HP Service Test

Software Version: 11.52

User Guide

Document Release Date: May 2013

Software Release Date: May 2013

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 Notice

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Google™ and Google Maps™ are trademarks of Google Inc

Intel® and Pentium® are trademarks of Intel Corporation in the U.S. and other countries.

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

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 Online 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

Contents

About the *HP Unified Functional Testing User Guide*

The *HP Service Test User Guide* describes how to use Service Test to test your applications. It provides step-by-step instructions to help you create, debug, and run tests, and report defects detected during the testing process.

Prerequisite Background

This guide is intended for Service Test users at all levels. Readers should already have some understanding of functional testing concepts and processes, and know which aspects of their application they want to test.

Topic Types

The content in the Service Test guides is organized by topics. Three main topic types are in use: **Concepts**, **Tasks**, and **Reference**.

Topic Type	Description	Usage
Concepts	Background, descriptive, or conceptual information.	Learn general information about what a feature does.
Tasks	**Instructional Tasks.** Step-by-step guidance to help you work with the application and accomplish your goals. Task steps can be with or without numbering: • **Numbered steps.** Tasks that are performed by following each step in consecutive order. • **Non-numbered steps.** A list of self-contained operations that you can perform in any order.	• Learn about the overall workflow of a task. • Follow the steps listed in a numbered task to complete a task. • Perform independent operations by completing steps in a non-numbered task.
	Use-case Scenario Tasks. Examples of how to perform a task for a specific situation.	Learn how a task could be performed in a realistic scenario.
Reference	**General Reference.** Detailed lists and explanations of reference-oriented material.	Look up a specific piece of reference information relevant to a particular context.

Topic Type	Description	Usage
	User Interface Reference. Specialized reference topics that describe a particular user interface in detail. Selecting Help on this page from the Help menu in the product generally open the user interface topics.	Look up specific information about what to enter or how to use one or more specific user interface elements, such as a window, dialog box, or wizard.
Troubleshooting and Limitations	**Troubleshooting and Limitations.** Specialized reference topics that describe commonly encountered problems and their solutions, and list limitations of a feature or product area.	Increase your awareness of important issues before working with a feature, or if you encounter usability problems in the software.

Service Test Help Contents

The Service Test Help includes the following:

Type	Included Documentation
Getting to Know Service Test Documentation	• **Readme** provides the latest news and information about Service Test. Select **Start > All Programs > HP Software > HP Service Test > Readme**. **Note:** For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33. • **Installation Guides** explain how to install and set up Service Test or the HP Functional Testing Concurrent License Server. Select **Help > Service Test Help** and click the link to the appropriate guides from the left pane. • **Tutorials** teach you basic Service Test skills and show you how to design tests for your applications. Select **Help > Service Test Tutorial**. • **Service Test Product Movies** provide an overview and step-by-step instructions describing how to use selected Service Test features. Select **Help > Product Movies** or.
Feature Documentation	**Service Test Help** includes: • *HP Service Test User Guide* describes how to use Service Test to test your application. • *HP Run Results Viewer User Guide* describes how to use the Run Results Viewer to view and analyze the run results from your tests or components. Select **Help > HP Service Test Help**.

Additional Online Resources

The following additional online resources are available from the Service Test Help menu:

Resource	Description
HP Software Support Site	Opens the HP Software Support Web site. This site enables you to browse the HP Software Self-solve knowledge base. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. Choose **Help > HP Software Support**. The URL for this Web site www.hp.com/go/hpsoftwaresupport. • Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. • To find more information about access levels, go to: http://h20230.www2.hp.com/new_access_levels.jsp • To register for an HP Passport user ID, go to: http://h20229.www2.hp.com/passport-registration.html
Testing Forums	Opens the testing forums for GUI Testing, API Testing, and Business Process Testing. where you can interact with other users of UFT and discuss topics related to GUI Testing, API Testing, and Business Process Testing. The URLs for these sites are: • API Testing: http://h30499.www3.hp.com/t5/Service-Test-Support-and-News/bd-p/sws-Serv_TEST_SF • Business Process Testing: http://h30499.www3.hp.com/t5/Business-Process-Validation/bd-p/sws-BPT_SF
Service Test Product Page	Opens the HP Service Test product page, with information and related links about Service Test.
Troubleshooting & Knowledge Base	Opens the Troubleshooting page on the HP Software Support Web site where you can search the HP Software Self-solve knowledge base. Choose **Help > Troubleshooting & Knowledge Base**. The URL for this Web site is http://h20230.www2.hp.com/troubleshooting.jsp.
HP Software Community Site	Opens the HP IT Experts Community site, where you can interact with other HP software users, read articles and blogs on HP software and access downloads of other software products.
Manuals Site	Opens the HP Software Product Manuals Web site, where you can search for the most up-to-date documentation for a selected HP Software product. The URL for this Web site is http://support.openview.hp.com/selfsolve/manuals (requires an HP Passport).

Resource	Description
What's New	Opens the Service Test What's New Help, describing the new features and enhancements in this version of Service Test.
Product Movies	Opens a page displaying a list of all product movies.
HP Software Web site	Opens the HP Software Web site. This site provides you with the most up-to-date information on HP Software products. This includes new software releases, seminars and trade shows, customer support, and more. Choose **Help > HP Software Web site**. The URL for this Web site is www.hp.com/go/software.

You can access the following sample applications from the **Start** menu. These applications are the basis for many examples in this guide:

Mercury Flight application. To access from the Start menu, select **All Programs > HP Software > HP Service Test > Sample Applications > Flight API**.

Part 1: Service Test Introduction

Chapter 1: Service Test Introduction

This chapter includes:

Service Test Testing - Overview

Service Test provides tools for the construction and execution of functional tests for headless (GUI-less) systems. For example, you can use Service Test to test standard Web Services, non-SOAP Web Services, such as REST, and so on.

You create an API test by dragging and dropping activities from the Service Test Toolbox pane into the test, displayed in the 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.

For more details about service testing in Service Test, including all of the optional steps included in the testing workflow, see " Testing Services" on page 372.

Service Test Integrated Testing - Overview

Integration for GUI and API Testing

You can integrate your GUI and service testing processes in a single test by including calls from your Service Test tests to GUI tests. When you insert a call to another test, the call is displayed as nested under the relevant action in the canvas.

You insert and modify calls to GUI tests from API tests by dragging the Call GUI Action or Test onto the canvas from the HP Automated Testing Tools node in the Toolbox pane. For more details, see "How to Call Tests from Other Applications" on page 411.

Integration with ALM

You can also use Service Test together with ALMto manage the entire testing process. For example, you can use ALM to:

- Create a project (central repository) of manual and automated tests
- Build test cycles
- Run tests
- Report and track defects

You can also create reports and graphs to help you review the progress of test planning, runs, and defect tracking before a software release.

Tests and components created in Service Test can be saved directly to your ALM project, and you can run Service Test tests and review and manage the results in ALM. For details see "ALM Integration" on page 322, and the *HP Application Lifecycle Management User Guide*.

Note: Unless otherwise specified, references to **Application Lifecycle Management** or **ALM** in this guide apply to all currently supported versions of ALM and Quality Center. Note that some features and options may not be supported in the specific edition of ALM or Quality Center that you are using.

For a list of the supported versions of ALM or Quality Center, see the *HP Service Test Product Availability Matrix*, available from the Service Test Help or the root folder of the Service Test DVD. The most up-to-date product availability matrix is available from the HP Software Product Manuals site, at http://h20230.www2.hp.com/selfsolve/manuals (requires an HP Passport).

For details on ALM or Quality Center editions, see the *HP Application Lifecycle Management User Guide* or the *HP Quality Center User Guide*.

Business Process Testing in Service Test - Overview

Business Process Testing enables Subject Matter Experts to create tests using a keyword-driven methodology for testing.

The Business Process Testing model is role-based, allowing non-technical Subject Matter Experts to collaborate effectively with Automation Engineers. Subject Matter Experts define and document business processes, business components, and business process tests, while Automation Engineers define the required resources and settings, such as shared object repositories, function libraries, and recovery scenarios. Together, they can use Service Test to build, data-drive, document, and run business process tests, without requiring programming knowledge on the part of the Subject Matter Expert.

Business Process Testing opens within Service Test, and provides Service Test users with full access to Business Process Testing functionality. To work with Business Process Testing from within Service Test, you must connect to an ALM project with Business Process Testing support.

Note: ALM projects with Business Process Testing support can be created only in ALM version 11.50 with the ALM Integration Enablement Pack installed or later ALM versions.

When working with ALM 11.50 with the ALM Integration Enablement Pack, Business Process Testing is supported only at a *Technology Preview* level.

For details about Business Process Testing, including relevant user roles and testing methodology, see the *HP Business Process Testing User Guide*.

Service Test Program Management

The following sections describe how you can manage your Service Test program, including available licenses and access levels.

Licensing

Working with Service Test requires a license. When you install Service Test, you select one of the following license types:

- A permanent **seat** license that is specific to the computer on which it is installed (includes a 30-day demo license)
- A network-based **concurrent** license that can be used by multiple Service Test users

You can change your license type at any time (as long as you are logged in with administrator permissions on your computer). For example, if you are currently working with a seat license, you can choose to connect to a concurrent license server, if one is available on your network.

For information on modifying your license information, see the *HP Service Test Installation Guide*.

Note: You can also open Service Test using a legacy license, although the functionality will be limited to the service that you are licensed to use. For example, you can open Service Test using a legacy QuickTest Professional or Service Test license and access GUI testing or API testing functionality.

Access Permissions

You must make sure the following access permissions are set to run Service Test or to work with ALM.

Permissions Required When Working with Service Test

You must have the following file system permissions:

- Full read and write permissions for all the files and folders under the Service Test installation folder
- Full read and write permissions to the Temp folder
- Read permissions to the Windows folder and to the System folder

You must have the following registry key permissions:

- Full read and write permissions to all the keys under **HKEY_CURRENT_USER\Software\Mercury Interactive**
- Read and Query Value permissions to all the **HKEY_LOCAL_MACHINE** and **HKEY_CLASSES_ROOT** keys

Permissions Required When Working with ALM

You must have the following ALM-specific permissions to use Service Test with ALM:

- Full read and write permissions to the ALM cache folder
- Full read and write permissions to the Service Test Add-in for ALM installation folder

Permissions Required When Working with Business Process Testing

The ALM Project Administrator can control access to a project by defining which users can log in to it and by specifying the types of tasks each user may perform. The ALM Project Administrator can assign permissions for adding, modifying, and deleting folders, components, steps, and parameters in the Business Components module of an ALM project.

You need to make sure you have the required ALM permissions before working with business components and application areas.

- To work with component steps in ALM, you must have the appropriate **Add Step**, **Modify Step**, or **Delete Step** permissions set. You do not need **Modify Component** permission to work with component steps. The **Modify Component** permission enables you to work with component properties (the fields in the component Details tab).
- To work with parameters in ALM or in a testing tool, you must have all the parameter task permissions set in ALM.
- To modify application areas, you must have the required permissions for modifying components, and adding, modifying, and deleting steps. All four permissions are required. If one of these permissions is not assigned, you can open application areas only in read-only format.

For more information on setting user group permissions in the Business Components module, see the *HP Business Process Testing User Guide*.

Software Updates

Relevant for: GUI testing and API testing

By default, Service Test automatically checks for online software updates once every seven days. When you start Service Test, it checks to see if the last automatic update took place more than seven days ago, and if so, it checks for updates.

You can also manually check for updates to all HP products installed on your computer at any time by choosing **Help > HP Update** from within Service Test, or by choosing **Start > Programs > HP Software > HP Service Test > HP Update**. (You must be logged on with Administrator privileges to use HP Update.)

If updates are available, you can choose which ones you want to download and (optionally) install. Follow the on-screen instructions for more information.

Note: For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" below.

