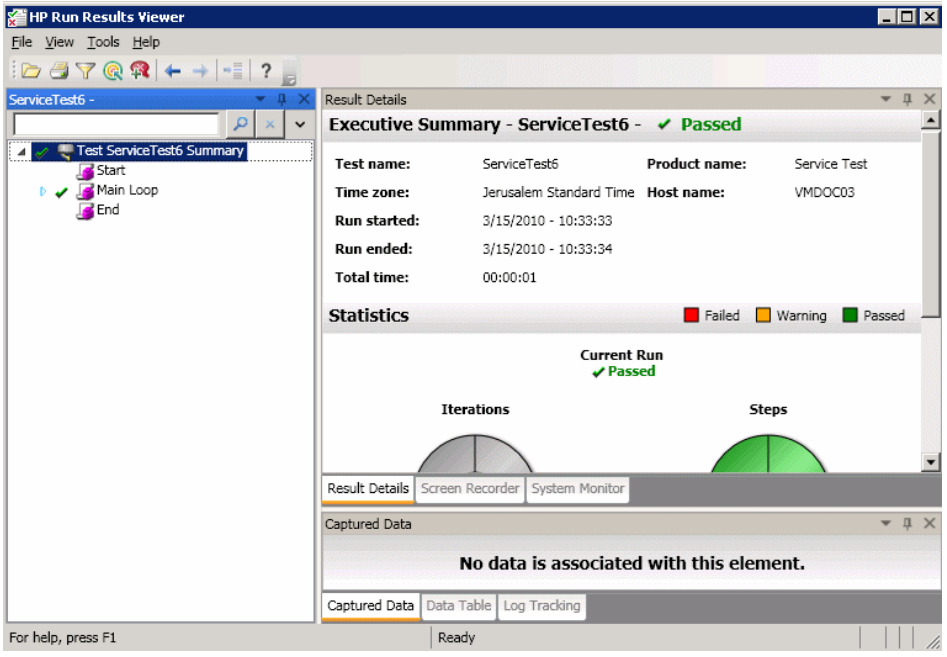
When you click **Save**, the file is exported in the specified format to the designated location.

Reference

Run Results Viewer User Interface

This window enables you to view the results of a test run.

The following example shows an Executive Summary for a Service Test run:



To access	Do one of the following: ➤ Enable it to open automatically after a test run. To enable or disable, select **Edit** > **Settings** > **Show Results After Run.** ➤ Choose **Test** > **View Test Results.**
Relevant tasks	"How to Run a Test" on page 243

By default, the left pane contains the Run Results tree. The right side of the window contains additional panes. These user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Run Results Viewer menu bar and toolbar>	See "Run Results Viewer Commands" on page 334.
Executive Summary pane	➤ A high-level results overview report (pass/fail status) ➤ ALM/QC information for your test (if the test was run from ALM/QC
Run Results Tree pane	➤ A graphical representation of the results in an expandable tree ➤ Displays the test or component steps, specifying exactly where application failures occurred
Result Details pane	Detailed explanations of each step and checkpoint pass or failure, at each stage of the test or component
Captured Data pane	A still image of the state of your application at a particular step. For details, see "Captured Data Pane" on page 345.
<status bar>	Displays the status of the currently selected command.

Note: Certain panes are available only if the results include QuickTest Professional steps.

Run Results Viewer Commands

The Run Results Viewer menu bar and toolbar contain commands to help you view run session results.

Button	Command	Shortcut Key	Description
	File > Open	CTRL+O	Opens the Open Run Results dialog box, enabling you to open saved run results from the file system or from ALM/QC. For details, see "How to Open Run Results" on page 326.
	File > Print	CTRL+P	Opens the Print dialog box, enabling you to print the results of the run session. For details, see "Print Dialog Box (Run Results Viewer)" on page 356.
--	**File > Print Preview**	CTRL+F2	Opens the Print Preview dialog box, enabling you to preview the results of the run session prior to printing. For details, see "Print Preview Dialog Box (Run Results Viewer)" on page 358.
--	**File > Export To File**	--	Opens the Export Run Results dialog box, enabling you to save various parts of the results as external files. For details, see "Export Run Results Dialog Box" on page 351.
--	**File > Recent Files**	--	Lists the recently viewed files.
--	**File > Exit**	--	Closes the Run Results Viewer session.
--	**View > Run Results Viewer Toolbar**	--	Shows or hides the Run Results Viewer toolbar.

Button	Command	Shortcut Key	Description
--	**View > Status Bar**	--	Shows or hides the status bar, which indicates: ➤ a hint about the currently selected command ➤ the status of the Run Results Viewer ➤ the ALM/QC server name and project to which the Run Results Viewer is connected
--	**View > Result Details**	--	Shows or hides the Results Details pane. For details, see "Result Details Pane" on page 341.
--	**View > Captured Data**	--	Shows or hides the Captured Data pane. For details, see "Captured Data Pane" on page 345.
--	**View > Restore Layout**	--	Restores the default layout of the Run Results Viewer.
	View > Filters	CTRL+T	Opens the Filters dialog box, enabling you to filter the information displayed. For details, see the "Filter Dialog Box" on page 353.
--	**View > Expand All**	--	Expands all of the branches in the run results tree. Also available as a context-menu option.
--	**View > Collapse All**	--	Collapses all of the branches in the run results tree. Also available from the shortcut menu.

Button	Command	Shortcut Key	Description
	Tools > Add Defect	--	Enables you to add a defect to your ALM/QC project. If you are not currently connected to ALM/QC, opens the ALM/QC Connection - Server Connection dialog box. For more information, see "How to Manually Submit Defects to ALM/QC" on page 329.
	Tools > ALM/QC Connection	--	Opens the ALM/QC Connection - Server Connection dialog box, enabling you to connect to a ALM/QC project. For more information, see the "HP ALM Connection - Server Connection Dialog Box" on page 359.
	Go to Previous Node	BACKSPACE	Moves the cursor to the previously selected node in the run results tree. For more information, see "How to Navigate the Run Results Tree" on page 327.
	Go to Next Node	ALT+RIGHT ARROW	Moves the cursor to the node you selected in the run results tree prior to clicking the **Go to Previous Node** button. For more information, see "How to Navigate the Run Results Tree" on page 327.
	Help > Help Topics	--	Opens the online help for the Run Results Viewer.
--	**Help > About Run Results Viewer**	--	Displays version information about the Run Results Viewer.

Run Results Viewer Panes

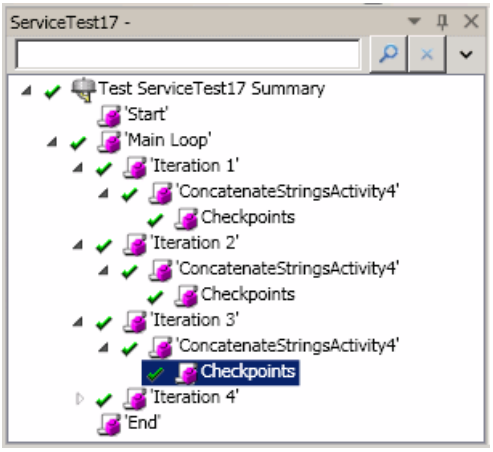
This section includes:

➤ "Run Results Tree Pane and Search Box" on page 337

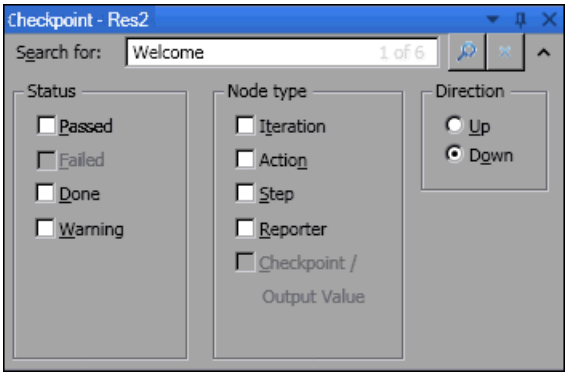
➤ "Result Details Pane" on page 341

Run Results Tree Pane and Search Box

This pane displays the **run results tree**—a hierarchical representation of the test run results.



The following image shows an example of an expanded **Search** box. Six instances of the searched for text, Welcome, were found.



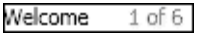
To access	This pane is displayed by default on the left side of the Run Results Viewer. It cannot be hidden.
Important information	You can collapse or expand a node in the run results tree to change the level of detail that the tree displays. You can also use the Search and Filter commands to control what is displayed in the run results tree. For details, see "Run Results Tree Pane and Search Box" on page 337 and "Filter Dialog Box" on page 353.

User interface elements are described below:

UI Elements	Description
✔	A step that succeeded. **Note**: If a test does not contain checkpoints, no icon is displayed.
✘	A step that failed. Note that this causes all parent steps (up to the root action, test, or component) to fail as well.
!	A warning, meaning that the step did not succeed, but it did not cause the action, test or component to fail.

UI Elements	Description
	A step that failed unexpectedly.
	The run session was stopped before it ended.

Search box user interface elements are described below:

UI Elements	Description
	Text box in which you can optionally enter text for which to search. If the specified text is found in one or more tree nodes, the text area indicates this, as shown below:  In this example, 1 of 6 indicates that there are six nodes displaying the text, Welcome, and the first matching node is highlighted in the tree.
	Search. Finds the next instance that matches the criteria you specified in the **Search** box. Click this button to jump to each node that matches your search criteria.
	Cancel. Clears the **Search for** text box.
	Expand or **Collapse**. Shows or hides the lower part of the **Search** box.

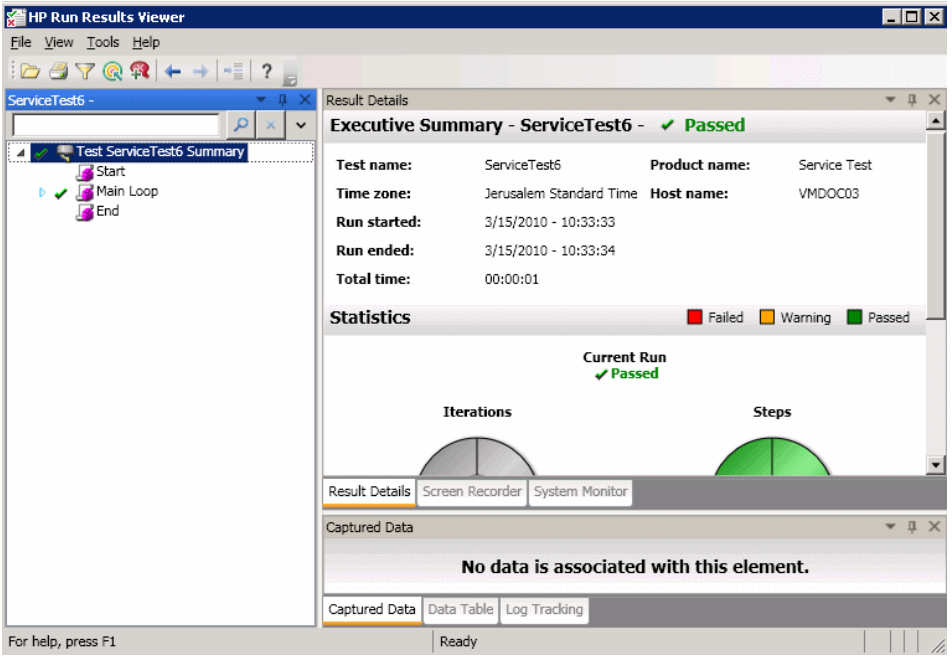
UI Elements	Description
Status	The status for which you want to search. (Optional) Possible values: ➤ **Passed.** Searches for steps that passed and match your other selection criteria. ➤ **Failed.** Searches for steps that failed and match your other selection criteria. ➤ **Done.** Searches for steps with the status **Done** (steps that were performed successfully but did not receive a pass, fail, or warning status) that match your other selection criteria. ➤ **Warning.** Searches for steps with the status **Warning** (steps that did not pass, but did not cause the test to fail) that match your other selection criteria. **Note:** If the tree does not contain any steps that match a particular status, that option is grayed out in the **Search** box.
Node type	The type of node for which you want to search (together with your other search criteria). (Optional) Possible values for Service Test: ➤ **Step.** Searches for steps that match your other selection criteria.
Direction	The direction to search in the tree. Possible values: ➤ **Up** ➤ **Down**

Result Details Pane

This pane displays the details for an individual iteration, an action, or a step that is currently selected in the run results tree.

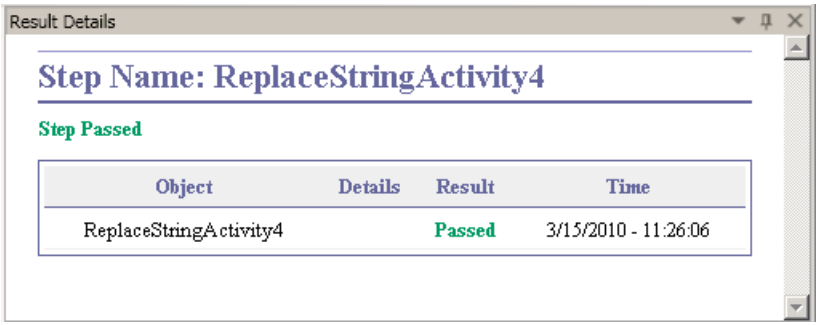
Example of Executive Summary:

The Executive Summary is displayed when the topmost node in the run results tree is selected.



Example of Result Details:

Result details are displayed when any node (other than the topmost node) is selected in the run results tree.



To access	Do the following: **1** Select a node in the run results tree: ➤ To open the **Executive Summary** page, select the topmost node in the tree. ➤ To open the **Result Details** for a step, select the relevant node in the tree. **2** Select the **Result Details** tab. (This assumes that the default layout is displayed.) **Tip:** If the Result Details pane is hidden, select **View** > **Result Details** to show it.
Important information	By default, when the Run Results Viewer opens after a run session, an **Executive Summary** is displayed in the Result Details pane. For any other node, the details in the Result Details pane are specific for the step selected in the run results tree. For details, see "Captured Data" on page 321.