Accessing Service Test in Windows 8 Operating Systems

Service Test applications and files that were accessible from the **Start** menu in previous versions of Windows are accessible in Windows 8 from the **Start** screen or the **Apps** screen.

- **Applications (.exe files).** You can access Service Test applications in Windows 8 directly from the **Start** screen. For example, to start Service Test, double-click the **HP Service Test** shortcut  .

  Other examples of applications accessible from the **Start** screen include:

  - The Run Results Viewer
  - All Service Test tools, such as the Password Encoder and the License Validation Utility
  - The API testing sample Flight applications
- **Non-program files.** You can access documentation from the **Apps** screen.

Note: By default, the Start and Apps screens on Windows 8 are set to open Internet Explorer in Metro Mode. However, if User Account Control is turned off on your computer, Windows 8 will not open Internet Explorer in Metro mode. Therefore, if you try to open an HTML shortcut from the Start or Apps screen, such as the Service Test Help or Readme file, an error will be displayed.

To solve this, you can change the default behavior of Internet Explorer so that it never opens in Metro mode. In the **Internet Properties** dialog box > **Programs** tab, select **Always in Internet Explorer on the desktop** for the in the **Choose how you open links** option. For more details, see http://support.microsoft.com/kb/2736601 and http://blogs.msdn.com/b/ie/archive/2012/03/26/launch-options-for-internet-explorer-10-on-windows-8.aspx.

Chapter 2: Service Test at a Glance

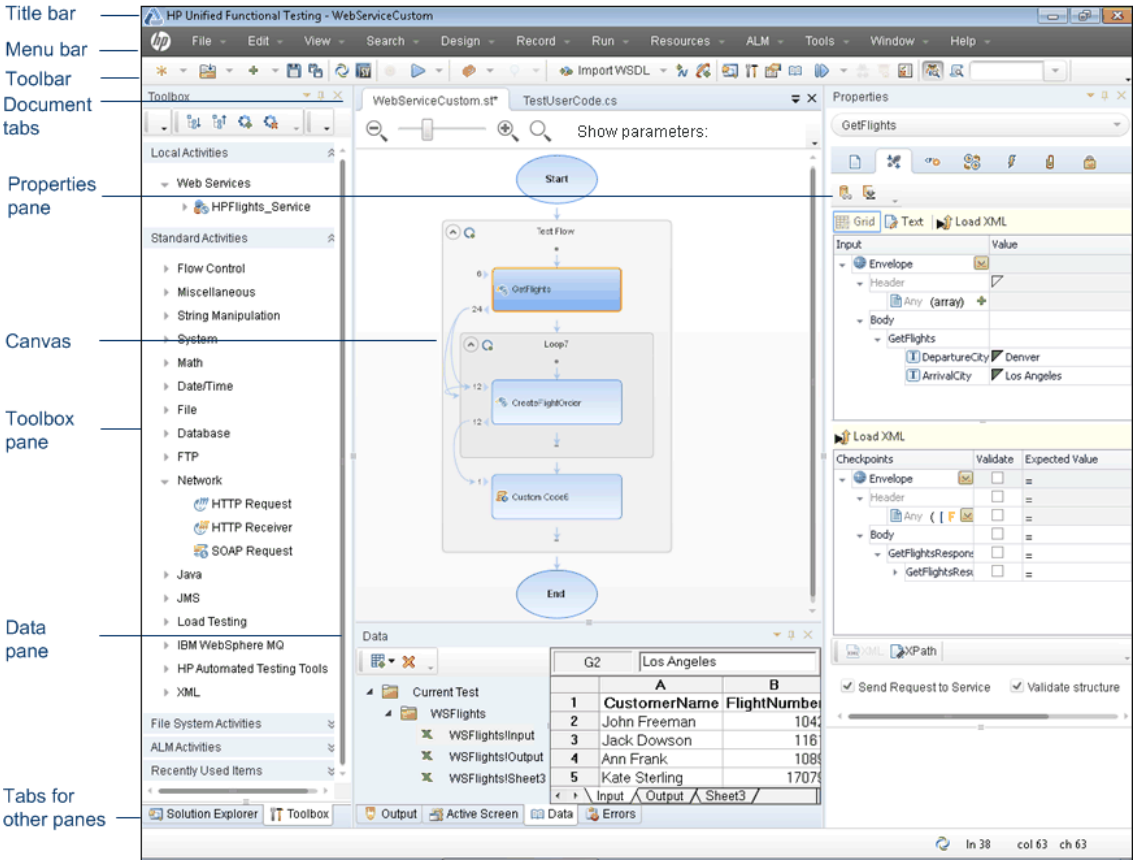
This chapter includes:

Concepts

Service Test Main Window Overview

The main window in Service Test provides multiple areas to design, edit, debug, and run testing documents. The following table introduces the key elements in the Service Test window:

Element	Description
"Start Page"	Welcomes you to Service Test. You can use the shortcut buttons to open new and existing documents. For details, see "Start Page" on page 51.
Document Pane	The main area to design and edit testing documents. • **Canvas.** Visually displays the test flow as a series of steps. For details, see "The Canvas" on page 373. • **Editor.** Displays each test step as a line of code and enables you to create user code using C# code. For details, see "Editor at a Glance" on page 37.
"Service Test Panes at a Glance"	Provide information and functionality about the current test. For details, see "Service Test Panes at a Glance" on page 38.

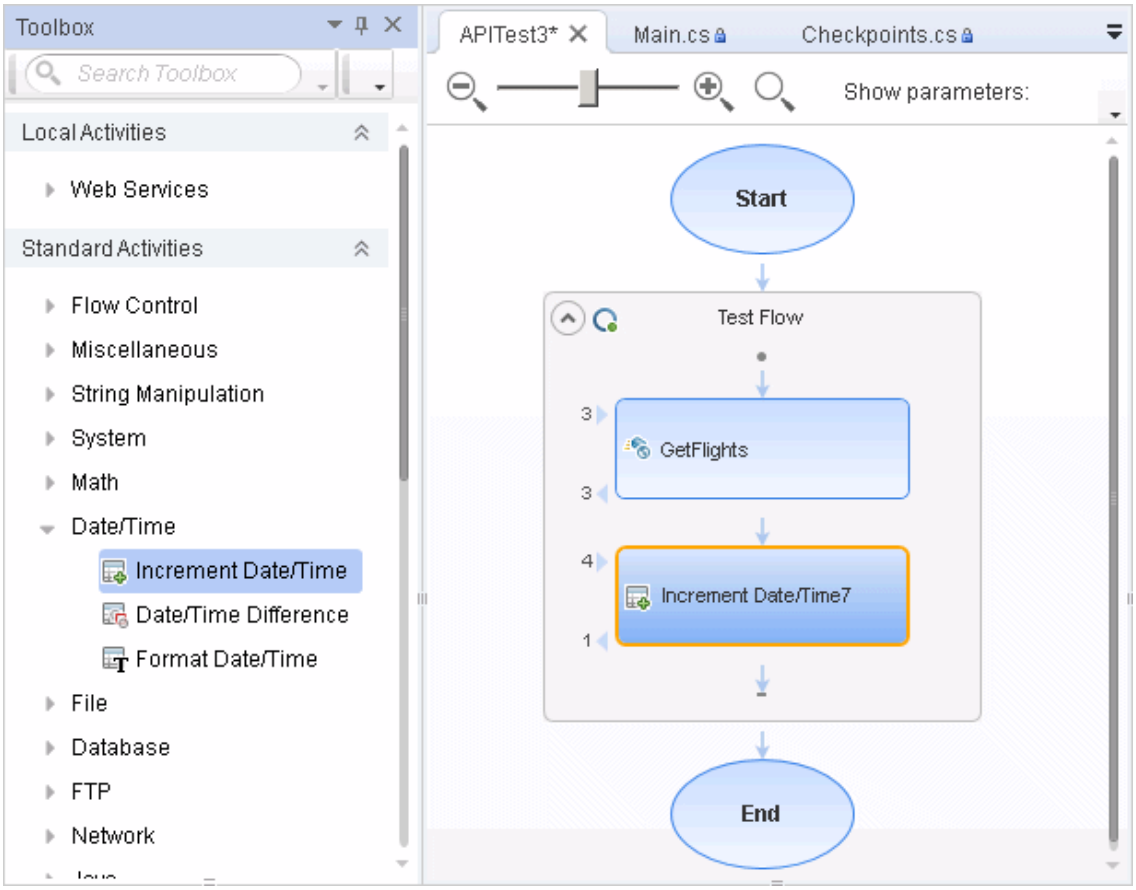


For details about the key areas displayed in the main window, see the Glance subsections in the remainder of this chapter.

Canvas at a Glance

The canvas provides a visual representation of the test flow.

The canvas displays a flow of the steps found within a test, including parameter links between steps. You add steps to the canvas by dragging them from the **Toolbox** pane.

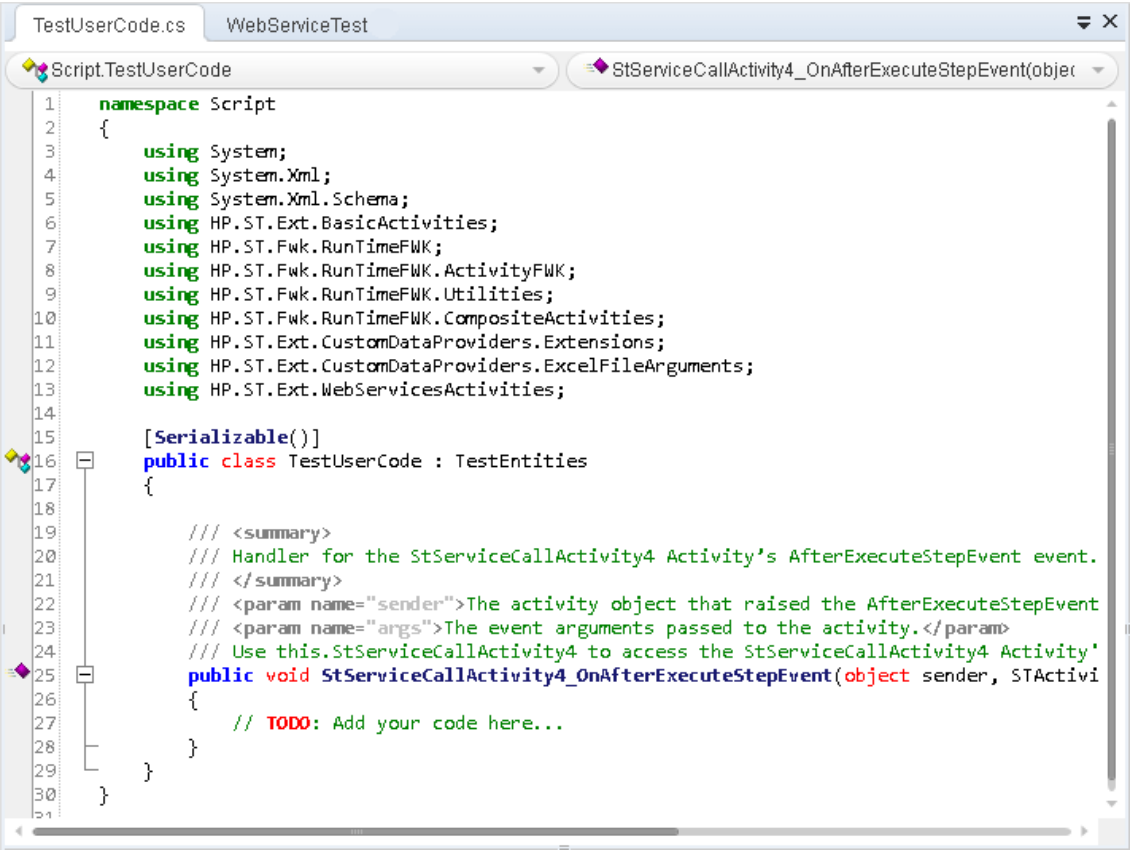


For details on the canvas, see "The Canvas" on page 373.