User interface elements are described below (unlabeled elements are shown in angle brackets):

Executive Summary Page

UI Elements	Description
Executive Summary	Includes: ➤ the test or component name and result details ➤ time-related information for the run ➤ the product from which the test or component was run ➤ HP ALM/QC server and project, if Service Test was connected to a ALM/QC project during the run **Note:** If a test or component that is stored in ALM/QC is run from QuickTest, but you choose to store the results in a temporary location, the **Test set** and **Test instance** fields are not displayed in the results.
Statistics	Graphical, status-related statistics for the current run and the previous run (if any). If the test or component was run previously, you can click **Open** to open a previous run results.

Result Details for Step

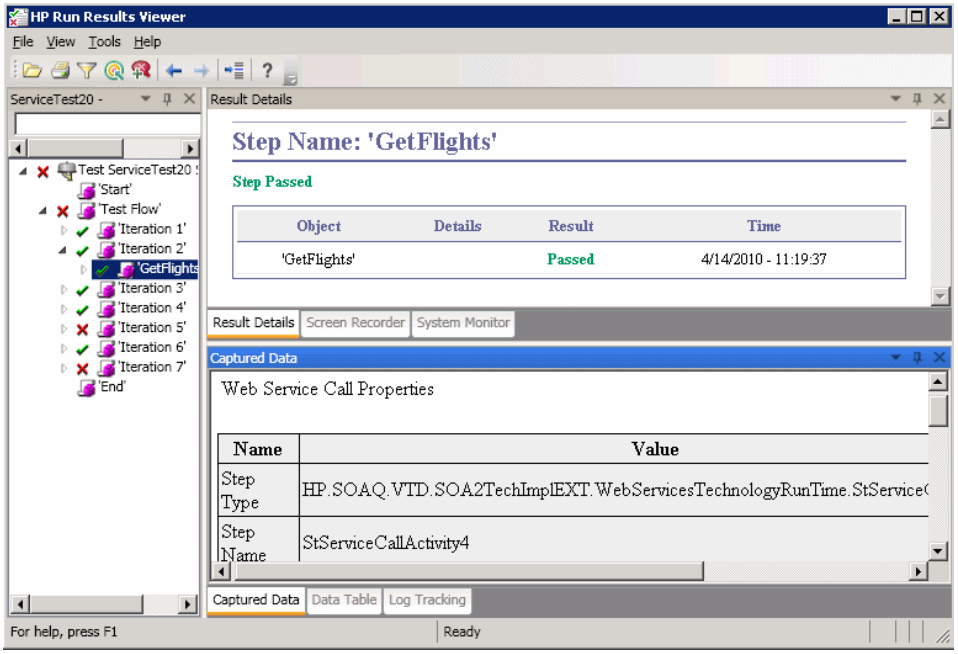
UI Elements	Description
<step name>	The name of the step.

UI Elements	Description
<step status>	The status of the step. Possible values: ➤ **Done.** Relevant for iterations, actions, and steps that ran successfully, but do not contain checkpoints. ➤ **Failed.** Relevant for iterations, actions, and steps that contain checkpoints. ➤ **Passed.** Relevant for iterations, actions, and steps that contain checkpoints. ➤ **Warning.** Relevant for steps that were not successful, but did not cause the test or component to stop running. **Note:** A test, component, iteration, or action containing a step marked **Warning** may still be labeled **Passed** or **Done**.
<step details>	Details about the step, such as the object on which the step was performed, the timestamp, the step results, and so on. The information in this area changes according to the type of step. For more details, see "Captured Data" on page 321.

Captured Data Pane

The captured data tab in the bottom pane shows the data generated during a test run. The contents of the tab differs depending on the level you select in the step pane tree in the left pane.

The following image shows the Captured Data pane showing the Web Service Call properties.



To access	Select **View** > **Captured Data** or click the **Captured Data** tab.
Important information	**Programmatically adding information to the results:** You can programmatically add items to the report. For details, see "Custom Fields" on page 325.
Relevant tasks	"How to Navigate the Run Results Tree" on page 327

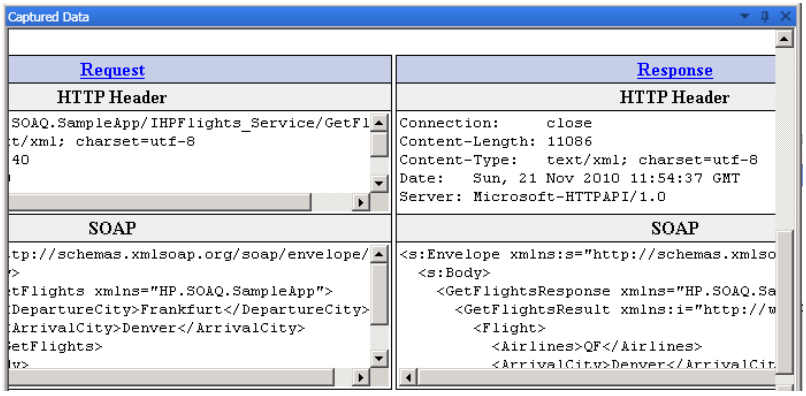
The contents of the Captured Data in Service Test pane differs depending on the level you select in the run results tree.

- **Start, End.** General information about the Start and End activities.
- **Parent step.** Information about the loop to which the test steps belong, such as **Test Flow**.
- **Iteration**. Information about the iteration to which the activities belongs such as the status of the iteration and its display name.
- **Activity.** Captured data from the activity, or operation in the case of Web Services. For Service type activities, this level shows the Request and Response data.
- **Checkpoints.** Data about the checkpoints such as the Expected and Actual values, the method of evaluation (Equals, Does Not Equal, and so forth) and status.

Samples of Captured Data

Request and Response

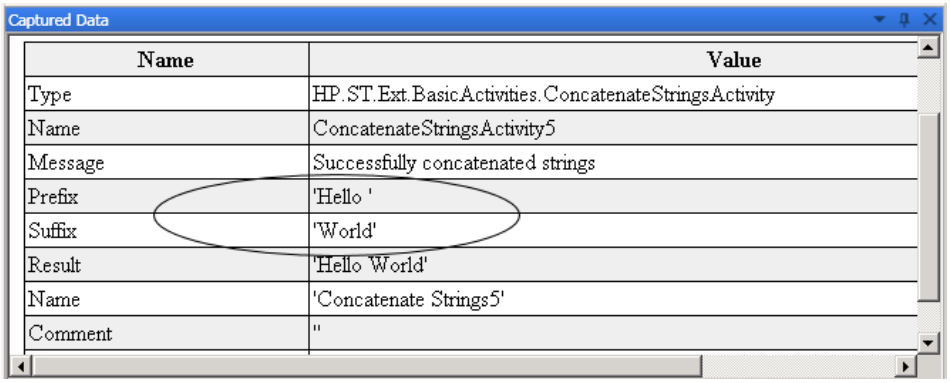
The following example shows the Request and Response data for the **GetFlights** operation from the sample Web Service, on the Step level.



Data Driven and Parameterized Properties

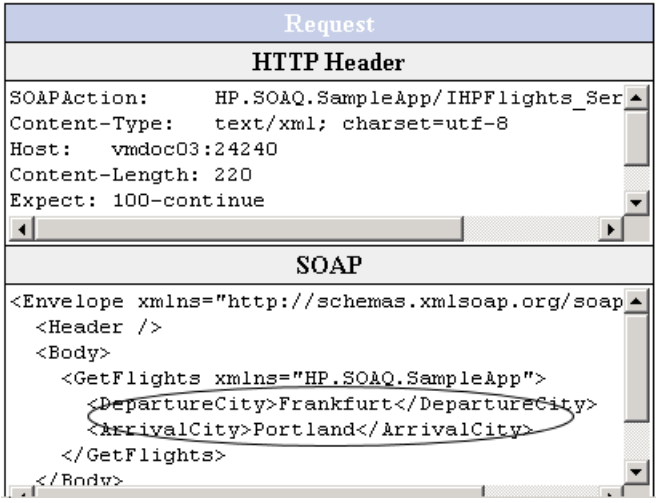
If you parameterized or applied data driving to an activity's properties, the viewer lists the actual values used during the test run, per iteration.

For built-in activities, the **Step Properties** table shows the values used during the test run.



Name	Value
Type	HP.ST.Ext.BasicActivities.ConcatenateStringsActivity
Name	ConcatenateStringsActivity5
Message	Successfully concatenated strings
Prefix	'Hello '
Suffix	'World'
Result	'Hello World'
Name	'Concatenate Strings5'
Comment	''

For service requests, you can observe the actual values in the **Request/Response** table.



Web Service Activity

The following tables shows the results of a Web Service operation from the Sample application, **GetFlights**:

In the results tree:	In the Result Details pane:
Step Type	The type of step as it is defined in the source code. This field is not editable by the user. For example HP.ST.Ext.WebServicesTechnologyRT.StServiceCallActivity.
Step Name	The step name as it appears in the test's source code, visible when you create an event handler for a step. For example StServiceCallActivity1. This field is read-only.
Service Name	The service name as defined in the WSDL.
Port Name	The name of the operation's port.
Operation Name	The name of the operation.
Endpoint Address	The endpoint for the service to which the requests are sent.
SOAP Action	The full path of the SOAP action, such as HP.SOAQ.SampleApp/IHPFlights_Service/CreateFlightOrder.
Content Type	The content type of the request message, for example, text/xml, charset=utf
Message Exchange Pattern	The pattern of the messages, for example Request {followed by) Response.
HTTP Status Code	The HTTP Status code returned by the server.

Concatenate String Activity

The following tables shows the results of a the String Manipulation Activity, Concatenate String.

In the results tree:	In the Result Details pane:
Type	The internal type name of the step. HP.ST.Ext.BasicActivities.ConcatenateStringsActivity

In the results tree:	In the Result Details pane:
Name	The step name as it appears in the test's source code, ConcatenateStringsActivity. This is visible when you create an event handler for a step.
Message	A message indicating the status and purpose of the step. For example, "Successfully concatenated strings."
Prefix	The built-in property, **Prefix**.
Suffix	The built-in property, **Suffix**.
Result	The resulting text after the step was executed.
Name	The name of the step as it appears in the canvas, **ConcatenateStringsActivity**, or a custom name.
Comment	The comment entered in the step's **General** tab.
Status	The run status, such as **Done**, **Failed**, and so forth.

Checkpoint Captured Data

The Checkpoint level captured data is only presented if you enable Checkpoints for the test run. A grid provides the following information for each checkpoint:

In the results tree:	In the Result Details pane:
Name	The checkpoint index. For example, Checkpoint(), Checkpoint1, and so forth.
Result	Passed or Failed, indicated graphically with a check or X mark.
Property	The path of the checkpoint in the SOAP envelope. For example, Envelope[1]\Body[1]\CreateFlightOrderResponse[1]\CreateFlightOrderResult[1]\OrderNumber[1].
Actual Result	The actual output value received in the response.

In the results tree:	In the Result Details pane:
Evaluation Style	The type of comparison operator, such as **EqualTo**, and so forth.
Expected Values	The response you expected to receive, from the Checkpoint grid's **Expected Value** field.
Details	Checkpoint details, when relevant. For example, if you add an **Assert** function to the step's event handler, the text will appear in this field. For details, see the "How to Write Checkpoint Events" on page 389.

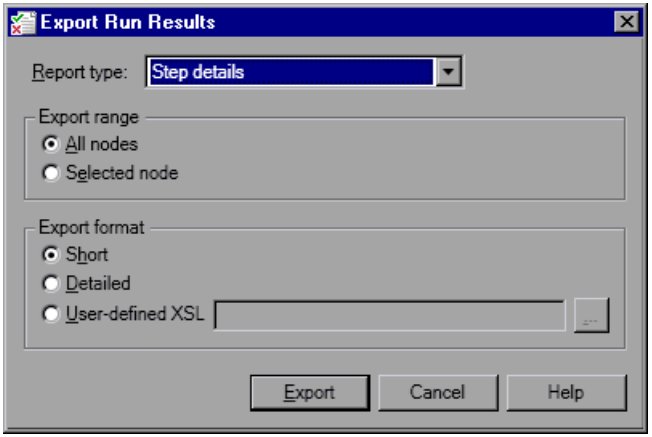
Run Results Viewer Dialog Boxes

This section includes (in alphabetical order):

- ➤ Export Run Results Dialog Box on page 351
- ➤ Filter Dialog Box on page 353
- ➤ Open Run Results Dialog Box on page 355
- ➤ Print Dialog Box (Run Results Viewer) on page 356
- ➤ Print Preview Dialog Box (Run Results Viewer) on page 358
- ➤ HP ALM Connection - Server Connection Dialog Box on page 359

Export Run Results Dialog Box

This dialog box enables you to export run results to a file so that you can view them even if the Run Results Viewer is unavailable. For example, you can send the file containing the run results in an e-mail to a third-party who does not have the Run Results Viewer installed.



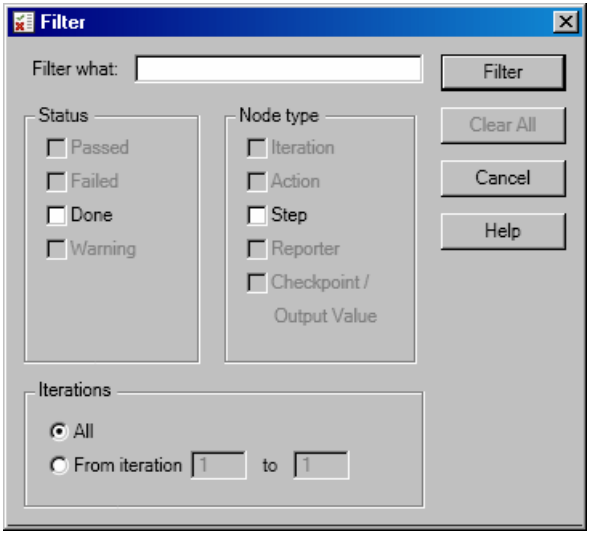
To access	Select **File** > **Export to File**.
Important information	**The export time varies according to the size of the results file and the file type you select.** When selecting the file type, consider the length of time it will take to generate different document types, especially for a report with many images. HTML files generate the fastest, followed by PDF and DOC. When exporting a report with 100 or more images to a DOC file, a dialog box is displayed reminding you that it may take a long time to generate the file. The dialog box gives you the option to continue exporting with images, continue exporting without images, or to export to PDF.
Relevant tasks	"How to Export Run Results" on page 330

User interface elements are described below:

UI Elements	Description
Report type	The type of report you want to export, for example, Step Details.
Export range	Relevant only for **Step Details** report type. ➤ **All nodes.** Exports the results for the entire test or component. ➤ **Selected node.** Exports run result information for the selected branch in the run results tree.
Export format	Relevant only for **Step Details** report type. ➤ **Short.** Exports a summary line (when available) for each item in the run results tree. This option is only available if you selected **All nodes** in **Export range**. ➤ **Detailed.** Exports all available information for each item in the run results tree, or for the selected branch, according to your **Export range** selection. (In the Run Results Viewer, these images are displayed in the Captured Data pane.) ➤ **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the exported report, and the way it should appear. For more information, see the "Run Results XML File" on page 319.