Editor at a Glance

The Editor displays test steps as lines of code and enables you to add steps to a user code file using C# code.

The Editor also enables you to write customized user code for run during test steps or to write custom code which defines the properties and behavior of different test steps.



For details on using the Editor, see "Editing Text and Code Documents" on page 133.

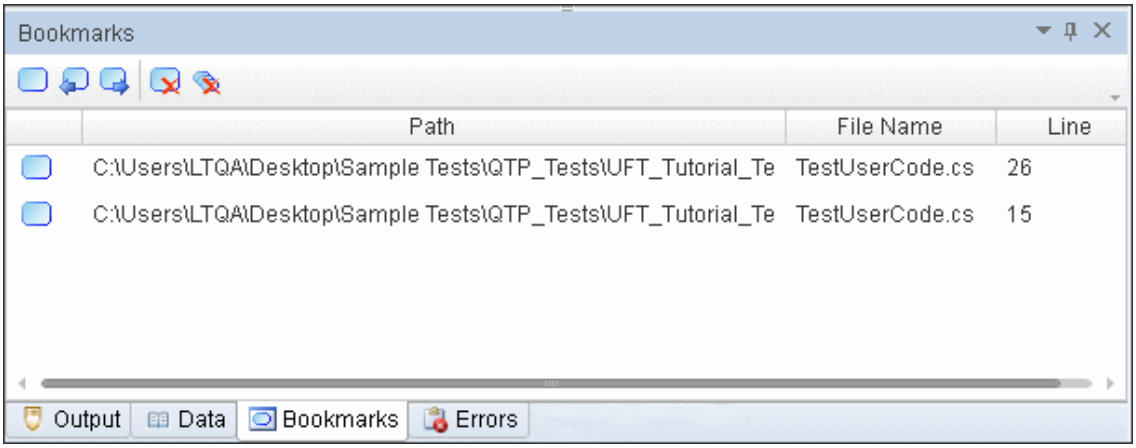
Service Test Panes at a Glance

The following sections describe the available Service Test panes:

Relevant for: GUI tests scripted GUI components, function libraries, and user code files

The Bookmarks pane displays all bookmarks inserted into your user code files. The pane enables you to create bookmarks, view the information about inserted bookmarks, and navigate to the user code file containing the bookmark.

To view the Bookmarks pane, select **View > Bookmarks.**



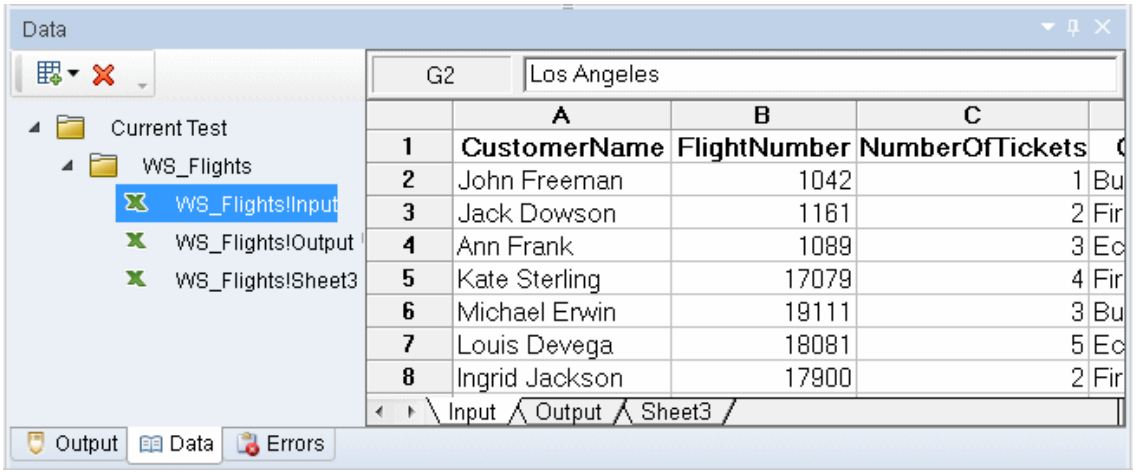
For details, see "Bookmarks Overview" on page 84.

Data Pane at a Glance

The Data pane displays the relevant data for your test or component.

The Data pane can display data from a local data table, an Excel spreadsheet, a database connection, or an XML file. These data sources are then used to parameterize properties for test steps.

To view the Data pane, select **View > Data**.



For details about the Data pane, see "Data Pane" on page 96

Errors Pane at a Glance

The Errors pane provides a list of missing resources and coding syntax errors from your test. If the pane is not currently displayed, Service Test automatically opens it when an error is detected. You can double-click an error to locate in the error in your test or component.

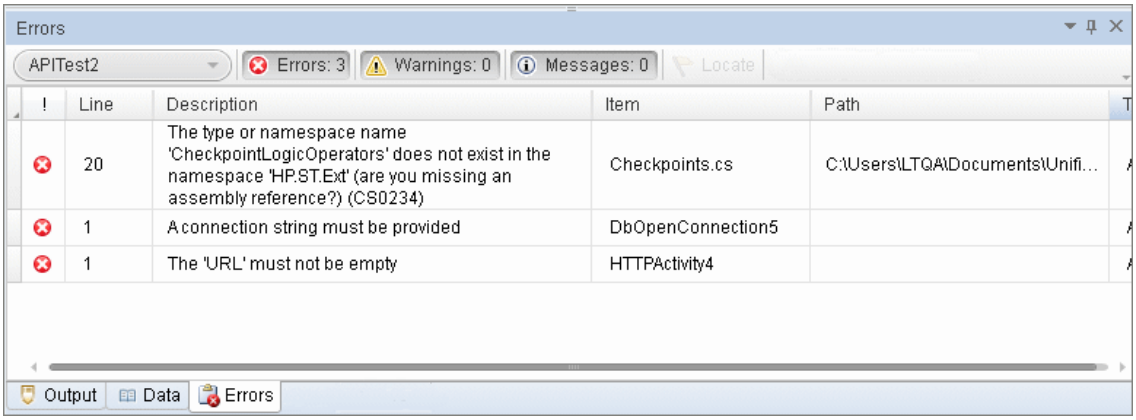
Missing resources are the resources that are specified in your test or component but cannot be found. Information errors provide a list of coding syntax errors in your user code.

Missing resources listed in this pane are missing resource files, such as references called by the test during a run session or user functions or objects called by a test.

The pane also displays coding syntax errors or individual step property errors. If there is a coding syntax error in a user code file or a property value error in a step, it is displayed in the Errors pane.

In addition, if a property value for a step is undefined, it is listed in the Errors pane as a test error.

To view the Errors pane, select **View > Errors**.



For details, see "Errors Pane" on page 172.

Debug Panes at a Glance

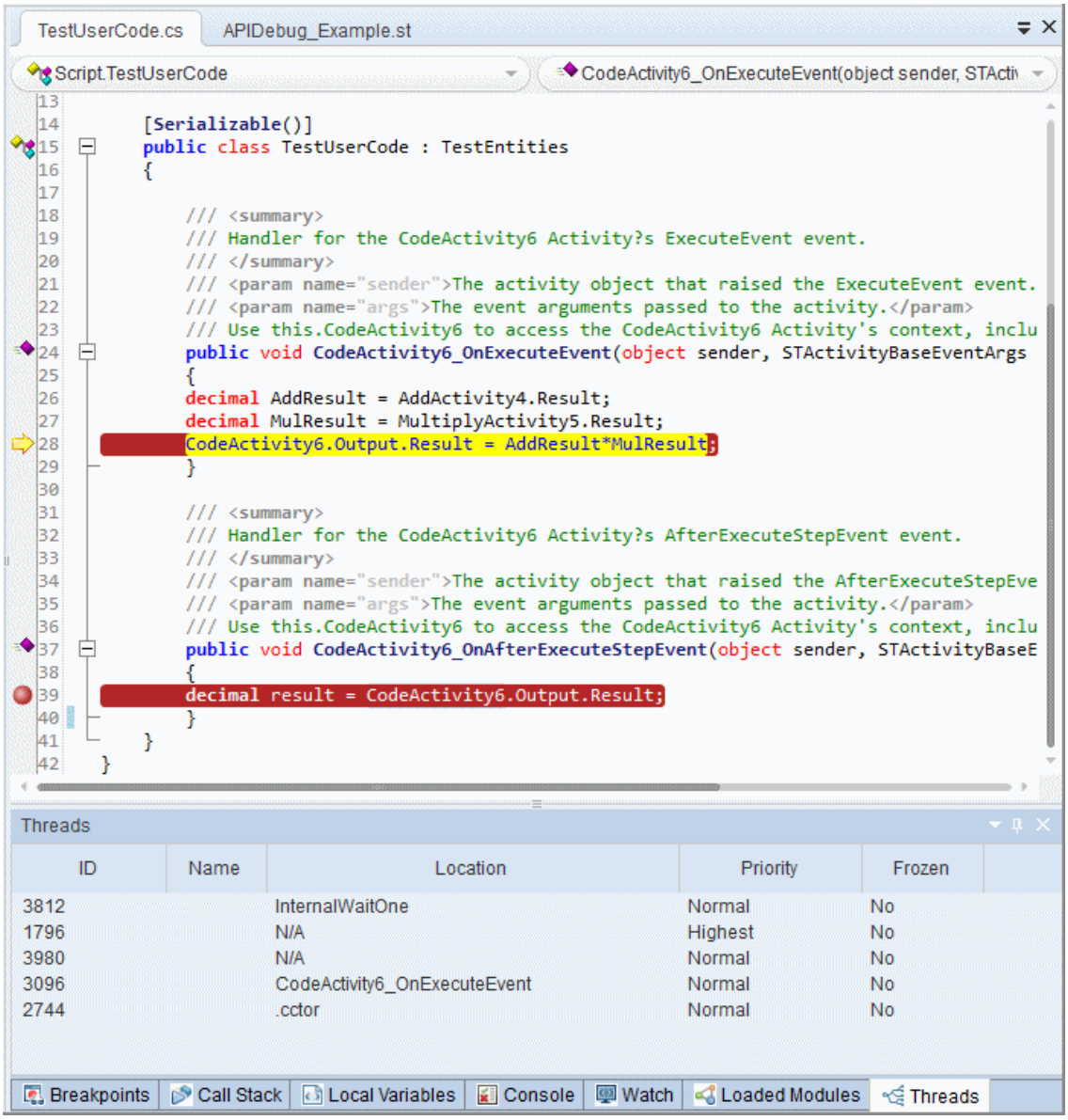
The Debug panes assist you in debugging your user code file, and contains the following options:

Tab	Description
Breakpoints	Displays information about breakpoints inserted into user code files and allows you enable or disable the breakpoints. For details, see " Breakpoints Pane" on page 114.
Call Stack	Displays information about the function and method calls currently running in a test or user code file. For details, see "Call Stack Pane" on page 116.

Tab	Description
Loaded Modules	Displays information on the `.dll` files associated with the current run session. For details, see "Loaded Modules Pane " on page 119.
Threads	Displays information about all the threads running in the current context of the test. For details, see "Threads Pane " on page 120.
Local Variables	Displays the current value and type of all variables that were recognized up to the last step performed during the run session that you are debugging. You can also set or modify the values of the variables that are displayed. For details, see "Local Variables Pane" on page 122.
Console	Enables you to run lines of script to set or modify the current value of a variable, code object, property, method, or function call in your user code file. For details, see "Console Pane" on page 124.
Watch	Displays the current value and type of any variable or expression that you added to the Watch pane. For details, see "Watch Pane" on page 126.

To view the Debug panes, select **View > Debug** and select the specific Debug pane you want to view.

This image below shows the Threads pane visible with the other panes open as individual tabs. You can display any debug pane by clicking on the relevant tab.

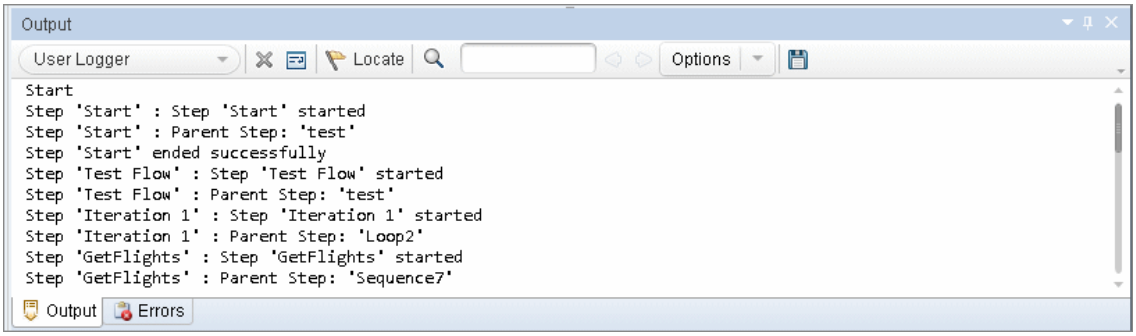


For details, see "Debug Panes" on page 110.

Output Pane at a Glance

This pane enables you to view the Output log for the compilation and test run.

To view the Output pane, select **View > Output**.



For details, see "Output Pane" on page 180.

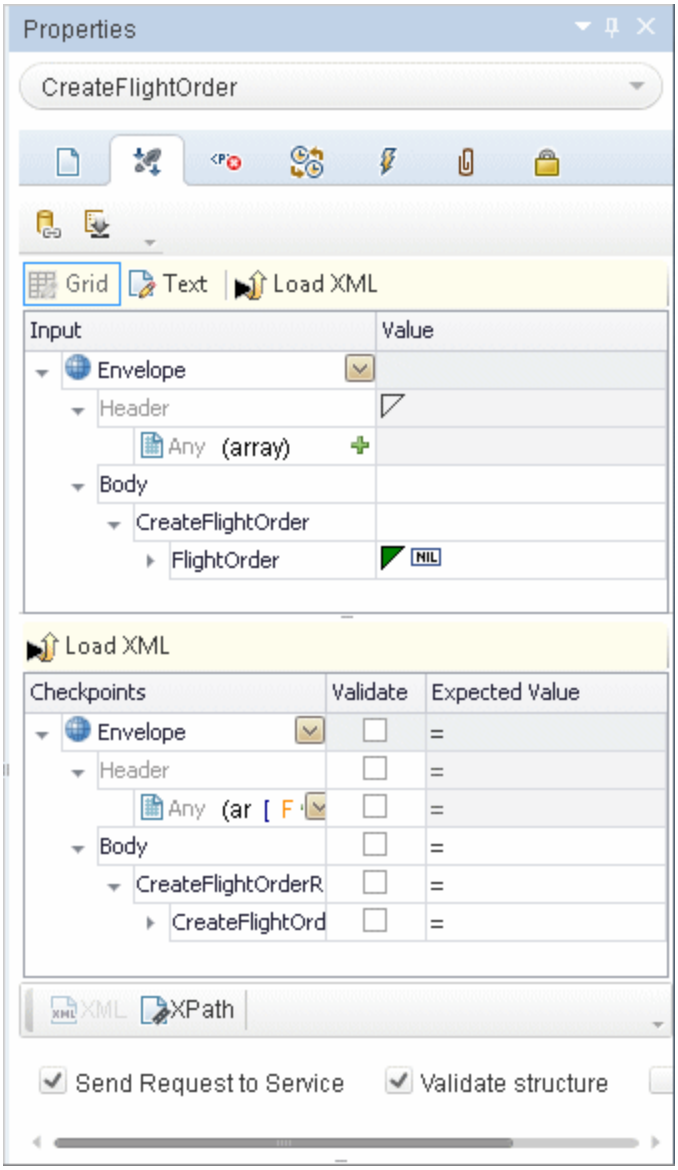
Properties Pane at a Glance

The Properties pane displays and enables you to edit properties and parameters for a selected test, action, or component. The information displayed in this pane is dependent upon the active document.

The Properties pane enables you view and edit the property values, schemas, and event handlers for the different activities in the test canvas. Using the **Input/Checkpoints** tab, you can specify input and output values for the highlighted activity and set checkpoints for that activity. Using the **Events** tab, you can open the `TestUserCode.cs` file in the Editor, enabling you to write event code to use during a particular test step. Using the **Data Sources** tab (visible only when the test flow is selected), you can select data sources to use with your test.

To view the Properties pane, select **View > Properties**.

The image below shows the **Input/Checkpoints** view for a test step. You can display the other tabs by clicking on them.



For details on the Properties pane, see "Properties Pane User Interface" on page 186.

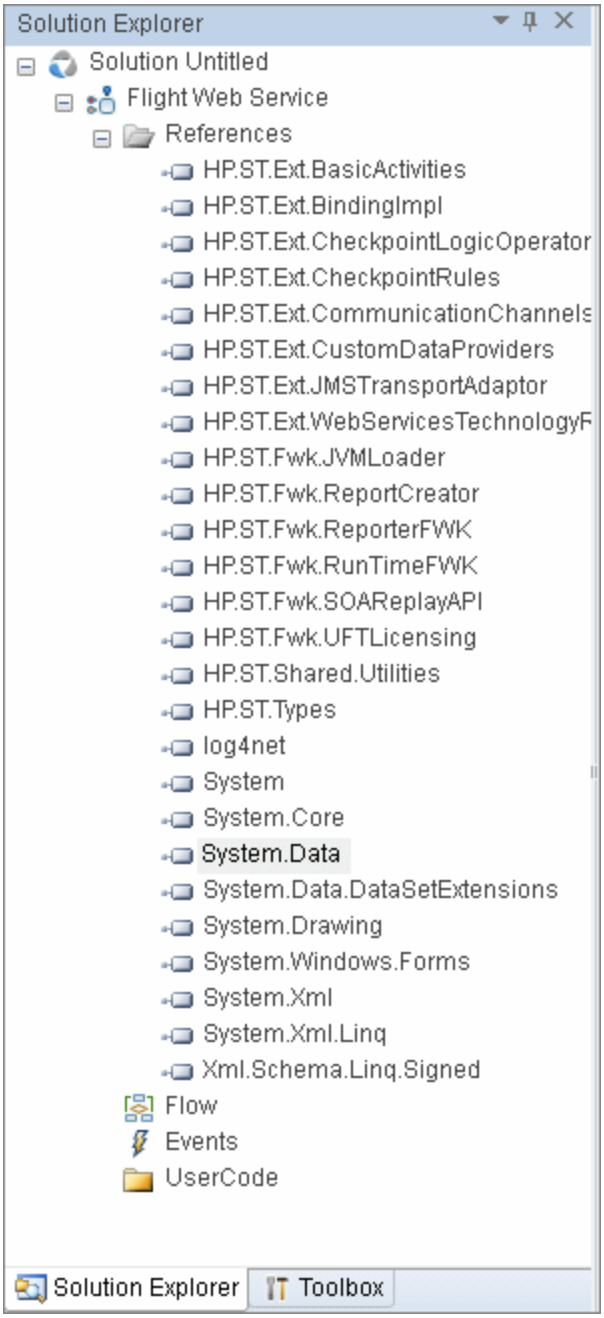
Solution Explorer Pane at a Glance

The Solution Explorer Pane displays all the resources associated with a test or component. You can add, remove, and manage all of the resources in your test or to view and open all of the resources in your component through this pane. In addition, the Solution Explorer pane enables you to combine multiple types of tests, components, and user code files into a single solution.

Test nodes can include numerous references, events, and actions. The Solution Explorer pane displays all external references for a test, the event handler code file (**TestUserCode.cs**), any actions created as part of a test, and any user-defined code files added to a test.

You can open most associated resources and references from the Solution Explorer pane by double-clicking on the name of the resource or reference.

To view the Solution Explorer pane, select **View > Solution Explorer**.



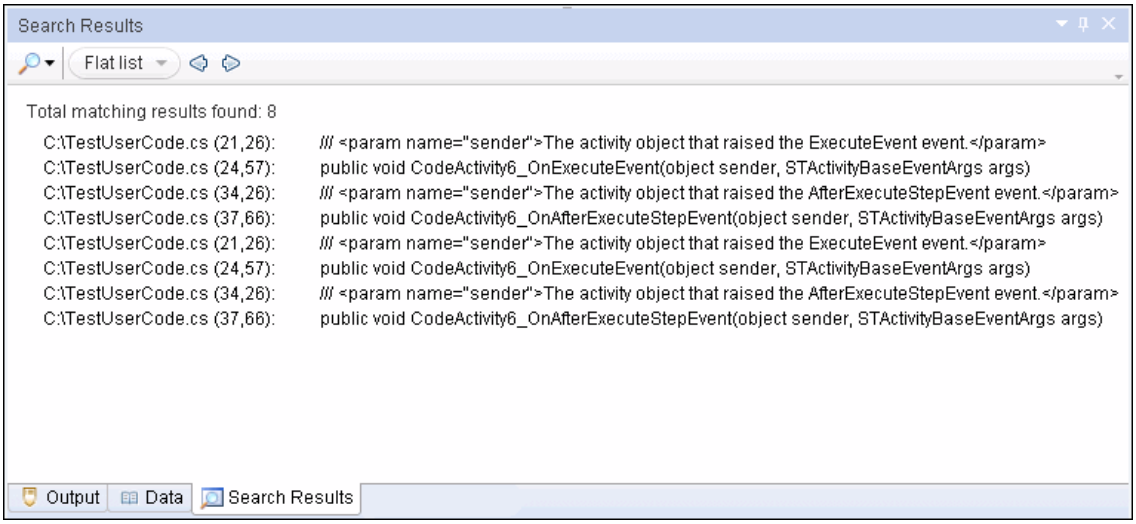
For details, see "Solution Explorer Pane Overview" on page 235.

Search Results Pane at a Glance

The Search Results pane displays the results of searches performed using the options in the **Search** menu. Using this pane, you can browse the results of your search, locate a specific result, and perform recent searches in order to receive updated results.

To view the Search Results pane, select **View > Search Results**.

The image below shows the Search Results pane for API testing.



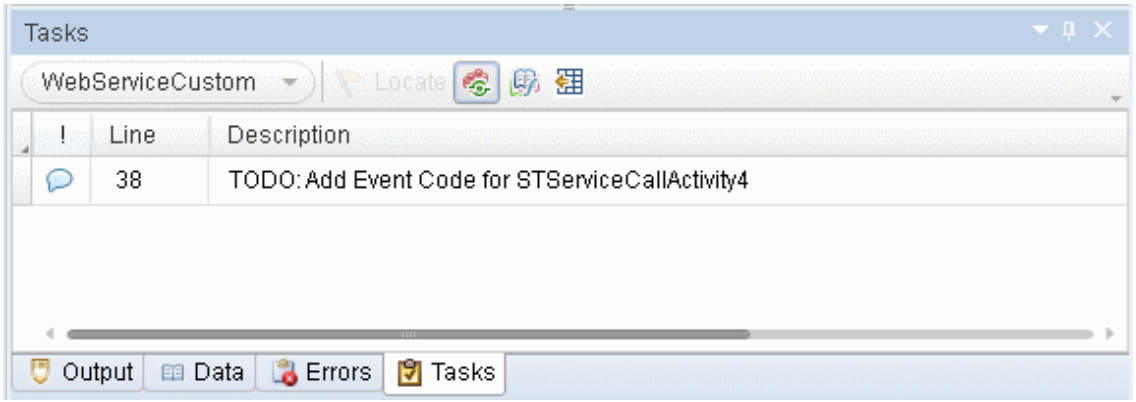
For details, see " Search Results Pane Overview" on page 228.