Filter Dialog Box

This dialog box enables you to filter the results tree to display only those nodes that match the conditions that you specify.



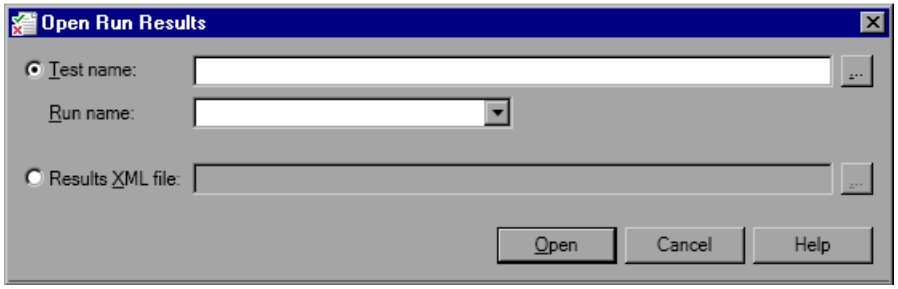
To access	Select **View** > **Filters.**

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Filter box>	The text by which you want to filter.
Status	➤ **Passed.** Displays the run results for the steps that passed. ➤ **Failed.** Displays the run results for the steps that failed. ➤ **Done.** Displays the run results for the steps with the status **Done** (steps that were performed successfully but did not receive a pass, fail, or warning status). ➤ **Warning.** Displays the run results for the steps with the status **Warning** (steps that did not pass, but did not cause the test to fail).
Node type list	Displays all results that match your selection criteria based on: (Optional) ➤ **Iteration.** Displays the run results for the iterations specified in the **Iterations** area. ➤ **Action.** Displays the run results for all actions in the run results tree that match your other selection criteria. ➤ **Step.** Displays the run results for all steps in the run results tree that match your other selection criteria. ➤ **Reporter.** Displays the run results for all Reporter steps in the run results tree that match your other selection criteria. (QuickTest Professional only) ➤ **Checkpoint/Output Value.** Displays the run results for all checkpoint and output value steps in the run results tree that match your other selection criteria.
Iterations	➤ **All.** Displays run results from all iterations. ➤ **From iteration X to Y.** Displays the run results from the specified range of test iterations.

Open Run Results Dialog Box

This dialog box enables you to select the results to display in the Run Results Viewer.



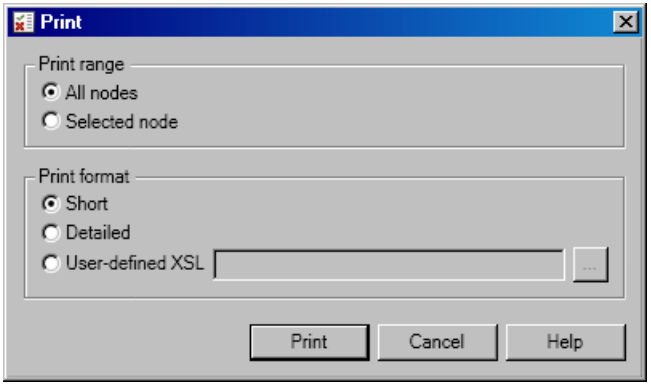
To access	Select **File** > **Open** or click the **Open** button .
Important information	➤ To view results stored in ALM/QC, you must first connect to your ALM/QC project. For details, see "HP ALM Connection - Server Connection Dialog Box" on page 359. ➤ By default, results files for tests are stored in the Reports folder under the user's My Documents folder.
Relevant tasks	"How to Open Run Results" on page 326

User interface elements are described below.

UI Elements	Description
Test name	The name of the test for which you want to view results. The test can be located in the file system or in a ALM/QC project. If you select the **Test name** option, you must also specify the **Results folder name**. The Browse button allows you to navigate to the location of the test.
Results folder	The results for a particular run.
Results XML file	An XML file in which the results are stored. The file must be located in the file system.

Print Dialog Box (Run Results Viewer)

This dialog box enables you to print run results from the Run Results Viewer. You can select the type of report you want to print, and you can also create and print a customized report.



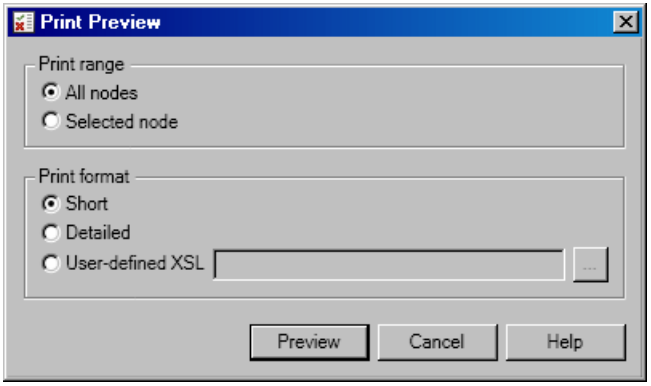
To access	Use one of the following: ➤ Select **File** > **Print.** ➤ Click the **Print** button .

User interface elements are described below:

UI Elements	Description
Print range	➤ **All nodes.** Prints the run results for the entire test or component. ➤ **Selected nodes.** Prints the run results information for the selected branch in the run results tree.
Print format	➤ **Short.** Prints a summary line (when available) for each item in the run results tree. This option is only available if you selected **All** in **Print range**. ➤ **Detailed.** Prints all available information for each item in the run results tree, or for the selected branch, according to your selection in **Print range**. The printed report includes still images associated with the steps in your run results. If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included. ➤ **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the printed report, and the way it should appear.

Print Preview Dialog Box (Run Results Viewer)

This dialog box enables you to preview run results on screen before you print them. You can select the type and quantity of information you want to view, and you can also display the information in a customized format.



To access	Select **File** > **Print Preview.**

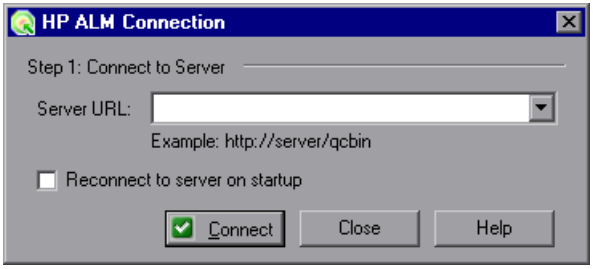
User interface elements are described below:

UI Elements	Description
Preview	Displays the run results on screen as they will appear when printed. **Tip:** If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the **Page Setup** button in the Print Preview window and change the page orientation from **Portrait** to **Landscape**.

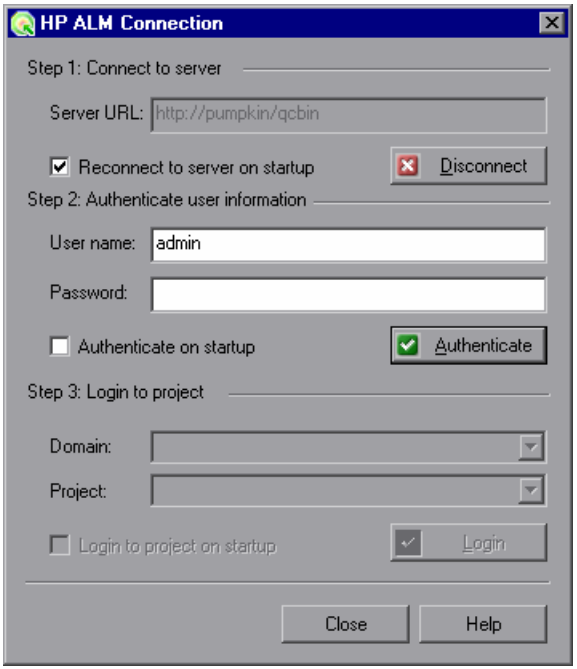
UI Elements	Description
Print range	➤ **All nodes.** Previews the run results for the entire test. ➤ **Selected nodes.** Previews run results information for the selected branch in the run results tree.
Print format	➤ **Short.** Previews a summary line (when available) for each item in the run results tree. This option is only available if you selected **All** in **Print range**. ➤ **Detailed.** Previews all available information for each item in the run results tree, or for the selected branch, according to your selection in **Print range**. The preview includes still images associated with the steps in your run results. If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included. ➤ **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the preview, and the way it should appear. For more information, see "Run Results XML File" on page 319.

HP ALM Connection - Server Connection Dialog Box

This dialog box enables you to connect or disconnect to or from a ALM/QC project.



After you connect to the server, the dialog box expands to include the remaining connection fields.



To access	Select **Tools** > **ALM/QC Connection.**
Important information	**Connect.** The connection process has two stages. First, you connect to a local or remote ALM/QC server. This server handles the connections between Service Test and the ALM/QC project. Next, you log in and choose the project you want the Run Results to access.
Relevant tasks	"How to Open Run Results" on page 326
See also	"Run Results Viewer Overview" on page 318

User interface elements are described below:

UI Elements (A-Z)	Description
Authenticate / Change User	Authenticates your user information against the ALM/QC server. **Note:** After your user information has been authenticated, the edit boxes in the **Authenticate** user information area are displayed in read-only format. The **Authenticate** button changes to a **Change User** button. **Tip:** You can log in to the same ALM/QC server using a different user name by clicking **Change User**, and then entering a new user name and password and clicking **Authenticate** again.
Authenticate on Startup	Automatically authenticates your user information against the ALM/QC server every time you open the Run Results Viewer.
Close	Closes the ALM/QC Connection dialog box.
Connect / Disconnect	Connects or disconnects the Run Results Viewer to or from the selected ALM/QC server. (After you successfully connect to a ALM/QC server, the button changes to **Disconnect**.)
Domain	The domain that contains the ALM/QC project. **Note:** Only those domains to which you have permission to connect are displayed.
Login / Logout	➤ **Login.** Logs into the selected domain and project using the current user information. (After you successfully log into a project, the button changes to **Logout**.) ➤ **Logout.** Logs out of the selected domain and project.
Login to project on startup	Automatically logs into the selected project every time the Run Results Viewer opens.
Password	Your ALM/QC password.
Project	The ALM/QC project with which you want to work. **Note:** Only those projects for which you are a defined user are displayed.

UI Elements (A-Z)	Description
Reconnect to Server on startup	Automatically reconnects to the ALM/QC server every time you open the Run Results Viewer.
Server URL	The URL address of the Web server where ALM/QC is installed. You can choose a ALM/QC server accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).
User name	Your ALM/QC user name.

Guidelines for Windows Vista, Windows 7, Server 2008, and Server 2008 R2 Users

The security settings in the following operating systems may prevent you from connecting to an HP ALM or ALM/QC project:

- Windows Vista
- Windows 7
- Windows Server 2008
- Windows Server 2008 R2

This can occur when the UAC (User Account Control) option is set to ON, and you have never connected to an HP ALM or ALM/QC project.

To connect to HP ALM or ALM/QC for the first time, you must disable the UAC option. After you successfully connect to HP ALM or ALM/QC, you can turn the UAC option on again. Thereafter, you should be able to connect to HP ALM or ALM/QC, as needed.

To enable Service Test to connect to an HP ALM or ALM/QC project:

For Microsoft Windows Vista and Windows Server 2008:

1. Log in as an administrator.
2. From the Control Panel, select **User Accounts** > **Change Security Settings**, and clear the **Use User Account Control (UAC) to help protect your computer** check box. You may need to restart your computer for the changes to take effect.

3 Connect to HP ALM or ALM/QC, as described above.

4 Select the **Use User Account Control (UAC) to help protect your computer** check box and click **OK** to turn the UAC option on again.

For Microsoft Windows 7 and Windows Server 2008 R2:

1 Log in as an administrator.

2 From the Control Panel, select **User Accounts** > **User Accounts** > **Change User Account Settings**.

3 In the User Account Control Settings window, move the slider to **Never notify**. You may need to restart your computer for the changes to take effect.

4 Connect to HP ALM or ALM/QC, as described above.

5 Return to the User Account Control Settings window, and restore the slider to its previous position to turn the UAC option on again.

9

ALM/QC Integration

This chapter includes:

Concepts

- Managing Tests Through HP ALM/QC Overview on page 366
- ALM/QC Tests and Version Control on page 367

Tasks

- How to Work with Tests in ALM/QC on page 368
- How to Save Tests to ALM/QC Projects on page 370
- How to View a List of Previous Test Versions on page 371

Reference

- ALM/QC User Interface on page 372

Troubleshooting and Limitations - ALM/QC Integration on page 375

Concepts

Managing Tests Through HP ALM/QC Overview

Service Test integrates with supported versions of HP ALM/QC (Application Management Lifecycle/Quality Center). ALM/QC integration allows you maintain projects with a different kinds of tests (such as Service Test and QuickTest tests, business process tests, manual tests, tests created using other HP products, and so forth) that cover all aspects of your application's functionality. Each test in your project is designed to fulfill a specified testing requirement of your application. To meet the goals of a project, you organize the tests in your project into unique groups.

HP ALM/QC provides an intuitive and efficient method for scheduling and running tests, collecting results, analyzing the results, and managing test versions. They also features a system for tracking defects, enabling you to monitor defects closely from initial detection until resolution.