Tasks Pane at a Glance

The Tasks pane enables you to create, view, and manage your TODO tasks. A TODO task is anything that needs to be done in a test or component, such as providing information relevant for handing over a testing document, or adding a reminder to yourself to add steps that test a new page in your application. Your TODO tasks are saved with the test or component.

The Tasks pane also enables you to view the TODO comments that exist in an open user code file and navigate to the line in the user code file containing the TODO comment.

To show or hide the Tasks pane, select **View > Tasks.**

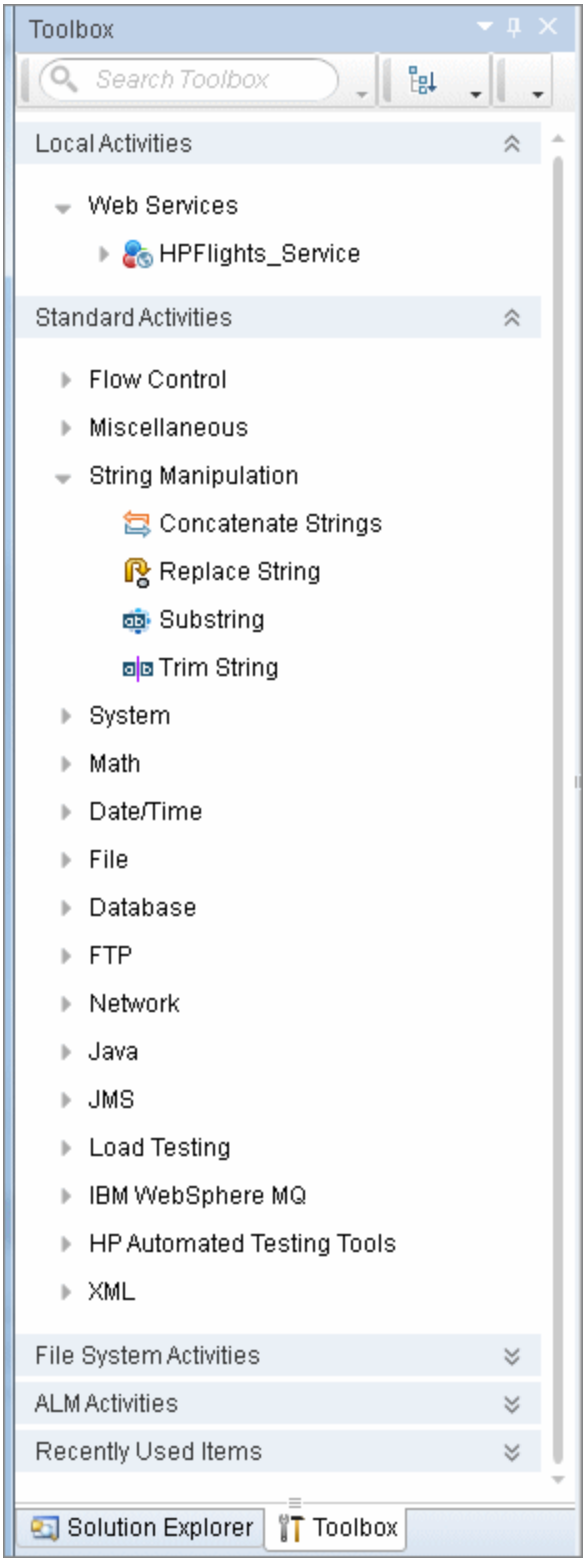


For details, see "Tasks Pane User Interface" on page 251.

Toolbox Pane at a Glance

The Toolbox pane contains all the activities and flow control activities you can use to create a test. You can drag and drop activities from the Toolbox pane into the canvas to create a test, or you can double-click activities to create a test flow. Custom activities (such as Web services) are also added to the Toolbox pane, under the **Local Activities** node.

To view the Toolbox pane, select **View > Toolbox**.



For details on the Toolbox pane, see "Toolbox Pane Overview" on page 255.

Tasks

How to Start Service Test

1. From the **Start** menu, select **All Programs > HP Software > HP Service Test > HP Service Test**, or double-click the **HP Service Test** shortcut  on your desktop.

   The Service Test window opens, displaying the "Start Page".

2. If your installation of Service Test integrates with ALM, connect to your ALM project, as described in "How to Work with Tests and Components in ALM" on page 327.

3. Begin creating or editing your tests. For details on how to manage testing documents, see "How to Create and Manage Tests" on page 56.

4. Set your global test variables. For details on setting test variables in the Properties, pane, see "How to Define Test Properties or User/System Variables" on page 59.

Reference

About Service Test Dialog Box

This dialog box enables you to view information regarding the features installed on your computer, as well as other basic information about your computer. This information is useful for troubleshooting and when working with HP Software Support.

To access	Select **Help > About HP Service Test**.

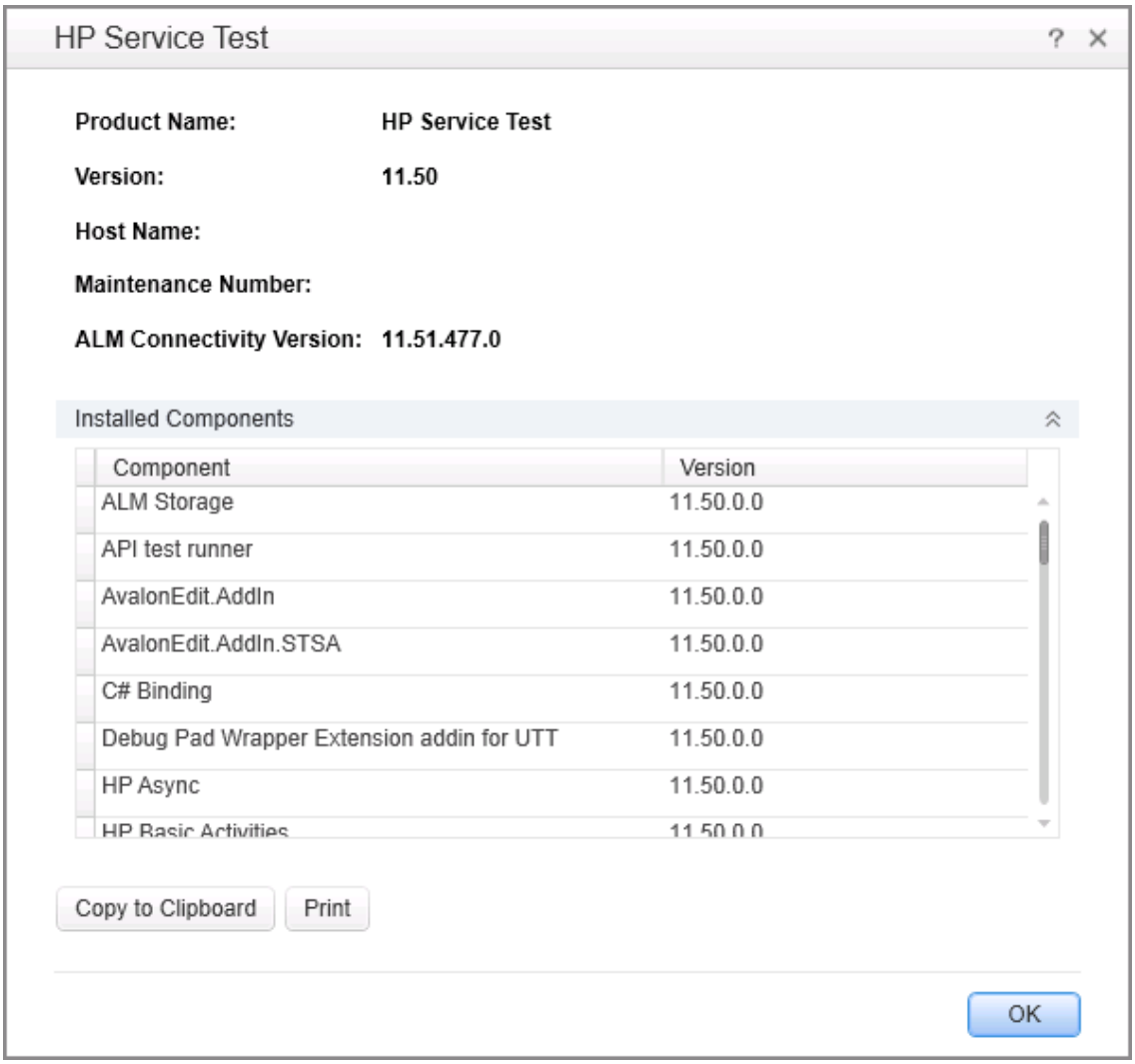
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
Version information	The version of Service Test that is installed on your computer and its build number.
License Type	Displays the license type installed on your computer: **Seat** or **Concurrent**.

UI Element	Description
Installed Features	The list of installed add-ins that are installed on your computer. This window displays the feature name and version number for each add-in.
Detailed Info	Opens the "About Service Test Detailed Information Window" (described on page 50), which displays all information about the application components installed on your computer.
System Info	Opens the Windows system information window which displays information about the hardware and software components running on your computer.
License	Opens the License Summary dialog box detailing the installed license. Select **Modify License** to change the type of license.

About Service Test Detailed Information Window

This window displays detailed information on Service Test components installed on your computer.



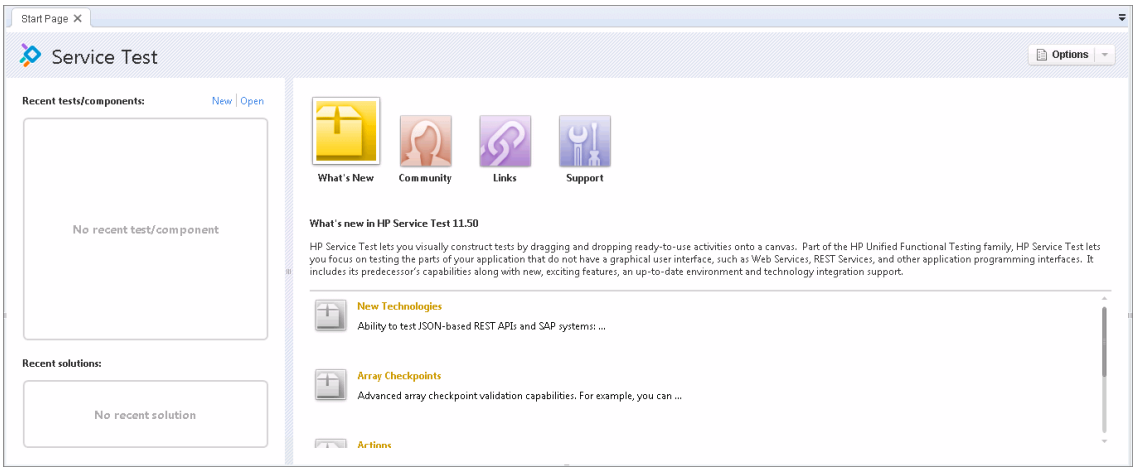
To access	In the About Service Test dialog box, click the **Detailed info** button.
See also	"About Service Test Dialog Box" on page 49

User interface elements are described below:

UI Elements	Description
Product Name	The name of the product.
Version	The version number of the product listed in this window.
Host Name	The name of the computer hosting Service Test and the operating system.
ALM Connectivity Version	The version number of the ALM Connectivity Add-in.
Maintenance Number	The maintenance number for your software.
Installed Components	The list of Service Test components installed on your computer, including version number.

Start Page

This page welcomes you to Service Test and describes the new features in this release, including links to more information about these features.



To access	1. Start Service Test, as described in "How to Start Service Test" on page 48. 2. If the Start Page window is not displayed, select **View > Start Page.**

User interface elements are described below:

UI Elements	Description
Recent Tests/Components Recent Solution	Enables you to create a test or component or a solution by clicking the **New** or **Open** buttons after selecting a test, component or solution name.
Options	Options for viewing the start page. You can select **Display the Start Page on Startup** to display the Start Page immediately upon the launch of Service Test. You can select **Close Start Page after test loads** if you want to close the start page after loading a test.
What's New area	Click the **What's New** button to display information on the list of the newest features, enhancements, and supported environments in the latest version of Service Test.
Community Area	Click the **Community** button to open news about Service Test, links to Service Test in the HP Software site, links to the API Testing forum, and the Business Process Testing Forum, or the Service Test blog.
Links Area	Click the **Links** button to open links to the Service Test Help and the tutorial.
Support Area	Click the **Support** button to open links to Service Test support, the Knowledge Base, and Troubleshooting areas of the HP Software Support Web site.

Service Test Program Folder Structure

After the Service Test setup process is complete, the following items are added to your Service Test program folder (**Start > All Programs > HP Software > HP Service Test**):

- **Documentation.** Provides the following links to commonly used documentation files:

Option	Description
HP Service Test Help	Opens the Service Test Help, which displays links to commonly used topics and movies that describe how to use Service Test, as well as additional links to HPSoftware Websites. The Service Test Help provides access to all guides available for Service Test, such as Getting Started guides, Helps, reference files, and links to printer-friendly (PDF format) documentation. It contains various navigation options to help you find the information you need. The **Welcome to the Service Test Help** section describes how to navigate, use, and get updates for the Service Test Help.
Service Test Tutorial	Opens the Service Test tutorial, which teaches you basic skills and shows you how to start testing your applications.

- **Sample Application.** Contains the following links to sample application that you can use to

practice testing with Service Test:

Option	Description
Sample Application	Opens a GUI-less flight service application. **Note:** You must have administrator privileges to use this application.

- **Tools.** Contains the following utilities and tools that assist you with the testing process:

Option	Description
Activity Wizard	Opens the "Activity Wizard" (described on page 684), which enables you to create custom activities that will be visible in the Toolbox pane.
Additional Installation Requirements	Opens the Additional Installation Requirements Utility, which displays any prerequisite software that you must install or configure to work with Service Test. For details, see the
Commuter License Tool	Opens the Commuter Licensing utility, to check in or check out concurrent licenses when working at a remote location.
License Validation Utility	Opens the License Validation utility, which enables you to retrieve and validate license information. For details, click the **Help** button in the License Validation Utility window.
soapUI to API Test Converter	Converts soapUI tests to a Service Test Test.
Test Batch Runner	Opens the Test Batch Runner application, which enables you to set up Service Test to run several tests in succession. For details, see "How to Create and Run a Test Batch" on page 292.

- **HP Service Test Product Availability Matrix.** Provides a complete list of all environments, programs, and versions that are supported for Service Test.
- **HP Service Test**. Opens the Service Test application.
- **Readme.** Opens the *HP Service Test Readme*, which provides the latest news and information on Service Test.

- For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

Chapter 3: Service Test File Operations

This chapter includes:

Concepts

Testing Documents Overview

A testing **document** is any file that enables you to test your application. You create, maintain, and edit steps, parameters, functions, and other details in the document to enable the testing of your application.

Testing documents can include:

- **Service Test documents:**
  - Tests
  - Actions
  - C# files containing event handler code (`TestUserCode.cs` files)
  - C# files containing user-generated code
- **Business Process Testing documents:**
  - Business process tests
  - Business process flows
  - API components

You work with testing documents in the "Document Pane" (described in "Document Pane" on page 131). In Service Test, you can perform standard file operations with testing documents, including creating, opening, saving, and printing multiple types of documents. You can also save a test with its resources.

The document pane supports the following types of views and functionality:

- **Canvas**. Graphically displays and enables you to edit the flow of your test, action, or component. For details, see "The Canvas" on page 373.
- **Editor**. Provides text and code editing features that facilitate the design of your text, script, and code documents. For details, see "Editing Text and Code Documents" on page 133.

Tasks

How to Create and Manage Tests

The task describes how to manage tests and includes the following steps:

- "Create a new standalone test" below
- "Create a new test within an existing solution" below
- "Open an existing test" on next page
- "Add an existing test to your solution" on next page
- "Edit an open test" on next page
- "Save the current test" on next page
- "Save all open tests" on page 58
- "Save a portable copy of an API test" on page 58
- "Close the current test" on page 58
- "Close all tests" on page 58
- "Switch between open tests" on page 58
- "Delete a test" on page 59

Create a new standalone test

1. (Optional) If you are working with ALM, connect to the relevant ALM project. For details, see "ALM Connection Dialog Box" on page 335.
2. Do one of the following:
   - Select **File > New** and select the type of test to create.
   - Click the **New**  button down arrow and select the type of test to create from the drop-down list.

   A dialog box opens:
   - If you are creating a test or component, see "New <Document> Dialog Box" on page 67.

     This dialog box enables you to specify the type of test or component you want to create, the location in which the new item is stored, and, optionally, the solution in which it is included.
   - If you creating a solution, see "Open/New <Document>/<Resource> Dialog Box" on page 69.

     This dialog box enables you to specify the location in which the new item is stored.

   After you finish entering the information for your new test in the dialog box, a new test is automatically saved and displayed as a tab in the document pane.

Create a new test within an existing solution

Do one of the following:

- Select **File > Add** or click the **Add**  down arrow to add a new test to your solution.
- In the "Solution Explorer Pane" (described on page 240) right-click the <**solution name**> node, select **Add > Add New** and select the type of test to create.

A dialog box opens. For details, see: "Add Test/Component to Solution Dialog Box" on page 64.

After you finish entering the information for your new test in the dialog box, a new test is automatically saved and displayed as a tab in the document pane.

Open an existing test

1. If your test is stored in ALM, connect to the relevant ALM project. For details, see "ALM Connection Dialog Box" on page 335.
2. Do one of the following:
   - Select **File > Open**, click the **Open**  down arrow, or select the list of **Recent** files in the **File** menu to open an existing test. In the "Open/New <Document>/<Resource> Dialog Box" (described on page 69), browse to and select your test.
   - If your test is part of a solution, in the "Solution Explorer Pane" (described on page 240) double-click the selected node for your test.

Add an existing test to your solution

Do one of the following:

- In the "Solution Explorer Pane" on page 234 (described on page 240), right-click the <**solution name**> node and select **Add > Add Existing <Document>**, and select the type of test to add.
- Select **File > Add Existing** or the **Add** down arrow  to add a test to your solution.

A dialog box opens. For details, see "Add <Existing Document> to Solution Dialog Box" on page 63.

Edit an open test

The features available for viewing and editing documents depend on the document type. For details, see "Document Pane Overview" on page 132.

Save the current test

Do one of the following:

- Make sure that the test you want to save is the active tab (you can select the test's tab to bring it into focus), and click the **Save** button  .
- Right-click the tab of the test you want to save and select **Save <test name>** or **Save <test name> As**.

Service Test saves the test with your changes. If you are using **Save As**, the "Save <Resource>/Save <Document> As Dialog Box" (described on page 72) opens.

Note: When you edit a document, an asterisk (*) is displayed in the document tab (for

standalone documents like user code files) or the canvas tab (if the document is part of a test) until the document is saved.

Save all open tests

Select **File > Save All** or click the **Save All** button  .

Note: When you edit a document, an asterisk (*) is displayed in the document tab (for standalone documents like user code files) or the canvas tab (if the document is part of a test) until the document is saved.

Save a portable copy of an API test

1. Prerequisites:
   - If your test was last modified using Service Test 11.10 or later, make sure that the test and its resources are upgraded to Service Test 11.50. Open the test in Service Test 11.50 and save it (**Save** or **Save As**). If the test contains calls to external actions (actions stored in other tests), open and save those tests too.

     Note: Tests last modified in Service Test 11.00 or earlier cannot be opened in Service Test 11.50.

   - Make sure that your resource file (such as WSDL or REST service) has at least one step in the current test.
2. In the Toolbox pane, right-click on the service name and select **Move to > File System Activities**. The service moves to the File System Activities section of the Toolbox pane.
3. Select **File > Save (Other) > Save with Resources**. The "Save <Resource>/Save <Document> As Dialog Box" (described on page 72) opens.

Close the current test

Do one of the following:

- Make sure that the test you want to close is the active test and select **File > Close**.
- To close all tests included in the current solution and the solution, select **File > Close solution**.
- Right-click a test tab and select **Close.**

Close all tests

Do one of the following:

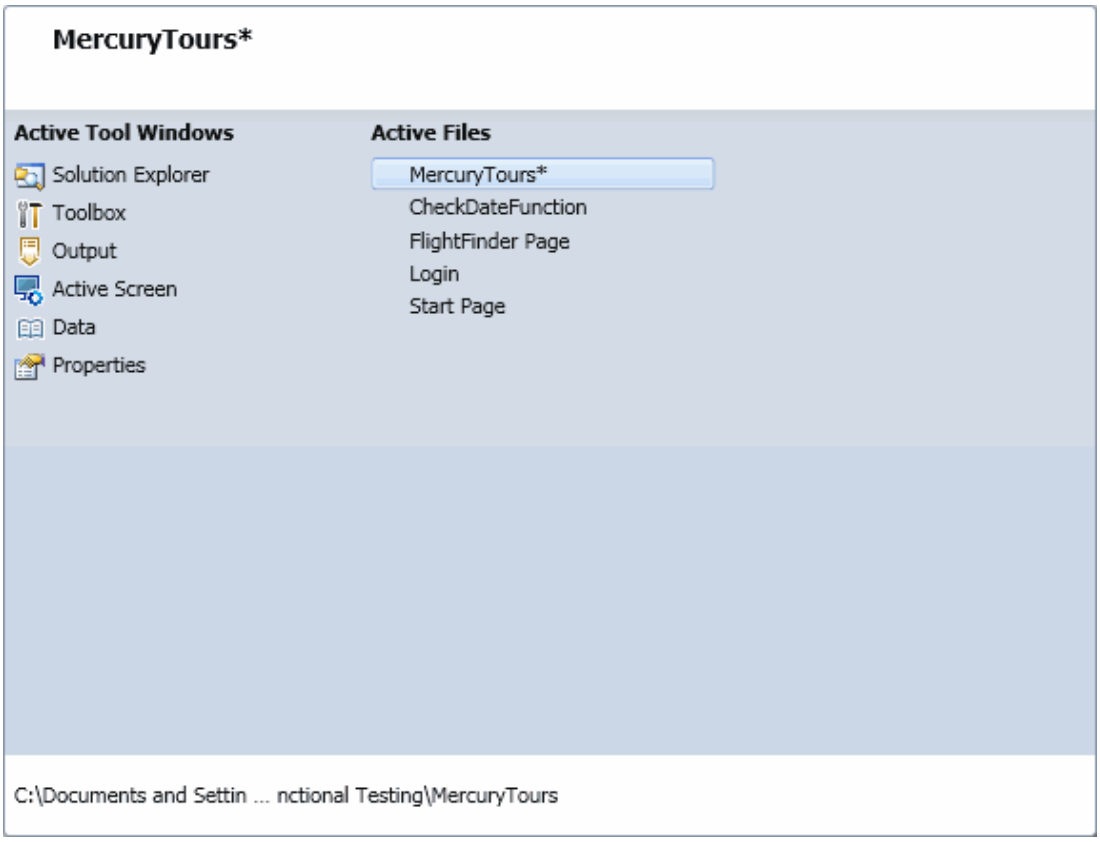
- To close all open tests, right-click any test's tab and select **Close All**.
- To close all open tests except the active test, right-click on the test tab and select **Close All but This**.

Switch between open tests

- To navigate to an open test, select the relevant tab.

> **Note:** If some tabs are not visible due to a lack of space, click the document display arrow in the upper-right hand corner to display a list of all open tests. Select the test you want to view.

- To select your document from a menu of open tests, press and hold CTRL+TAB on your keyboard. A window opens showing a list of all open tests and panes.