Service Test integrates with Service Test Management, the ALM/QC tool for storing and managing application components, such as Web services. Service Test Management is an extension of ALM/QC and it provides an efficient method for storing and retrieving application components.

In order for Service Test to work with ALM, you must connect to the Web server on which ALM/QC is installed. You can connect to either a local or remote server.

For more information on working with ALM/QC, see the *HP Application Lifecycle Management User Guide*.

ALM/QC Tests and Version Control

If your test is saved in an ALM/QC project that uses version control, the process of opening and saving a test is different.

Tests with version control are either in a state of checked-in or checked-out. When you are working with a test in a checked-out state, any changes you make will not be saved on the ALM/QC server until you check in the test. If you save the test from within Service Test, a temporary file is saved onto your machine that protects the test in case your computer crashes.

If you are working with a test in a checked-in state, the test is read-only and you cannot save your changes until you check out the test.

If a particular test is being saved to ALM/QC for the first time, and the project uses version control, the test automatically starts in a checked-out state.

Tasks

How to Work with Tests in ALM/QC

The following steps describe the workflow of how to work with tests saved in an ALM/QC project.

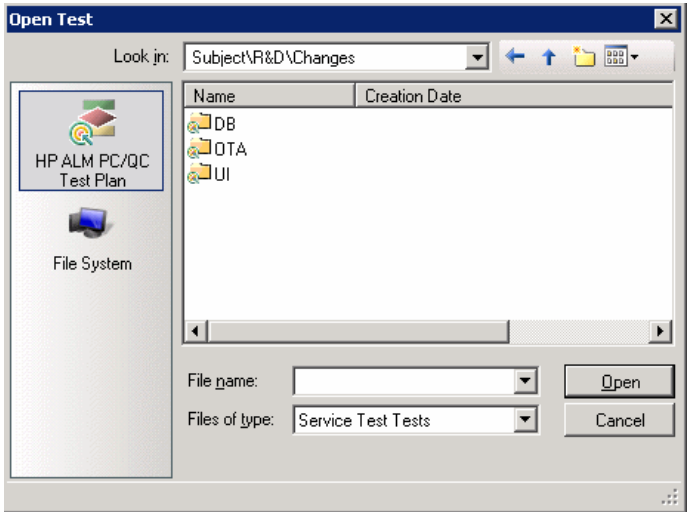
- "Connect to ALM/QC" on page 368
- "Open the test" on page 369
- "(Version control only) Check in/out the test" on page 369
- "(Version control only) Cancel a check out" on page 369
- "Save the test" on page 369

Connect to ALM/QC

Open a connection to the ALM/QC server. Log into the project that contains the test. For task details, see "How to Connect to ALM/QC" on page 370.

Open the test

Select **File** > **New** menu to create a new test, or choose **Open** > **Test** to open a test already saved in ALM/QC. Select the To find the test, select the **HP ALM/QC Test Lab** module, and drill down in the folder tree until you locate it.



(Version control only) Check in/out the test

If the ALM/QC project has version control, tests are defined as being either checked-in or checked-out. For more details, see "ALM/QC Tests and Version Control" on page 367. To check in and check out scripts, select **File** > **ALM/QC Version Control** > **Check In/Out**.

(Version control only) Cancel a check out

If you checked out a test and do not want to save the changes, you can return the status of the test to checked-in without saving by selecting **File** > **ALM/QC Version Control** > **Undo Check Out**.

Save the test

Select **File** > **Save Test As**. Click the **HP ALM/QC Test Plan** button and navigate to the desired location. If you connected to a project that uses version control but is not checked out, the test is only saved as a temporary file on your local machine.

How to Connect to ALM/QC

To store and retrieve scripts from ALM/QC, you need to connect to a project.

1 Select **File** > **HP ALM/QC Connection**.

2 Enter a URL, user name and password and click **Connect**. Use the complete URL. For example, http://servername.mylab:8080/qcbin.

Note: An **http/https** prefix and port number are mandatory.

---

3 Select a domain and project for your test and click **Login**.

4 To logout from the project, but remain connected to ALM/QC, click **Logout**. This is useful if you want to switch to a different project.

5 To disconnect from ALM/QC, click **Disconnect**.

For user interface details, see "HP ALM/QC Connection Dialog Box" on page 373.

How to Save Tests to ALM/QC Projects

The following steps describe how to save a Service Test project to the ALM/QC repository:

➤ "Open a test" on page 370

➤ "Connect to ALM/QC" on page 371

➤ "Save the test" on page 371

➤ "Save the test to ALM/QC" on page 371

Open a test

Create or open a test. Choose **File** > **Open** > **Test** or click the **Create New Test** link on Service Test's **Start Page** tab.

Connect to ALM/QC

Open a connection to the ALM/QC server and project that you want to store the test. For task details, see "How to Connect to ALM/QC" on page 370.

Save the test

Choose **File** > **Save** to save the test to a temporary folder.

Save the test to ALM/QC

Choose **File** > **Save As** and select the **HP ALM/QC Test Plan** button. Navigate to the desired location or use the New Folder button to create a new directory for the test. Specify a **File Name** and click **Save**.

How to View a List of Previous Test Versions

If your test is saved in an ALM/QC project that uses version control, you can view a list of the versions of the test. The following steps describe how to do this.

- "Connect to ALM/QC" on page 371
- "Open a test" on page 371
- "View a list of the versions" on page 371

1 Connect to ALM/QC

Open a connection to the ALM/QC server and project that you want to store the test. Make sure the ALM/QC project supports versioning.

For task details, see "How to Connect to ALM/QC" on page 370.

2 Open a test

Create or open a test. Choose **File** > **Open** > **Test**.

3 View a list of the versions

Select **File** > **HP ALM/QC Version Control** > **Version History** and view the list.

Reference

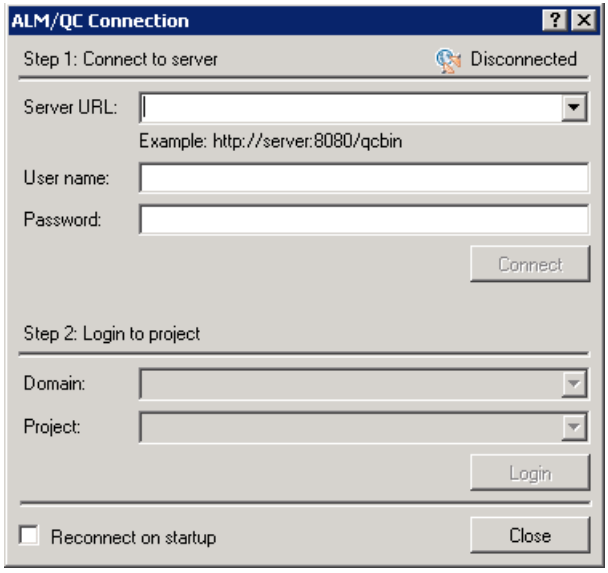
ALM/QC User Interface

This section includes:

➤ HP ALM/QC Connection Dialog Box on page 373

HP ALM/QC Connection Dialog Box

This dialog box enables you to connect to a ALM/QC project from within Service Test.



To access	Select **File** > **HP ALM/QC Connection ...**
Relevant tasks	➤ "Managing Tests Through HP ALM/QC Overview" on page 366 ➤ "How to Work with Tests in ALM/QC" on page 368

User interface elements are described below:

UI Elements (A-Z)	Description
Step 1: Connect to Server	➤ **Server URL.** The URL of the server that contains ALM/QC. ➤ **User Name.** Your ALM/QC user name. ➤ **Password.** Your ALM/QC password. ➤ Connect Connects to the ALM/QC server with your user information. ➤ Disconnect Disconnects from the ALM/QC server.
Step 2: Login to Project	➤ **Domain.** The domain that contains the ALM/QC project. Only those domains containing projects to which you have permission to connect to are displayed. ➤ **Project.** Enter the ALM/QC project name or select a project from the list. Only those projects that you have permission to connect to are displayed. ➤ Login Logs in to the selected project. ➤ Logout Logs out from the current project.
Reconnect on startup	➤ Connects to ALM/QC the next time your open Service Test.

Troubleshooting and Limitations - ALM/QC Integration

This section describes troubleshooting and limitations for ALM/QC integrations.

➤ The following prerequisite applies if there is a forward proxy with Basic Authentication between the server and client machines. Before the first connection to an ALM platform, each ALM client must configure the proxy credentials by using the **Webgate Customization Tool**. If you do not run WebGate, you may be unable to connect, or you may need to enter your credentials multiple times.

To run the tool:

a Go to the following location on the ALM client machine: http://<ALM Platform server name>[<:port number>]/qcbin/Apps/

b In **WebGate Customization**, click in the **Proxy Credentials** area. Select the **Use these credentials** check box, and type values in the **Proxy Username** and **Proxy Password** boxes.

c Click **Save** and then **Close**.

➤ For Windows 2003 Server installations of Service Test, you must perform the following steps in order run a test stored in ALM/QC.

a Select **Start** > **Run**.

b In the Run dialog type dcomcnfg. The **Component Service** Console opens.

c In the left pane, expand the **Component Services** tree. Select the **Computers** -> **My Computer** -> **COM+ Applications** node.

d In the right pane, select **Service Test Remote Agent** and select **Properties** from its shortcut menu.

e Select the **Identity** tab. Select **This User** and enter the credentials of a user with administrator privileges.

f Click **OK** and close all of the dialog boxes.

Note: This is only required if ALM/QC is not installed on the same server as Service Test.

- For Service Test Management version 10.50—Before importing a WSDL from Service Test Management, you must connect at least once to the Service Test Management- ALM/QC server, through Internet Explorer.
- To enable test versioning in ALM/QC versions prior to 11.00, create a file in the <QC installation folder>\repository\sa\DomsInfo\Metadata\TEST folder called ServiceTest.xml with the following content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Test Type="SERVICE-TEST">
  <Repository>
    <Folders ID="1" Filter="." BlindCopy="true" />
  </Repository>
  <Versioning Enabled="true" />
  <Baselining Enabled="false" />
 </Test>
```

- The HP ALM/QC **Baseline** feature is not supported.
- The HP ALM/QC **ERI** feature is not supported. Due to this limitation, you cannot save tests with resources, such as QTP tests, external data tables, and so forth, to ALM/QC. Resources remain stored locally.
- When calling a Service Test test from a QuickTest test, and both tests are stored in ALM/QC, the results of the Service Test step may not appear in the Run Results.
- If you are connected to a ALM/QC server, and you want to connect to a different server, disconnect from the first ALM/QC server, restart Service Test, and connect to the second server.
- Remote Execution is not supported when Web authentication is enabled.

Appendixes

A

Coding Service Test Events

This supplement includes:

Concepts

➤ Writing Code for Events Overview on page 380

Tasks

➤ How to Open a Window for Writing Custom Code on page 382
➤ How to Access Input and Output Properties on page 384
➤ How to Set Input and Output Properties on page 385
➤ How to Use the Logging Function on page 387
➤ How to Use the Report Function on page 388
➤ How to Write Checkpoint Events on page 389
➤ How to Retrieve and Set Test Variables on page 391
➤ How to Encrypt and Decrypt Passwords on page 393
➤ How to Use the OnReceiveResponse Web Service Event on page 394

Reference

➤ Search and Replace for Events User Interface on page 396
➤ Context Methods on page 400
➤ Logging Options on page 401
➤ Assert Options on page 402

Troubleshooting and Limitations - Event Coding on page 402

Concepts

Writing Code for Events Overview

Service Test lets you write customized code for your test in two ways:

- Event Handlers for all activities
- Custom Code activity

For each activity, you can define event handlers for common events, such as: **CodeCheckPointEvent**, **BeforeExecuteStepEvent,** and **AfterExecuteStepEvent**. Certain activity types such as Web Services and HTTP have additional built-in events. For a list of the events, see the "Events View" on page 65.

The **Custom Code** activity allows you to define the behavior of your own activity and its properties. You define this behavior through an event handler that is triggered before, during, or after the execution of the activity.

Service Test's code editor allows you to use autocompletion to prepare the event handler. For task details, see "How to Open a Window for Writing Custom Code" on page 382.

Tip: To instruct Service Test to ignore an event handler, select the handler and press DELETE to remove the name of the handler from the grid.

Casting

The Event Handler code in the **TestUserCode.cs** file uses the following objects:

- **sender.** The object that raises the activity's event.
- **args.** The event arguments passed to the activity.

You can set an activity's properties in the following way:

```
CodeActivity4.Input.a="My string";
```

To access an activity's property, use the **this** object. The following example retrieves an output property from an earlier **CodeAcivity** step:

```
CodeActivity4.Output.secondOutput = " Previous Code Activity Output : " +
this.CodeActivity2.Output.FirstOutpuParameter
```

Tasks

How to Open a Window for Writing Custom Code

For non-custom code activities, you can define default event handlers for checkpoints, before step execution, and after step execution.

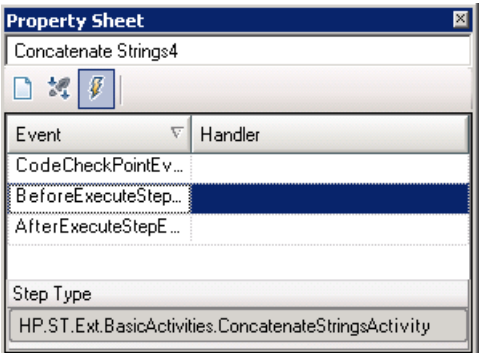
The checkpoint event handlers help you verify the output values in your test. You can use a Report, Assert, or Log function to gather information about your service.

This task includes the following steps:

- "Open the Events view" on page 382
- "Select an event" on page 383
- "Edit the code" on page 383
- "Enable access to the activity's properties" on page 383
- "Save the changes" on page 383

1 Open the Events view

Select an activity title in the canvas and open the **Events** view in the Property Sheet.

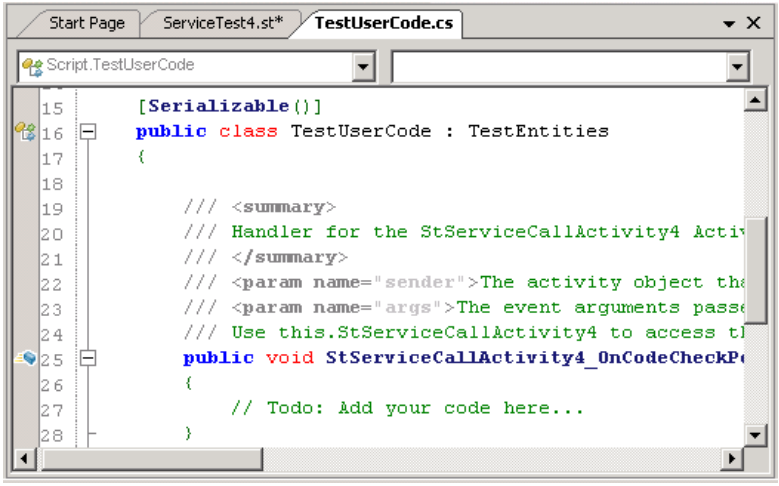


2 Select an event

Select the event for which you want to provide code. Double-click in the **Handler** column. A **TestUserCode.cs** tab opens.

3 Edit the code

Select the **TestUserCode.cs** tab. Locate the **Todo** section and add your custom code. You can use autocompletion to assist you in writing the code.



4 Enable access to the activity's properties

Use the **this** object to access the activity's context, including input and output properties. For example:

```
this.StServiceCallActivity4.InputEnvelope
```

5 Save the changes

Click **File** > **Save All** to save all buffers of the test.

How to Access Input and Output Properties

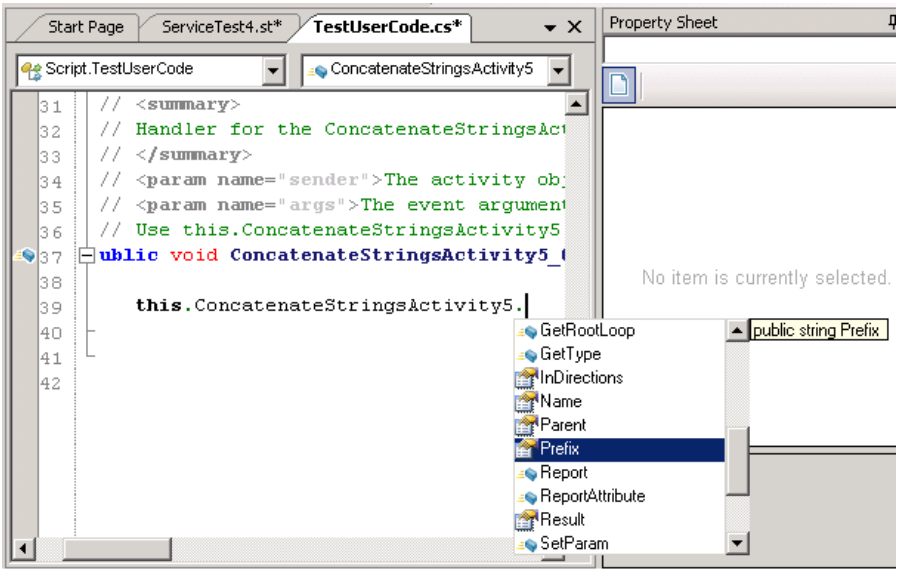
You can set the values of input and output properties through event handlers.

This task includes the following steps:

- "Select a property" on page 384
- "Access the activity's parent" on page 385

1 Select a property

Use autocompletion to locate and select a property from the drop-down list. The drop-down shows all the activity-specific properties. For example, for the **ConcatenateString** activity, the drop-down shows **Prefix** and **Suffix**.



2 Access the activity's parent

To access the activity's parent, select the **Parent** property from the drop-down list. A parent refers to the step or condition that encloses the activity. For example, for a step in a conditional branch, the parent is the name of the condition step.

```
string ParentName = this.ConcatenateStringsActivity5.Parent.Name
```

To obtain the parent of an activity's loop, use a **this** object.

```
this.ConcatenateStringsActivity5.GetParentLoop();
```

How to Set Input and Output Properties

You can set the values of input and output properties using event handlers.

This task includes the following steps:

- "Access the activity's properties" on page 385
- "Select a property" on page 386
- "Access the activity's parent - optional" on page 386

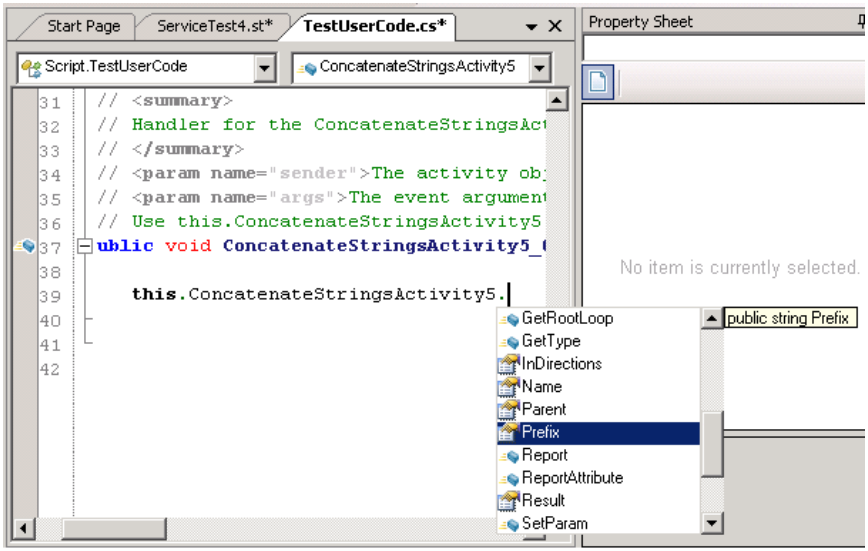
1 Access the activity's properties

Access the activity's properties using the this object. For example:

```
this.ConcatenateStringsActivity4
```

2 Select a property

Use autocompletion to locate and select a property from the drop-down list. The drop-down shows all the activity-specific properties. For example, for a **ConcatenateStrings** activity, the drop-down shows **Prefix** and **Suffix**.



3 Set the activity's property values - optional

Set the property values using hard-coded values or variables that you defined earlier. For example:

```
this.ConcatenateStringsActivity5.Prefix = "hello ";
this.ConcatenateStringsActivity5.Suffix = "world"
```

4 Access the activity's parent - optional

A parent property refers to the step or condition that encloses an activity. To access an activity's parent property, use the **Parent** property. For example, for a step in a conditional branch, the parent is the name of the condition step. To obtain the activity's parent loop, use GetParentLoop.

```
string ParentName = this.ConcatenateStringsActivity5.Parent.Name;
this.ConcatenateStringsActivity5.GetParentLoop();
```

How to Use the Logging Function

The **UserLogger** function generates messages during replay. You can view these messages in the **Output** tab or in the log file.

This task includes the following steps:

- "Select the UserLogger object" on page 387
- "Choose a log type" on page 387
- "Complete the function" on page 388
- "View the Output Log" on page 388

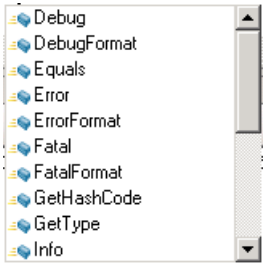
1 Select the UserLogger object

Use autocompletion from the activity's **Context** object, and select the **UserLogger** function.

```
this.StServiceCallActivity4.Context.UserLogger
```

2 Choose a log type

Use autocomplete to choose a log type.



For a list of the log types, see "Logging Options" on page 401.

3 Complete the function

Complete the function as indicated by the tooltip. Provide the necessary arguments.

The following example logs unformatted **Debug** information to the log from a variable called debug_information.

```
this.StServiceCallActivity4.Context.UserLogger.Debug(debug_Information);
```

4 View the Output Log

View the debug information that you instructed Service Test's added, in the **Output** tab or in the log file.

To view the output tab:

a Run the test at least once (F5).

b Select **View** > **Output**. View the **Output** tab in the bottom section of the Service Test window.

To view the output log file:

a Run the test at least once (F5).

b Select **Test** > **Open Script Directory** and open the **Log** directory.

c Open the **vtd_user.log** file.

How to Use the Report Function

Use the **Report** function to show the results of a test's execution in the Run Results Viewer. The report opens automatically after you run the test. You can also view it at any time by selecting **Test** > **View Test Results**.

This task includes the following steps:

➤ "Select the Report object" on page 389

➤ "Complete the function" on page 389

➤ "View the run results" on page 389

1 Select the Report object

Select the **Report** object using autocompletion.

```
this.ConcatenateStringsActivity5.Report
```

2 Complete the function

Provide a keyword and value, based on the function's prototype:
Report(string keyword, string value);

The following example sends an employee's last name to the report. You can also use a variable that was defined earlier in the code.

```
this.ConcatenateStringsActivity5.Report("Employee Name",Jones);
```

3 View the run results

Run the test and view the Run Results. The report displays the keyword and its value in the Iterations node.

How to Write Checkpoint Events

You write checkpoint event handlers using the **checkpoint** object. The Assert library functions run a comparison between actual and expected values, and show the results in the step's **Checkpoint** node. You can also use the **checkpoint** object to control checkpoints in the user interface and customize your reports.

This task includes the following steps:

- "Create a checkpoint handler" on page 390
- "Select the Checkpoint object" on page 390
- "Validate a checkpoint - optional" on page 390
- "Ignore selected checkpoints - optional" on page 391
- "Customize the checkpoints reporting - optional" on page 391
- "View the results" on page 391

1 Create a checkpoint handler

Double-click the **Handler** column of the **CodeCheckPointEvent** row in the activity's **Events** view.

2 Select the Checkpoint object

Use the **args** object, and select the **Checkpoint** object using autocomplete.

```
args.Checkpoint
```

3 Validate a checkpoint - optional

To validate a value during run-time, follow these steps:

a Use the **args.Checkpoint** object, and select **Assert** using autocomplete.

b Use autocomplete to select a comparison operator such as **Equals**, and so forth. For details, see "Assert Options" on page 402.

c Complete the function as indicated by its tooltip. Provide the necessary arguments. For example, to check that the expected and actual values match, use the following expression:

```
args.Checkpoint.Assert.Equals(<Expected Value>, <Actual Value>)
```

The following example checks if the actual value (first argument) equals the expected value (second argument) for the **ConcatenateString** activity. Note that you access the property values using the this.<activity_instance>.<property> object.

```
args.Checkpoint.Assert.Equals(this.ConcatenateStringsActivity4.Prefix+this.Concate
nateStringsActivity4.Suffix, this.ConcatenateStringsActivity4.Result);
```

The second example checks if the text value (alphabetical order for a string) of the prefix is less than the suffix. To ensure that the checkpoint succeeds, enter a prefix value that is greater than the suffix, for example a prefix of **aa** and a suffix of **bb**.

```
args.Checkpoint.Assert.Less("Concatenate test",
this.ConcatenateStringsActivity4.Prefix, this.ConcatenateStringsActivity4.Suffix,
    "The prefix is less than the suffix");
```

4 Ignore selected checkpoints - optional

You can instruct Service Test to ignore the checkpoints selected in the Property Sheet's **Input/Checkpoints** view. This is useful if you need to ignore the checkpoints temporarily.

a Use the **args.Checkpoint** object, and select the **RunUICheckpoints** object using autocomplete.

b Set the value to **false**. You can enable the checkpoints for this step at any time by removing this line or by setting it to **true**.

```
args.Checkpoint.RunUICheckpoints = false;
```

5 Customize the checkpoints reporting - optional

You can instruct Service Test to send a specific message to the Checkpoints node in the report. The standard Report function sends a message to the step's node, but the Checkpoint's Report function allows you to send messages to the Checkpoint sub-node.

a Use the **args.Checkpoint** object, and select the **Report** object using autocomplete.

b Complete the function syntax, based on the function's tooltip, providing the actual and expected result and a logical operator.

6 View the results

Run the test and view the Run Results Viewer. Drill down to the checkpoints and check your results.

How to Retrieve and Set Test Variables

The **Context** object's **TestProfile** method allows you to set or get the value of a user test variable from the active Test Profile.

This task includes the following steps:

➤ "Select the TestProfile object" on page 392

➤ "Get or Set a Variable" on page 392

1 Select the TestProfile object

Use the activity's **Context** object, and select **TestProfile** from the autocomplete menu.

2 Get or Set a variable

From the autocomplete menu, select **GetVariableValue** or **SetVariableValue** methods.

The following example retrieves the value of the *TestName* test variable.

```
string testName =
this.StServiceCallActivity4.Context.TestProfile.GetVariableValue("TestName");
```

How to Encrypt and Decrypt Passwords

Password fields expect an encrypted string—if you provide an unencrypted string, the authentication will fail.

The **EncryptionMngr** method lets you encrypt and decrypt strings within your events.

This task includes the following steps:

- "Encrypt the password" on page 393
- "Decrypt the password - optional" on page 393

1 Encrypt the password

Use the activity's context's **EncryptionMngr** method, and select **Encrypt** from the autocompletion drop-down. The following example encrypts a password.

```
string plainText = "myPassword";
string encryptedText =
this.StServiceCallActivity4.Context.EncryptionMngr.Encrypt(plainText);
```

2 Decrypt the password - optional

Use the **Decrypt** method from the autocompletion drop-down.

The following example decrypts a password and validates it against the original string.

```
string decryptedText =
this.StServiceCallActivity4.Context.EncryptionMngr.Decrypt(encryptedText);
bool equalText = decryptedText.Equals(plainText);
```

How to Use the OnReceiveResponse Web Service Event

Service Test provides several custom event handlers for Web Services. The following example shows how to send the inner SOAP envelope as a string to the execution report, when a response is received.

Note: To manipulate the Output Envelope (**activity.OutputEnvelope** property), use the **AfterExecuteStepEvent**—the **activity.OutputEnvelope** property is not available for the **OnReceiveResponse** event.

For a complete list of the Web Service events, open the Property Sheet's **Events** view for a Web Service's step.

This task includes the following steps:

- "Create an event handler" on page 394
- "Add the Using text" on page 395
- "Define variables for the message" on page 395
- "Define an XmlDocument and load the XML" on page 395
- "Create a new string message" on page 395

1 Create an event handler

Select a Web Service step and double-click the **Handler** column of the **OnReceiveResponse** row in the activity's **Events** view. A code editor containing the new event handler opens.

2 Add the Using text

In the namespace section, make sure you have the **using.System.Text** entry.