Delete a test

- If the test is stored in the file system, select **File > Open**, **File > New**, or **File > Save as** to open a file operation dialog box. Browse to the test, right-click and select **Delete**.

> **Caution:** Before you delete a document, make sure that it is not used by or associated to any other documents.

- If the document is stored in ALM, you delete it from within ALM, regardless of whether it was created in Service Test or in ALM. For details, see *HP Business Process Testing User Guide*.

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

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 Properties pane, open the **Test Input Parameters** view .
2. Click the **Add** button to define a new input or output parameter.
3. In the "Add Input/Output Property/Parameter Dialog Box" (described on page 497), specify a parameter name and data type. All types that were defined in any reference file, are available.
4. Click in the **Default Value** column and specify a value.

Define user variables

You can 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 "Set user variable values " below.

1. Click in a blank area of the canvas. In the Properties pane, 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 Properties pane's **Test Variables** view, click in the **Profile** column and manually enter values.
- Open the Toolbox pane and drag the **Set Test Variable** activity from the **Miscellaneous** category into the **Test Flow** (or loop). In the Properties pane, 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" (described on page 630), select the **Test Variables** option. Select a name and value for the **Variable key** and **Variable value**.
- Open the Properties pane'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 "Coding Service Test Events" on page 641.

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" (described on page 74), specify a name and indicate the profile (if any) from which to copy the properties. If you do not copy the properties 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 details, see the "Test Variables Tab (Properties Pane)" on page 212.

Set OS variable values for the test - optional

This step lets you set global operating system variables that will apply to all steps in the current test run.

1. From the Toolbox pane and drag the **Set OS Environment Variable** activity from the **System** category into the Test Flow or another custom loop.
2. In the Properties pane's Input/Checkpoints tab, 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 Properties pane, open the **Test Variables** view . The System variables are listed in the lower pane.

How to Upgrade Tests Using the Batch Upgrader

Service Test11.50 provides an upgrade tool, `STBatchUpgrader.exe`, located in the `<Service Test installation>/bin` folder. This tool lets you run a batch file to upgrade tests last modified in version 11.10 or 11.20, making them compatible for version 11.50.

Note: The Batch Upgrader tool may not successfully upgrade tests stored in ALM11.00 when working on Windows 2003.

If you do not upgrade your tests with the batch upgrade tool, when you open a test last modified in version 11.10 or 11.20, it prompts you to upgrade the test.

Tests last modified in Service Test 11.00 must first be opened and saved in Service Test 11.10, before you can upgrade them to version 11.50.

For the tool's options, see "Batch Upgrader Command Line Options" on page 76,

The following steps describe how to use the **STBatchUpgrader** tool.

- "Prerequisite" below
- "Locate the Batch Upgrader tool" below
- "Add command line options" below
- "Run the command" below

1. ## Prerequisite

   Make sure that Service Test is not running.

   If you ran the upgrader tool once while Service Test was running, the logs may become corrupted. Backup and delete all of the existing logs in the `<Service Test installation>\bin\logs` folder before proceeding.

   If desired, create a backup copy of the older tests.

2. ## Locate the Batch Upgrader tool

   In the command line, enter the location of the `STBatchUpgrader.exe` file in the `Service Test/bin` sub-folder.

3. ## Add command line options

   Add the relevant options as described in "Batch Upgrader Command Line Options" on page 76, using the following syntax:

   ```
   STBatchUpgrader.exe source [destination] [/ALM url domain
   project] [/login username password] [/log logfile] [/report
   reportfile]
   ```

   For example, the following string runs the upgrade on all tests in the `ST_11_1` folder on the ALM server, `pumpkin`, for the `TEST1` project in the AUTOMATION domain. It places the report in `c:\logs\MyLogfile.log`.

   ```
   STBatchUpgrader.exe Subject\ST11_1 /ALM
   http://pumpkin:8080/qcbin AUTOMATION TEST1 /login user password
   /log c:\logs\MyLogfile.log.
   ```

4. ## Run the command

   Run the command. Verify the validity of the tests in the destination folder.

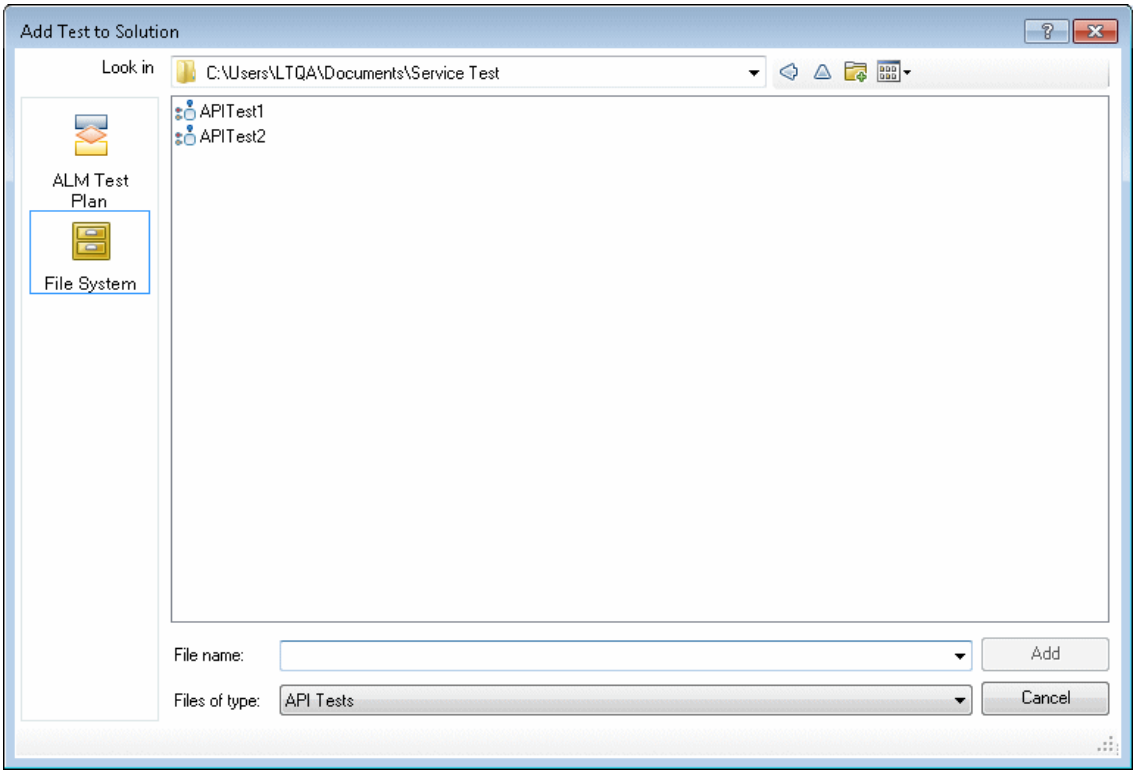
Reference

Add <Existing Document> to Solution Dialog Box

This dialog box enables you to add existing tests to a solution.

This section describes the following dialog boxes:

- Add Test to Solution dialog box
- Add Business Component to Solution dialog box



To access	Do one of the following: • In the "Solution Explorer Pane" (described on page 240), right-click the solution node and select **Add Existing <Document>**. • Select **File > Add Existing <Document>**. • Click the **Add** button down arrow and select the test.

Important information	• Solutions are saved in your file system only and cannot be saved in an ALM project. • Using **File > Add Existing Test** enables you to open multiple tests or components, simultaneously instead of using **File > Open** (which closes all open documents). • If you try to open a test and there is an error with loading the test (such as ALM connectivity problems or a changed test name), an error describing the problem is displayed in the Errors pane.
Relevant tasks	"How to Create and Manage Tests" on page 56

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

UI Element	Description
<sidebar>	The location in which the test is stored, for example `File System` or `ALM Test Plan`. **Note:** You may need to select a different location before browsing to your test.
Look in	The path for the test. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the file list area to navigate to the required folder.
<file list area>	The folders and/or tests stored in the current path.
File name	The name of the tests selected in the <file list area>.
Files of type	Filters a list of displayed files to show only the file types selected. If the current folder contains other file types, they are hidden.

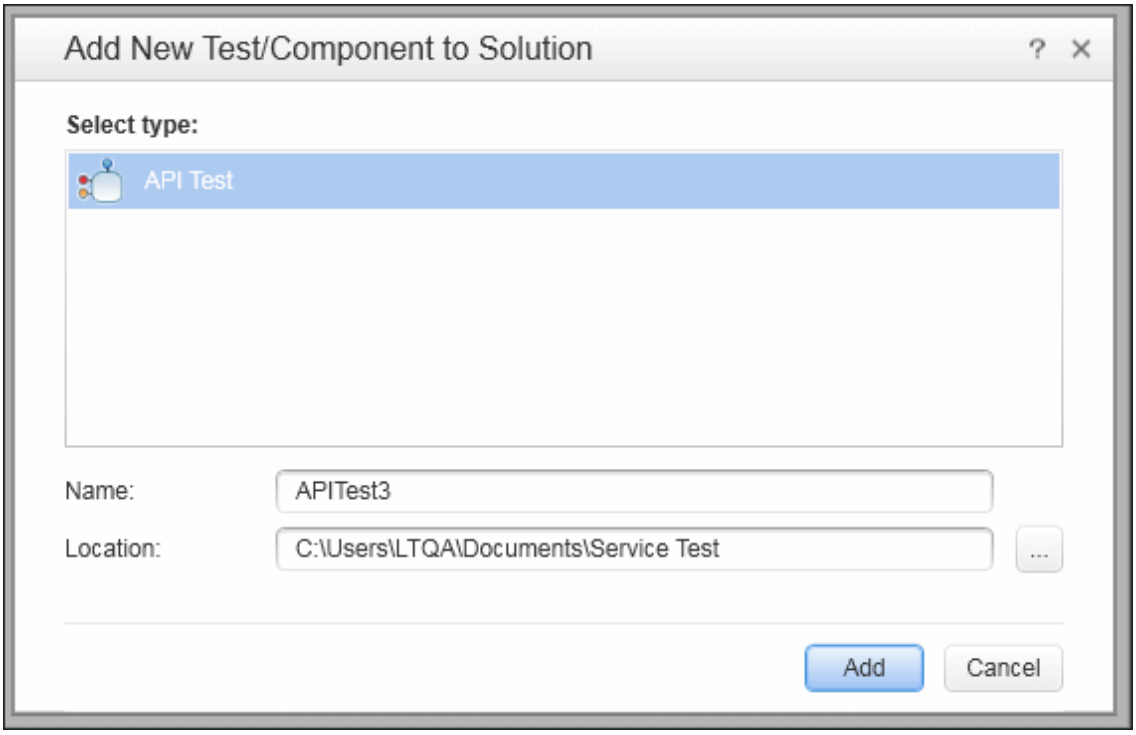
Add Test/Component to Solution Dialog Box

This dialog box enables you to add new tests or components to a solution.