```
namespace Script
{
   using System;
   using System.Xml;
   using System.Text;
   using System.Xml.Schema;
   using HP.ST.Ext.BasicActivities;
   …
```

3 Define variables for the message

In the **Todo** section, add two variable declarations as shown below.

```
HP.ST.Ext.WebServicesActivities.ActivityProcessRawMessageEventArgs args)
      {
         var tmp = args.Message;
         var envelope = Encoding.UTF8.GetString(tmp);
    …
```

4 Define an XmlDocument and load the XML

Use autocompletion to select the **XmlDocument** method. Define a name for the envelope and assign it the value of the XML envelope.

```
XmlDocument tmpXdoc = new XmlDocument();
tmpXdoc.LoadXml(envelope);
```

5 Create a new string message

Save the OuterXml content to a string, and apply UTF-8 encoding.

```
var newStrMessage = tmpXdoc.OuterXml;
args.Message = Encoding.UTF8.GetBytes(newStrMessage);
```

Reference

Search and Replace for Events User Interface

You can search for text within your event code or within any file type. You can use standard text, regular expressions, or wildcards.

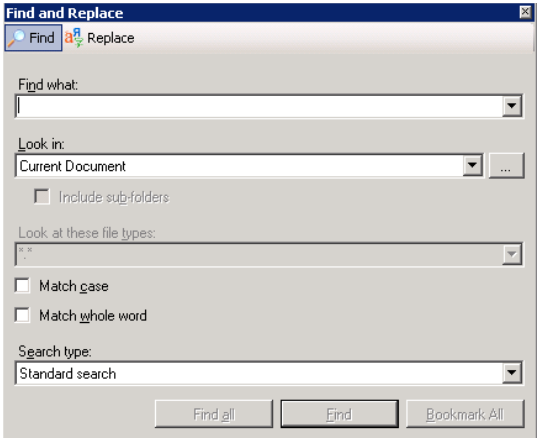
This section includes:

➤ "Find and Replace Dialog Box" on page 397

➤ "Search Results Tab" on page 399

Find and Replace Dialog Box

This dialog box enables you to find and replace strings in the event code.



To access	Do the following: **1** Select a code event tab (*.cs). **2** Select **Edit** > **Find and Replace...** **3** Select an option: ➤ **Find.** Find a string. ➤ **Replace.** Find a string and replace it with the specified text.

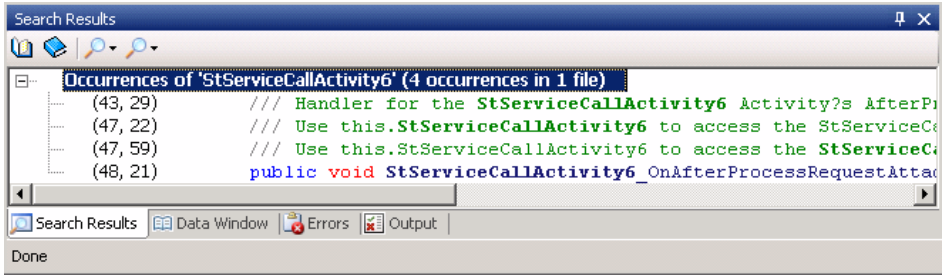
User interface elements are described below:

UI Elements	Description
	Displays the **Find** options for locating a string within the **TestUserCode.cs** file. The following buttons are available: ➤ **Find next.** Finds the next match of the string. ➤ **Find.** Finds the first match of the string and highlights it in the code. ➤ **Bookmark all.** Sets a bookmark for all matches. Service Test inserts the bookmarks notations in the left column of the event code in the **TestUserCode.cs** tab.
	Displays the **Replace** options for replacing a string within the **TestUserCode.cs** file. The following buttons are available: ➤ **Find all.** Finds all matches of the string and lists them in the **Search Results** tab. ➤ **Replace.** Replaces the selected match with the replacement string. ➤ **Replace all.** Replaces all matches with the replacement string.
Find what	The string for which to search.
Replace with	The string to replace the matched string **Note:** Available only when selecting the **Replace** tab.
Look in	The area in which to search: **Current Document, Current Selection, All Open Documents, Whole Project, Whole Solution,** or a folder. To specify a folder, click the **Browse** button and locate the folder. When specifying a folder, you can enable the **Include sub-folders** option.
Look at these file types	The file types to search. To search in all file types, use the default, *.*

UI Elements	Description
Match case	Indicates whether to distinguish between uppercase and lowercase characters during the search.
Match whole word	Searches for occurrences that are whole words, and not part of a larger word.
Search type	The search type: ➤ **Standard search** ➤ **Regular expression** ➤ **Wildcards**

Search Results Tab

This tab lets you select and view the search results. This tab opens when you select **Find all** in the Find and Replace dialog box.



User interface elements are described below:

UI Elements (A-Z)	Description
	Expand All. Expands all nodes of the various searches.
	Collapse All. Collapses all nodes of the various searches.

UI Elements (A-Z)	Description
	Show last search results. Expanding the menu allows you to select a recent search or to clear the history.
	Select Search mode. Allows you to select the representation of the search results. ➤ **Flat.** Lists all results together. ➤ **Per file.** Creates a separate node for each file.

Context Methods

This section describes some of the common **Context** methods.

UI Elements	Description
	Denotes a function.
	Denotes a string.
	Denotes an event.
EncryptionMngr	Allows you to encrypt and decrypt password strings.
TestInputParameters	Allows you to retrieve the test's input parameters.
TestIteration	Provides information about the test iteration.
TestOutputParameters	Allows you to retrieve the test's output parameters.
TestProfile	Allows you to set and retrieve user variables. For details on how to define user variables, see "How to Define Test Properties or User/System Variables" on page 136.

For a complete list of the functions, use the autocompletion feature.

Logging Options

The Service Test API provides the following logging options, available from the **Context.UserLogger** object:

UI Elements	Description
Debug, DebugFormat	Logs debug information collected during the test run.
Error, ErrorFormat	Logs errors that occurred during the test run.
Fatal, FatalFormat	Logs fatal errors that occurred during the test run.
Info, InfoFormat	Logs informational data from the test run.
Warn, WarnFormat	Logs warnings that were issued during the test run.

The Format option allows you to provide values based on the list of items. The following example logs formatted informational messages, using the **FirstName** variable's value for the first item, and the **LastName** variable for the second item, and so forth.

```
this.<activity>.Context.UserLogger.InfoFormat("Hello: {0}{1}", FirstName, LastName);
```

For task details, see "How to Use the Logging Function" on page 387.

Assert Options

The Assert object provides the following comparison operators:

UI Elements
Equals
Greater
GreaterOrEqual
Less
LessOrEqual
NotEquals

For task details, see "How to Write Checkpoint Events" on page 389.

Troubleshooting and Limitations - Event Coding

This section describes troubleshooting and limitations for working with Service Test events.

➤ The **Loop.CurrentIterationObject** is deprecated and will always return null.

B

Extensibility in Service Test

The HP Service Test installation includes an SDK (Software Development Kit) for adding new activities to the Toolbox palette. After defining a custom activity, you can drag it into the HP Service Test canvas as you would with any other built-in activity.

Disclaimer: This is a preliminary version of the SDK. It enables you to extend the capabilities of HP Service Test. However, this SDK is subject to change in a future release, and these changes might require you to update any code that uses this preliminary version. Although HP endeavors to keep these changes to a minimum, we cannot guarantee that extensions created using the preliminary version of the SDK will continue to work without modification when upgrading to a new version of HP Service Test.

This supplement includes:

Concepts

- Creating New Activities - Overview on page 404
- Runtime Files on page 405
- Signature Files on page 411
- Addin Files on page 421
- Resource Files on page 424

Tasks

- How to Create a New Custom Activity on page 425
- How to Debug an Activity on page 430

Concepts

Creating New Activities - Overview

This section describes the structure of the files required to define a new activity in order to extend the capabilities of Service Test.

Once you create a new activity through this mechanism, it will be available in the Toolbox for all future tests.

To create a custom activity, you need to define the following files on all machines upon which you intend to run the test.

- Runtime Files
- Signature Files
- Addin Files

The **Runtime** file is the DLL that Service Test invokes to run the activity. For details, see "Runtime Files" on page 405.

The **Signature** file is an XML file that defines the input and output properties, events, and the runtime class that executes the activity. For details, see "Signature Files" on page 411.

The **Addin** file is an XML file that references all of the activity component data. For details, see "Addin Files" on page 421.

In addition, you can also define resource files to store text strings used by your activity. For details, see "Resource Files" on page 424.

The Service Test installation includes a sample project in the product's **ExtensibilitySamples** folder. Use this sample as a basis for a new activity.

All of the custom files—Signature, Addin, and Runtime—should be stored in the <*Service Test installation*>**addins****CustomerAddins**\<*addin_name*> folder. This allows Service Test to load them during startup.

For more details, see "How to Create a New Custom Activity" on page 425.

Runtime Files

In addition to the signature and addin files, you must provide a DLL to run when executing the activity. You create a C sharp solution and customize the code as required. You can use the sample **ReportMessageActivitySample.sln** provided with Service Test as a basis for your project.

The following example shows source code from the **ReportMessageActivitySample.cs** file.

This section contains the following topics:

- Using Statements
- Namespace and Class
- Internal Logging
- Initialize the Properties
- Retrieve the Property Values
- Define Events
- Execute the Step
- Set the Status

Using Statements

First, provide the mandatory **using** statements:

```
using HP.ST.Fwk.RunTimeFWK;
using HP.ST.Fwk.RunTimeFWK.Utilities;
using HP.ST.Shared.Utilities;

// If you need to implement Internal Logging
using log4net;
```

In your solution, you must also add a reference to the DLLs. The DLLs are located in the products installation's **bin** folder. You must always add a reference to **HP.ST.Fwk.RunTimeFWK.dll** and **HP.ST.Shared.Utilities.dll**. If you are using internal logging, you must also add a reference to **log4net.dll**.

Namespace and Class

Next you need define the namespace and the activity's runtime code. The class you define for your custom activity must inherit from the **STActivityBase** class.

```
namespace ReportMessageActivitySample
{
   [Serializable]
   public class ReportMessageActivitySample : STActivityBase
   {
```

Internal Logging

Use Service Test's built-in logger manager to instruct the activity to create an internal log during runtime. This example gets the property values of the input properties and sends the output to either the Run Results Viewer only or to the Run Results Viewer and Output window.

```
/// <summary> /// Internal log
      /// </summary>
      private static readonly ILog log =
        LogManager.GetLogger(typeof(ReportMessageActivitySample));
      const string runResults = "Run Results";
      const string runResultsAndOutputWindow = "Run Results and Output Window";
```

For details about other logging options, see "How to Use the Logging Function" on page 387.

Initialize the Properties

Initialize the custom Input and Output properties that you define in the signature file. The following example initializes the three input properties, **Status**, **Message**, and **Destination**.

```
/// <summary>
    /// Initializes properties.
    /// </summary>
    /// <param name="ctx">The runtime context</param>
    public ReportMessageActivitySample(ISTRunTimeContext ctx, string name)
        : base(ctx, name)
    {
        this.Status = String.Empty;
        this.Message = String.Empty;
        this.Destination = String.Empty;
    }
```

Retrieve the Property Values

This section retrieves or sets the input property values.

```
/// <summary>
    /// Gets or sets the status of the message to report.
    /// </summary>
    public string Status { get; set; }

    /// <summary>
    /// Gets or sets the details of the message to report.
    /// </summary>
    public string Message { get; set; }

    /// <summary>
    /// Gets or sets the destination where the message should be reported to.
    /// </summary>
    public string Destination { get; set; }
```

If you have array type properties that are not described by a schema, for example, key/value pairs, you must initialize all the members of the array explicitly, and indicate the actual number of elements.

The following example initializes 40 elements for the MyArrayName property. It contains 40 key and value pairs.

```
this. MyArrayName = new MyPair[40];
        for (int i=0; i<40; i++)
        {
            this. MyArrayName [i] = new MyPair();
        }

public MyPair[] MyArrayName;
      public class MyPair
      {
         string Key;
         string Value;
      }
```

For arrays defined by a schema or WSDL, you can use the standard Select Link Source dialog box and link directly to the array element.

Define Events

Define any custom events. Later, you invoke the event.

```
public event EventHandler CustomerEvent;
private void InvokeCustomerEvent(EventArgs MyArg)
      {
         EventHandler handler = this.CustomerEvent;
         if (handler != null)
         {
            handler(this, MyArg);
         }
      }
```

Execute the Step

Execute the step and send the runtime information to the log. You use the **STExecutionResult** data type and its **ExecuteStep** function defined in the **STActivityBase** class.

```
protected override STExecutionResult ExecuteStep()
      {
         string DetailsReport;

         if (this.Destination == runResultsAndOutputWindow)
         {
            LogInfo("\n" + this.Message.Replace("\\n", "\n"));
         }
/// <summary>
       /// Reports message to test results and output window.
       /// </summary>
// The line-breaks replacements allow the printing of multiple lines in the report
      DetailsReport = this.Message;
      DetailsReport = DetailsReport.Replace("\\n", "<BR>");
      DetailsReport = DetailsReport.Replace("\n", "<BR>");
      this.Report("Message", DetailsReport);
```

If you defined a custom event, invoke it after the call to **ExecuteStep**.

```
…
protected override STExecutionResult ExecuteStep()
{
InvokeCustomerEvent();
}
```

Set the Status

Set the Status of the test run. The **ReportMessageActivitySample.cs** sample file uses enumeration to set the status, based on the **STExecutionResult** value.

```
 switch (this.Status)
          {
case "Done":
                    this.Report(ReportKeywords.StatusKeywordTag,
ReportKeywords.DoneValueTag);
                    return new STExecutionResult(STExecutionStatus.Success);
case "Pass":
                    this.Report(ReportKeywords.StatusKeywordTag,
ReportKeywords.SuccessValueTag);
                    return new STExecutionResult(STExecutionStatus.Success);
case "Fail":
                    this.Report(ReportKeywords.StatusKeywordTag,
ReportKeywords.FailureValueTag);
                    return new STExecutionResult(STExecutionStatus.Success);
default:
                    return new STExecutionResult(STExecutionStatus.Success);
          }
```

Customizing and Compiling

After you customize the code, you compile the DLL. The DLL name should be the same as the name of the addin file. For example, the runtime file, **ReportMessageActivitySample.dll** corresponds to the **ReportMessageActivitySample.addin** file.

After you create the runtime file in your development environment, you copy it into the activity's directory and reference the DLL from the addin file.

Signature Files

The signature file describes the activity to Service Test. It typically has a Resource element followed by the following sections: **GeneralProperties**, **InputProperties**, **OutputProperties**, **Tabs**, and **Events**. The signature file must have an **.xml** extension.

This following sections describe the:

➤ "Resource Element" on page 411

➤ "Section Element" on page 413

➤ "Property Definitions" on page 417

Resource Element

The **Resource** element has the following attributes:

Attribute	Description
type	The type of entity, in this case **Activity**.
id	A unique string identifying the activity. You can reference this ID when writing event handler code. For details, see the section on API coding.
version	The version of the current addin mechanism. For example, 1.0.0
group	The parent group under which the activity will appear in the Toolbox palette, for example **String Manipulation**. To add it to an existing group, specify a group name as it appears in the Toolbox. If the group does not exist, Service Test adds a new group.
shortname	The short name of the activity as shown in the Toolbox hint area (above the description).
description	The description of the activity as shown in the Toolbox hint area.
assembly	The DLL file to call when running the activity, stored in the same folder as the signature and addin files.

Attribute	Description
className	The class implemented by the activity. **Note:** The class must inherit from the STActivityBase class.
image	The png image file for the icon representing the activity. Store the image in the same folder as the signature file.
visible	A boolean value indicating whether to display the activity is displayed in the Toolbox palette.
xmlns	The namespace that defines the schema for the signature file, http://hp.vtd.schemas/signature/v1.0. Keep the default value.
xmlns:xsi	The schema instance used for the signature file. Keep the default value, http://www.w3.org/2001/XMLSchema-instance.
xmlns:Location	The URL of the signature file's schema, **Signature.xsd**, referenced by the namespace. This value has two parts: the namespace and the file path. ➤ **Namespace**. http://hp.vtd.schemas/signature/v1.0 ➤ **File path**. <*Installation_Folder*>/dat/schemas/Signature.xsd Keep the default value.

The following example shows the **Resource** section from the signature file of the sample activity, **ReportMessageActivitySample**. The location of the sample is:
<*Installation_Folder*>\ExtensibilitySamples\ReportMessageActivitySample

```
<Resource
  type="Activity"
  id="ReportMessageActivitySample"
  version="1.0.0"
  group="Miscellaneous"
  shortName="ReportMessageActivitySample"
  description="The ReportMessageActivitySample allows you to send a custom
message to the report and/or log. "
  assembly="ReportMessageActivitySample.dll"
  className="ReportMessageActivitySample.ReportMessageActivitySample"
  image="toolbox_ReportMessageActivitySample.png"
  visible="true"
  xmlns="http://hp.st.schemas/signature/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://hp.st.schemas/signature/v1.0 ../../../dat/schemas/
Signature.xsd"
  >
```

Section Element

The Section definitions apply to all sections in the Signature file, such as **GeneralProperties**, **InputProperties**, and **OutputProperties**. The following table describes the **Section** element attributes.

Attribute	Description
name	The internal name of the section. To use a sub-element, it is recommended that you set the value to the name of the sub-element. For example name="Tab" or name="Alerts".
source	If set to **true**, it displays the source of the section.
dest	**destination**. If set to **true**, it displays the destination path of the section.
checkpoint	If set to **true**, displays the checkpoint check box in the **Validate** column.

Attribute	Description
isSharedMetadata	When **true**, allows the section to share its meta data with other sections.
propertiesType	The type of the properties in the section, for example, "XML".
showXmlControls	When **true**, displays the XML controls such as Text and XPath tabs in the section.
displayName	The name of the section as it will appear in the Property Sheet.

The following table describes the sub-elements of **Section**:

Element	Description
Tab	The tabs to appear in the Property Sheet using the following attributes. For details, see below. ➤ **name**. The internal name of the tab. Some of the built-in ones are General, InputOutput, Events, Attachments, and SOAPFault. ➤ **id.** The id of the tab referred to be the API. The id usually uses the name with an added suffix, "Tab". For example, GeneralTab, InputOutputTab, and EventsTab. ➤ **CanBeInToolbox.** If true, shows the tab on the Toolbox's toolbar. ➤ **CanBeInPropertySheet.** If true, shows the tab in the Property Sheet. ➤ **CanBeInDataLinkDialog.** If true, shows the tab in the Select Link Source dialog box. **Note:** To use the default tabs: **General**, **Input**/**Checkpoints**, and **Events**, you do not need to include this element. If you want to omit one of the tabs or add extra ones, then you need to include the Tabs sub-element and specify the desired tabs. For details about the tabs, see "The Property Sheet" on page 27.

Element	Description
Alert	Allows you to apply alerts to the properties in the section using the following attributes: ➤ **constraint.** The reason to show the alert, for example, NullValueConstraint. ➤ **target.** The Xpath of the property to which to apply the constraint. ➤ **section.** The internal name of the section containing the properties. ➤ **type.** The type of alert, such as error or warning The value of the element, is the alert message. For example: <Section name="Alerts" isSharedMetaData="true"> <Alert constraint="NullValueConstraint" target="/Url[1]" section="InputProperties" type="error">The 'URL' must not be empty</Alert> </Section>

Element	Description
Events	In the **Events** sub-element, you can customize the events that will be available for the activity. This sub-element uses the following attributes ➤ **name.** The internal name of the event. Use one of the built-in names or define a custom one. ➤ **CodeCheckpointEvent**. Allows you to create an event handler to run when the test is verifying checkpoints. ➤ **BeforeExecuteStepEvent.** Allows you to create an event handler to run before executing the activity. ➤ **AfterExecuteStepEvent.** Allows you to create an event to run after executing the activity. ➤ **<custom event>.** A custom event that you define. ➤ **description.** A textual description of the event. ➤ **eventArgs.** The source of the arguments for the event. The standard argument for **BeforeExecuteStepEvent** and **AfterExecuteStepEvent** event is STActivityBaseEventArgs. The built-in value for the **CodeCheckpointEvent** is CheckpointEventArgs. Web Service steps provide additional built-in events. For details, see "Defining Services" on page 147. **Note:** To access the default events: **CodeCheckpoint**, **BeforeExecute**, and **AfterExecute**, you need to include only the **Events** tab in the **Tab** sub-element, but you do not need to use the **Events** sub-element. If you want to omit one of the events or add custom events, then you need to include this sub-element and specify the desired events.

Property Definitions

The property definitions in the signature file, apply to all sections that use properties. The built-in sections that use properties are:

- **GeneralProperties.** Defines the properties in the **General** view of the Property Sheet, for example **Step ID** and **Name**.
- **InputProperties.** Defines the input properties located in the Input pane of the of the Property Sheet's **Input/Checkpoints** view.
- **OutputProperties**. Defines the output properties located in the **Checkpoints** pane of the of the Property Sheet's **Input/Checkpoints** view.

This section includes:

- "Elements and Sub-Elements" on page 418
- "Element Attributes" on page 419
- "Simple Elements with Enumeration" on page 420
- "Complex Array Elements" on page 420

Elements and Sub-Elements

The elements and attributes are defined in the standard XML schema file, http://www.w3.org/2001/XMLSchema, or Service Test's **types.xsd**, located in the <*Installation_Folder*>/dat/schema folder. The following table describes the elements and sub-elements that can be used in these sections. The level number indicates the level of the element or sub-element in the hierarchy.

Element	Description
xs:schema	The schema namespaces for the properties, as described by the **xml:ns** attribute. Keep the default values, http://hp.vtd.schemas/types/v1.0 and http://www.w3.org/2001/XMLSchema.
xs:import	The namespace to import using the **namespace** and **schemaLocation** attributes. Keep the default values, http://hp.vtd.schemas/types/v1.0 and ../../../dat/schemas/types.xsd.
xs:element	The element to define, using the attributes described in the table below.
xs:simpleType	A tag indicating the beginning of definitions of a simple type property. **Note:** You only need to enclose a simple type element with this tag, if you want to do enumeration with a drop-down list. For example, the following definition does not require an **xs:simpleType** tag. <xs:element name="ClientCertificate" type="types:Certificate" types:displayName="Client certificate" />

Element	Description
xs:complexType	A tag indicating the beginning of definitions for a node of multiple properties.
xs:sequence	A tag indicating the beginning of a list of properties in a complex type property.
xs:restriction	A tag restricting the value of the enumeration values of a property, using the **base** attribute. To restrict the value to a string, use base="xs:string".
xs:enumeration	A tag indicating the beginning of list of values in the drop-down menu for a property, using the **value** attribute.
xs:annotation	An annotation for the element Use an **xs:documentation** sub-element to compose text that will appear below the properties grid in the Property Sheet.

Element Attributes

The following table describes the primary attributes of the **xs:element**. For attributes in the standard XML schema, use an **xs:** prefix in the value, for example standard types use type=xs:string or type=xs:int.

For types defined in Service Test's **Types.xsd** schema, use a **types:** prefix in the attribute name. For example types:displayName.

Attribute	Description
name	The name of the property or grid in the Property Sheet. This is the name referenced by other calls and by the event handlers code. This is not the display name in the Property Sheet.
type	The type of property. Some common values are: xs:string, xs:int, xs:boolean, Multipart, Header, Part. For a value defined in the Types schema, use the **types**: prefix. For example type="types:filePath".
minOccurs	The minimum number of array elements for which the user must provide. For none, specify "**0**".

Attribute	Description
maxOccurs	The maximum number of array elements the user may provide. To allow an unlimited amount, specify "**unbounded**"
types:visible	When **true**, allows the parameter to be visible even before being expanded by the Add Array Element command.
types:argType	The type of the property: "XML" or "Object".
types:displayName	The property name as it will appear in the Property Sheet.

Simple Elements with Enumeration

The following **ReportMessageActivitySample** example defines an input parameter, **Status,** with an enumeration attribute. This code creates a drop-down list of values in the Property Sheet's input property grid.

```
<xs:element name="Status" default="Done" types:displayName="Status">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="Done"/>
              <xs:enumeration value="Pass"/>
              <xs:enumeration value="Fail"/>
            </xs:restriction>
          </xs:simpleType>
</xs:element>
```

Complex Array Elements

The following sample defines a complex property, with a Key and Value pair of values.

```
<xs:complexType name="NameValueType">
  <xs:sequence>
    <xs:element name="Key" type="xs:string"/>
    <xs:element name="Value" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Addin Files

The **addin** file provides the references for the activity you are defining. The file is in XML format and contains information such as activity names, dependencies, and runtime DLLs.

The addin file should be located in the installation directory under the **addins\CustomerAddins**<*addin_name*> folder, together with the signature file. The addin file must have an .addin extension.

Each addin file should contain the following sections:

- "Addin Section" on page 421
- "Manifest Section" on page 422
- "Runtime Section" on page 423

Addin Section

The Addin's attributes describe the activity to Service Test.

Element	Description	Attributes
<Addin>	Basic details about the addin.	➤ **name.** the name of the addin, for future use. ➤ **author.** the creator of the activity. ➤ **copyright.** the full path of a text file with the copyright information. ➤ **description.** a textual description of the activity. ➤ **version**. The addin file version, set to 1.0.

Element	Description	Attributes
Path	Details about the location of the activity.	➤ **name.** The logical path scanned by the framework, to identify addins. The physical location of this folder is addins\CustomerAddins\ <addin_name>.
Activity (sub-element of Path)	See attribute descriptions	➤ **id.** An identifying string corresponding to the ID in the signature file. ➤ **displayName.** The activity's display name in the Toolbox. ➤ **signatureFile.** The name of the XML signature file.

Manifest Section

The Manifest section contains a unique logical name for the addin and provides a list of dependencies.

Element	Description	Attributes
Identity	Basic details about the addin.	➤ **name.** The activity name corresponding to the **ID** in the signature file. When referring to this addin as a dependency, use this name.
Dependency	The activity upon which the current activity is dependent	➤ **addin**. The identity name of the dependent activity, from the name attribute in the **Manifest** section of its addin file. ➤ **requirePreload**. A boolean value indicating whether to preload the dependent addin before loading the current one.

Runtime Section

The Runtime section contains information about the addin's runtime file.

Element	Description	Attributes
Import	An assembly to import when running the activity.	➤ **assembly**. The name of an assembly. Use the DLL name without the DLL extension. **Note:** To import an addin from another activity, precede the addin name with a colon. For example, :HP.ST.Fwk.DesignerModel imports the DesignerModel addin.

The following example shows the **ReportMessageActivitySample.addin** file. For multiple activities, use unique **Addin** files.

```
<?xml version="1.0" encoding="utf-8"?>
<AddIn name        = "HP Report Message Activity Sample"
     author      = "John Doe"
     copyright   = "C:\Copyrights\copyright.txt"
     description = "Extensibility Sample - Report Message Activity"
     version="1.0">

  <Manifest>
    <!--<Must be unique -->
    <Identity name = "ReportMessageActivitySample"/>
  </Manifest>

  <Runtime>
    <Import assembly=":HP.ST.Fwk.DesignerModel"/>
  </Runtime>

<Path name = "/ST/Activities">
    <!--Misc Activities -->
    <Activity id    = "ReportMessageActivitySample"
          displayName   = "ReportMessageSample"
          signatureFile = "ReportMessageActivitySample.xml"
          assembly="ReportMessageActivitySample.dll"/>
  </Path>
</AddIn>
```

Resource Files

You can use resource files to retrieve values for elements and attributes. The **fromResource** function lets you name the resource containing the values.

In the following example, the signature file retrieves the **shortName** and **description** from a resource file.

```
<Resource
  type="Activity"
  id="ReportMessageActivitySample"
  version="1.0.0"
  group="Miscellaneous"
  shortName="fromResource(conc_str_short_name)"
  description="fromResource(conc_str_description)"
```

The resources are defined in a standard Microsoft ResX Schema version 2.0 Resource fie.

```
<data name="conc_str_description" xml:space="preserve">
    <value>Reports an activity's run status to the log</value>
  </data>
  <data name="conc_str_short_name" xml:space="preserve">
    <value>Report Message</value>
  </data>
```

The resource reference must be a compiled file with a **.resources** extension, compiled from the resx source file and stored in the same folder as the signature file.

You can generate the compiled file as a post-build operation using the **resgen** utility. For example:

resgen STBasicActivity.resx STBasicActivity.resources.

For step-by-step instructions on how to create a custom activity, see "Create and compile a runtime DLL" on page 425.

Tasks

How to Create a New Custom Activity

This task describes how to create a new activity and implement into Service Test.

To run a test with the custom activity on another machine, you need to copy all of the custom files to its <*Service Test installation*>**addins****CustomerAddins**\<*addin_name*> folder.

This task includes the following steps:

- "Create and compile a runtime DLL" on page 425
- "Create a signature file" on page 426
- "Create an addin file" on page 428
- "Provide a graphic for your activity - optional" on page 429
- "Check the implementation" on page 429

1 Create and compile a runtime DLL

a Create a C# project that implements your activity's actions. You can use the sample project file in the **<installation folder>\ExtensibilitySamples** folder as a basis.

b Initialize and define the properties and add invocations to custom events. For more details, described in "Runtime Files" on page 405.

c Send information to the report using as shown in the sample file.

d Declare statuses for the test.

e After creating the code, compile the DLL using the name of the activity. The compiled file should be located in the installation directory under the **addins****CustomerAddins**\<*addin_name*> folder.

2 Create a signature file

a Create a new signature file with an **.xml** extension. in the **addins\CustomerAddins**<*addin_name*> folder, together with the runtime file. Use the sample project in the <*installation folder*>**\ExtensibilitySamples** folder as a basis for your custom signature file.

b Customize the **Resource** section or copy the code provided below, modifying the bolded text for your needs. For information about each of the elements, see "Resource Element" on page 411.

```
<Resource
  type="Activity"
  id="ReportMessageActivitySample"
  version="1.0.0"
  group="Miscellaneous"
  shortName="ReportMessageActivitySample"
  description="ReportMessageActivitySample allows you to send a custom
message to the report and/or log. "
  assembly="ReportMessageActivitySample.dll"
  className="ReportMessageActivitySample.ReportMessageActivitySample"
  image="toolbox_ReportMessageActivitySample.png"
  visible="true"
  xmlns="http://hp.st.schemas/signature/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://hp.st.schemas/signature/v1.0 ../../../dat/schemas/
Signature.xsd"
  >
```

c Add the required sections, such as GeneralProperties, InputProperties, Tabs, Events and so forth. For a list of the built-in sections and their attributes, see the "Section Element" on page 413.

d Add properties to the relevant sections. For details and examples, see "Property Definitions" on page 417.

- **GeneralProperties**. Properties displayed in the Property Sheet's **General** tab. In most cases you can use the section as it appears in the sample file, without any modifications. By default, it will provide the **Step ID** and **Name** properties.
- **InputProperties**. Properties displayed in the Property Sheet's **Input/ Checkpoints** tab, in the **Input** pane.

➤ **OutputProperties**. Properties displayed in the Property Sheet's **Input/Checkpoints** tab, in the **Checkpoints** pane.

e Specify any external resource files as described in "Resource Files" on page 424.

f Close the file with the </Resource> tag.

```
</Resource>
```

For more details about the structure of the signature file, see "Signature Files" on page 411.

3 Create an addin file

a Create a new file with an **.addin** extension in the <*installation directory*>**addins****CustomerAddins**\<*addin_name*> folder, together with the signature file.

b Use the sample addin file in the **<installation folder>\ExtensibilitySamples** folder as a basis, or copy the code provided below, modifying the bolded text for your needs. For details, see "Addin Files" on page 421.

```
<?xml version="1.0" encoding="utf-8"?>
<AddIn name        = "HP Report Message Activity Sample"
       author      = "John Doe"
       copyright   = "prj:///doc/copyright.txt"
       description = "Extensibility Sample - Report Message Activity"
          version="1.0">
<Manifest>
   <!--<Must be unique -->
   <Identity name = "ReportMessageActivitySample"/>
 </Manifest>

 <Runtime>
  <Import assembly=":HP.ST.Fwk.DesignerModel"/>
 </Runtime>

 <Path name = "/ST/Activities">
  <!--Misc Activities -->
  <Activity id    = "ReportMessageActivitySample"
   displayName   = "ReportMessageSample"
    signatureFile = "ReportMessageActivitySample.xml"
   assembly="ReportMessageActivitySample.dll"/>
 </Path>
</AddIn>
```

c Create a unique **Addin** file for each activity—do not define multiple activities in a single **Addin** file. For details, see "Addin Files" on page 421.

d Define post-build tasks such as **resgen**, as described in "Resource Files" on page 424.

e Compile the project and copy the DLL to the <*Service Test installation*>**addins****CustomerAddins**\<*addin_name*> folder.

For additional details, see "Runtime Files" on page 405.

4 Provide a graphic for your activity - optional

a Copy an icon image for your activity into the *<Service Test installation>***addins****CustomerAddins***<addin_name>* folder. This file should meet the following requirements:

- a **.png** extension
- sized at 16 x 16 pixels
- 8-bit color depth

b Specify the name of the image file in the signature file's Resource Element.

5 Check the implementation

a Reopen Service Test and drag the new activity into the Test Flow. Verify that the activity and its properties appear as expected.

b Provide property values.

c Run the test and observe the Output log and Run Results Viewer.

d Enable checkpoints to verify the results and rerun the test.

How to Debug an Activity

This task describes how to debug a custom or built-in activity.

1 Open the source code and add a breakpoint

a In an existing test, select **File** > **Open** > **File**. Locate the source file of the solution, for example a **.cs** file. The file opens in Service Test.

b Use the F7 key to add a breakpoint in the **ExecuteStep()** function.

2 Run the test

a Run the test.

b Use Service Test's **Debug** menu commands such as **Step Into**, to execute the rest of the test.

Index

A

absolute path 131
activities
  adding to the test 33
  creating custom 403
  Database 99
  Date/Time 77, 81
  date/time 98
  file system 96, 98
  flow control 95
  FTP 103
  in toolbox 23
  Java 109
  JMS 110
  load testing 113
  math 97
  network 106
  overview 76
  string manipulation 97
  Transform XML 86
  unified functional testing 114
  Validate XML 84
  Web Services 115
  XML 114
Add New DB Data Query wizard 313
Add New Test dialog box 142
add-in file, extensibility 421
Advanced Settings dialog box (security) 229
advanced settings, security scenario 191
alerts 20
algorithm, for encryption 232
ALM/QC
  connecting to 370
  integration 365
  managing Vuser scripts 366
  reporting defects 329
  version control 367
API 379
Assert function 389
attach data source 304, 309
attachments
  adding 34
  view in Property Sheet 66
authentication
  certificate 190
  WSDL import 167

B

binary encoding 230
BPT
  benefits 129
  overview 128
build configurations 240
business component
  see BPT 135
business process component 134
Business Process Testing
  limitations in Service Test 130

C

Call Java Class Settings dialog box 124
Captured Data pane 345
casting 380
certificate authentication 190
checkpoints 28
  configuring 243
  HTTP 325
  keywords 275
  overview 241
  XPath 244

child relations
defining 274
dialog box 312
class, Java types 27
coding events 379
collapse all 46
Connection Builder dialog box 117
Context object
TestProfile 391
UserLogger 387
control flow 19
copy and paste 19
Create New Test dialog box 140
custom activities 403
Custom Binding scenario 214, 221
custom binding scenario 187
custom fields 325

D

data
access 287
assign 283
child relations 274
create data sources 276
defining 262
from data driving 269
from data source 264
keywords 274
navigation 272, 290
outgoing links 268
populating manually 263
data driving
data 269
settings 296
data navigation 309
Data Navigation dialog box 290
data query
Excel 299
data relation 312
data source
attaching 304, 309
child 312
data sources 24, 264
unique 271
Data Sources view 68
Data Window tab 294
Data/Time activities 77, 81
database
integration 313
retrieving data from 266
Database activities 99
Date/Time activities 98
Debug configuration 240
Debug menu 255
debugging window 257
decryption 393
defects, reporting 329
from Run Results 329
digital signatures 186
DLL files 405
do while loop 95
documentation, online 13

E

Edit Elements dialog box 301
encoding, for WS config file 230
encrypted data for Web Services security 186
encryption 393
end transaction 242
endpoint address 27
entropy mode 232
error messages 72
Errors tab 72
event handler
assert function 389
checkpoints 36
coding overview 380
ignoring 380
Report function 388
writing 379
event handlers
data access 287
Excel file
add query 299
as a data source 24
EXISTS keyword 274
expand all 46
Export Run Results dialog box 351

extensibility 24, 403
runtime files 405
signatures 411

F

Federation security scenario
details 223
with STS 204
File System activities 96, 98
find 397
flow control 19
Flow Control activities 95
for loop 95
foreign key 274
FTP activities 103

H

handler event
OnReceiveResponse 394
HTTP Binding 225
HTTP checkpoints 325
HTTP Multipart attachments 80
HTTP multipart input 108

I

Import Service from URL or UDDI dialog box 167
importing services
file system 165
from URL 167
Installation Guide 13
IsOneWay 27

J

Java activities 109
Java settings 124
Java testing 26
Java types, support for 27
Java virtual machine 52
JMS activities 110
JMS test settings 53
JNDI settings 53
JVM settings 52

K

keywords 274

L

load testing
preparing test 247
think time 242
transactions 242
load testing activities 113
LoadRunner integration 31
local data table 24
loop settings 95

M

Math activities 97
message signatures 186
messages 72
messages, sending to report 323
movies, Service Test features 13
MTOM 230
mutual certificates 206

N

namedPipe 204
navigating data 309
navigating in data 272
navigation properties 290
negative testing, Web Services 151
netTcp 204
nettcp 206
Network activities 106
NIL 274
NOT_FOUND 274

O

on 192
online documentation 13
online help 13
outgoing links 268
Output tabs 71, 257

P

Print dialog box, Run Results Viewer 356
Print Preview dialog box (Run Results Viewer) 358
profiles 136, 138
properties
  selecting data source 305
Protection Level 232

Q

query
  data 299
Query Builder dialog box 118
Query Designer 120
QuickTest Professional integration
  adding steps 88
  results 319
  testing 31

R

Readme 13
realm, Web Service security 234
relative path 131
Release configuration 240
Reliable Messaging 230
replay, preparing for 182
Report function 388
Report Message activity 322
resource files 424
REST service testing 25
results See run results
results, of search 399
Run dialog box 253
run results
  custom fields 325
  display 319
  exporting to a file 351
  filtering 328
  finding 328
  previewing before printing 358
  printing 356
  reporting defects manually 329
  Run Results window 332
  tree 337
Run Results Viewer
  standalone installation 318
Run Results window, toolbar 334
running tests 239
runtime files 405

S

save, save as command 39
scenario setting, advanced 191
script generator, custom
  implement 425, 430
  overview 404
search in files 397
Search Results tab 399
security
  port 182
  Web Service scenario 184
security elements, order 197
security scenarios
  Custom Binding 214, 221
  Federation 223
  HTTP Binding 225
  user interface 212
security settings, advanced 229
security, examples for Web Services 198
Select Certificate dialog box 235
Select Link Source dialog box 305
Select Service dialog box 165
Select Service from UDDI 168
send request to service 57
Service Test
  introducing 17
  main window 38
  working with 15, 75
services
  importing 167
  importing from file system 165
  references 148, 149
  testing 17
signature file 411
signatures, Web Service messages 186
SKIP 274
SOAP fault view 61
SOAP faults 151
SOAP request, manual 148, 149

software updates 13
SSL, testing Web Service 207
standalone installation, run results 318
start transaction 242
Step 194
String Manipulation activities 97

T

TCP transport 206
test results, see run results
Test Settings view 52
test variables 391
    defining 136, 138
Test Variables view 62
tests
    creating 127
    overview 128
    See also run results
Tests pane 43
think time 242
Toolbox 23
toolbox 93
transactions 242
transforming XML 86
transport
    customizing HTTP 234
    JMS 31

U

UDDI, importing services from 168
UFT activities 114
UFT testing 31
    how to implement 88
    results 319
unified functional testing activities 114
unique data sources 271
Update Step wizard 170, 173
updates 13
user variables 136, 138
UserLogger 387
Username Authentication (Message Protection) 190

V

validating XML 84
validation, via checkpoints 241
variables, test 391
version control 367
views
    Attachments 66
    Data Query Properties 68, 69
    Data Sources 68
    Events 65
    General 54
    HTTP 63
    Input/Checkpoints 55
    Multipart 64
    REST HTTP Input 63
    Security Settings 67
    SOAP fault 61
    Test variables 62
virtual machine settings 52
VM settings 52
VuGen integration
    adding steps 88
    calling a script 54
    UFT testing 31

W

W3ES security 206
warnings 72
WCF Custom Binding 214, 221
WCF Federation 223
WCF HTTP Binding 225
WCF services 187
Web service activities 115
Web Service security 181
Web Services
    Federation security 188
    message signatures 186
    negative testing 151
    security scenario 184
    testing 17
    transport JMS 31
Windows Authentication 190
Windows store 235

wizard
    Add New DB data Query 313
    Update Step/Service 170, 173
workbench 18
WS-Addressing version 209
WSE3 security 188
WsHttpBinding 189
WS-Reliable Messaging 230
WS-Security 231

X

X.509 certificates
    requiring 206
XML
    checkpoint 244
    transforming 86
    validating 84
XML activities 114
XML data grid 294
XPath checkpoint 244