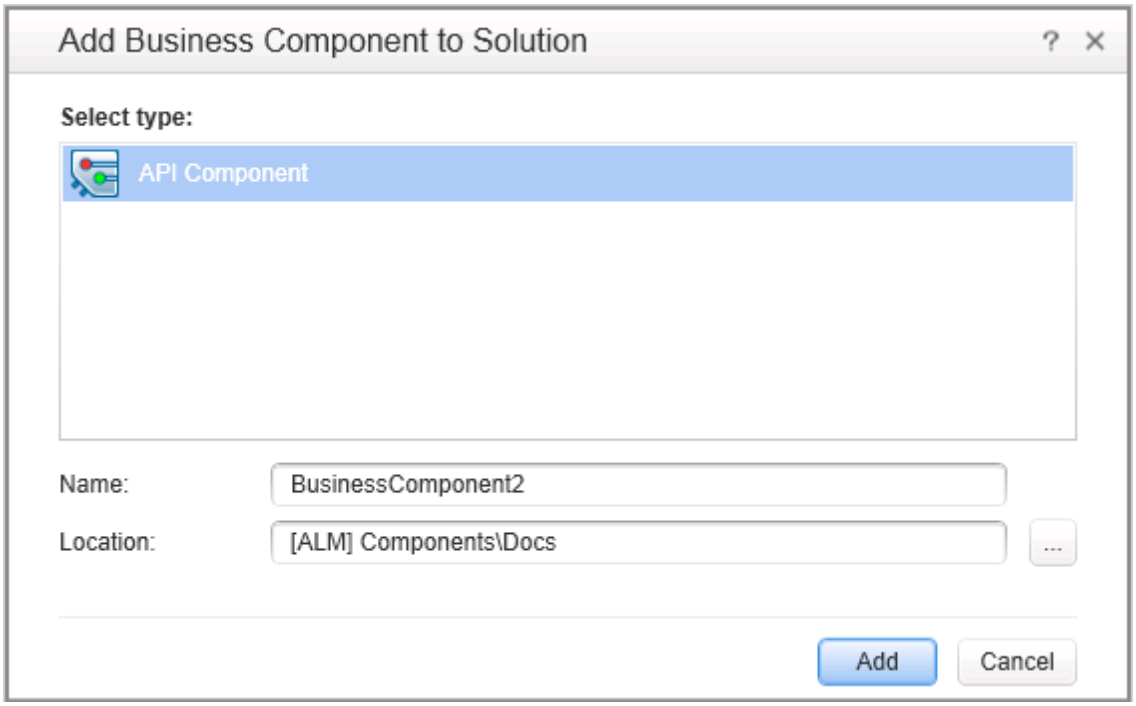
This section describes the following dialog boxes:

- Add New Test/Component to Solution dialog box
- Add Business Component to Solution dialog box

The image below shows the dialog box to add a new test to a solution (**File > Add > New Test**).



The image below shows the dialog box to add a new component to a solution (**File > Add > New Business Component**).



To access	Do one of the following: • In the "Solution Explorer Pane" (described on page240), right-click the solution node and select **Add New <Document>**. • Select **File > Add New <Document>**. • Click the **Add** button down arrow  and select the type of document.
Important information	• When you add a test to a solution, it is added to the solution in the Solution Explorer and it also opens as a new tab in the document pane. • Solutions are saved in your file system only and cannot be saved in an ALM project. For details, see "Solution Explorer Pane Overview" on page 235. • You can use this command to keep multiple tests or components open simultaneously instead of using **File > New** (which closes all open documents). • When you save a new test in the file system, Service Test suggests a default folder called **Service Test**. For all supported operating systems prior to Windows Vista, this folder is located under your Service Test installation folder. For Windows Vista and later operating systems, this folder is located under `My Documents`.
Relevant tasks	"How to Create and Manage Tests" on page 56

User interface elements are described below:

UI Element	Description
Select type	A list of available tests or components. For tests, you can select: • **APITest** • **APITest - Load Enabled** • **Business Process Test** • **Business Process Flow** For components, you can select**APIComponent**. **Note:** The options available differ according to the license loaded and the software installed on the Service Test computer.
Name	The name of the test. Use a descriptive name that helps you and others identify the test easily. Depending on the location you specify in the **Location** field, Service Test suggests possible names to avoid conflicts with existing tests.

UI Element	Description
Location	The folder of the test in the file system or an ALM project. **Note: (for ALM users)** The location of your test or component file is prefaced by an ALM path, shown as: `[ALM] Components/Subject \<folder name>`.
Application Area (GUI components only)	The folder location for the application area associated with your component. **Note: (for ALM users)** The file system path for your application area file is prefaced by an ALM path, shown as: `[ALM\Resource] Resources\<folder name>`

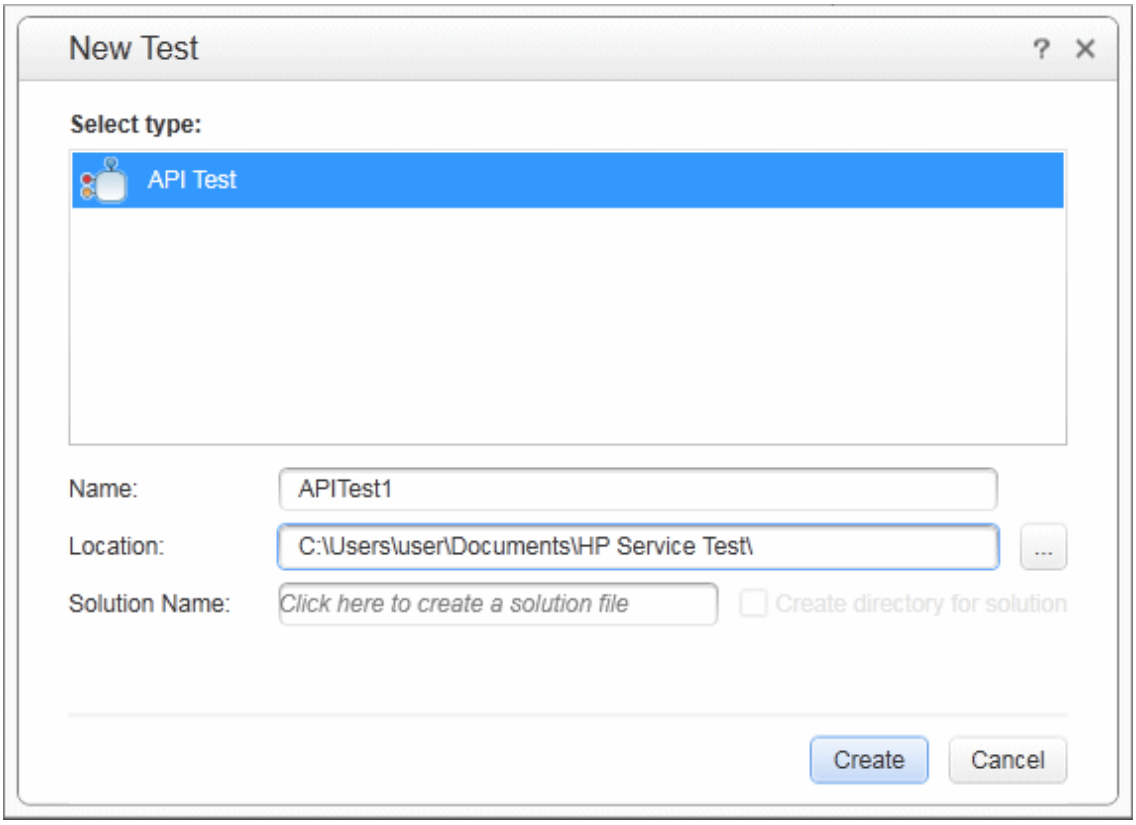
New <Document> Dialog Box

This dialog box enables you to create a new test.

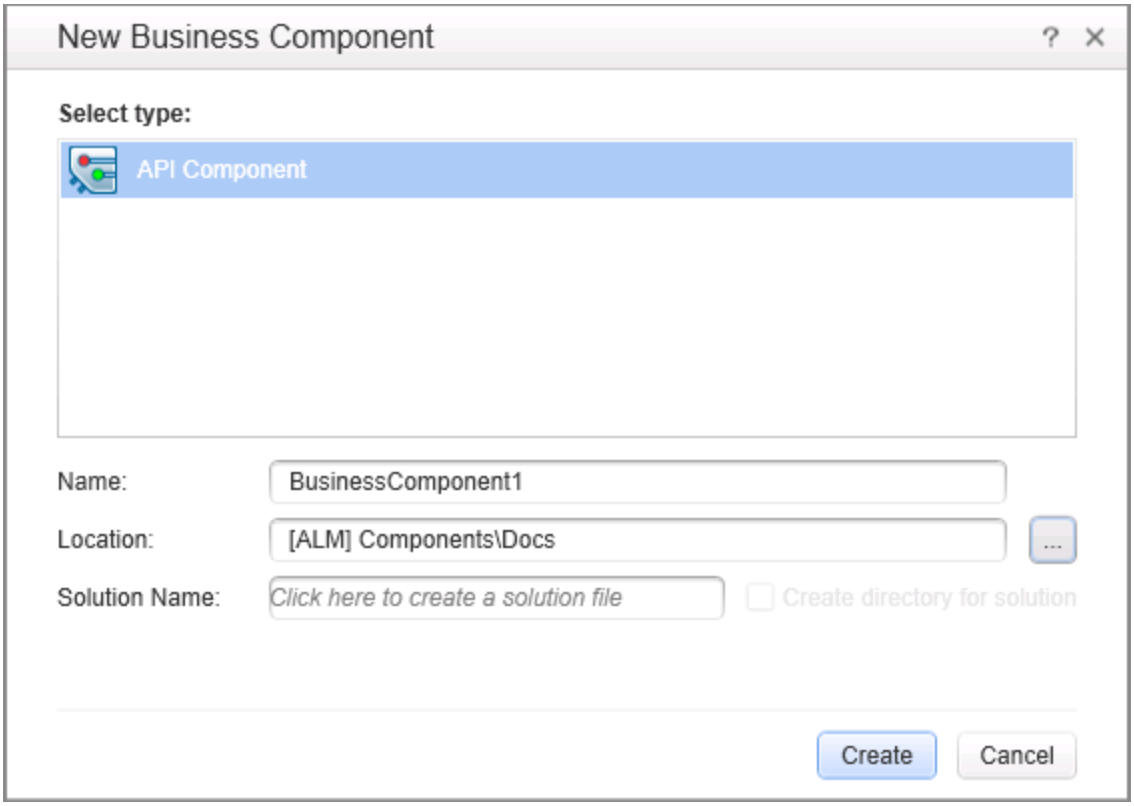
This section describes the following dialog boxes:

- New Test dialog box
- New Business Component dialog box

The image below shows the dialog box to create a new test (**File > New > Test**).



The image below shows the dialog box to create a new component (**File > New > Business Component**).



To access	1. **Prerequisite for ALM users:** Connect to your ALM project if needed. For details, see "ALM Connection Dialog Box" on page 335. 2. Do one of the following: ▪ Select **File > New > Test** (for tests only). ▪ Select **File > New > Business Component** (for components only). ▪ Click the **New** down arrow and select the type of test you want to create.
Important information	• After you create the test, it is automatically saved to the specified location. • When you save a new test in the file system, Service Test suggests a default folder called **Service Test**. For all supported operating systems prior to Windows Vista, this folder is located under your Service Test installation folder. For Windows Vista and later operating systems, this folder is located under `My Documents`. • In ALM, components are saved in the ALM Components module. You cannot save a component directly in the root folder of this module. Create a sub-folder within it, or select an existing sub-folder.
Relevant tasks	"How to Create and Manage Tests" on page 56

User interface elements are described below:

UI Element	Description
Select type	A list of available test or component types. For tests, you can select: • **API Test** • **API Test - Load Enabled** • **Business Process Test** • **Business Process Flow** For components, select**APIComponent**. **Note:** The options available differ depending on the license loaded and the HP applications installed on the Service Test computer.
Name	The name of the test. Use a descriptive name that helps you and others identify the test easily. Depending on the location you specify in the **Location** field, Service Test suggests possible names to avoid conflicts with existing tests.
Location	The folder of the test in the file system or an ALM project. **Note: (for ALM users)** The location of your test file is prefaced by an ALM path, shown as: `[ALM] Components/Subject \<folder name>`.
Application Area (GUI components only)	The folder location for the application area associated with your component. **Note: (for ALM users)** The location of your application area file is prefaced by an ALM path, shown as: `[ALM\Resources] Resources\<folder name>`.
Solution Name	The name of the solution to create that contains your test. If you do not specify a name for a solution, a generic solution called **Solution Untitled** is created in the Solution Explorer, and the test is associated with this default solution. **Note:** All new tests must be part of a solution.

Open/New <Document>/<Resource> Dialog Box

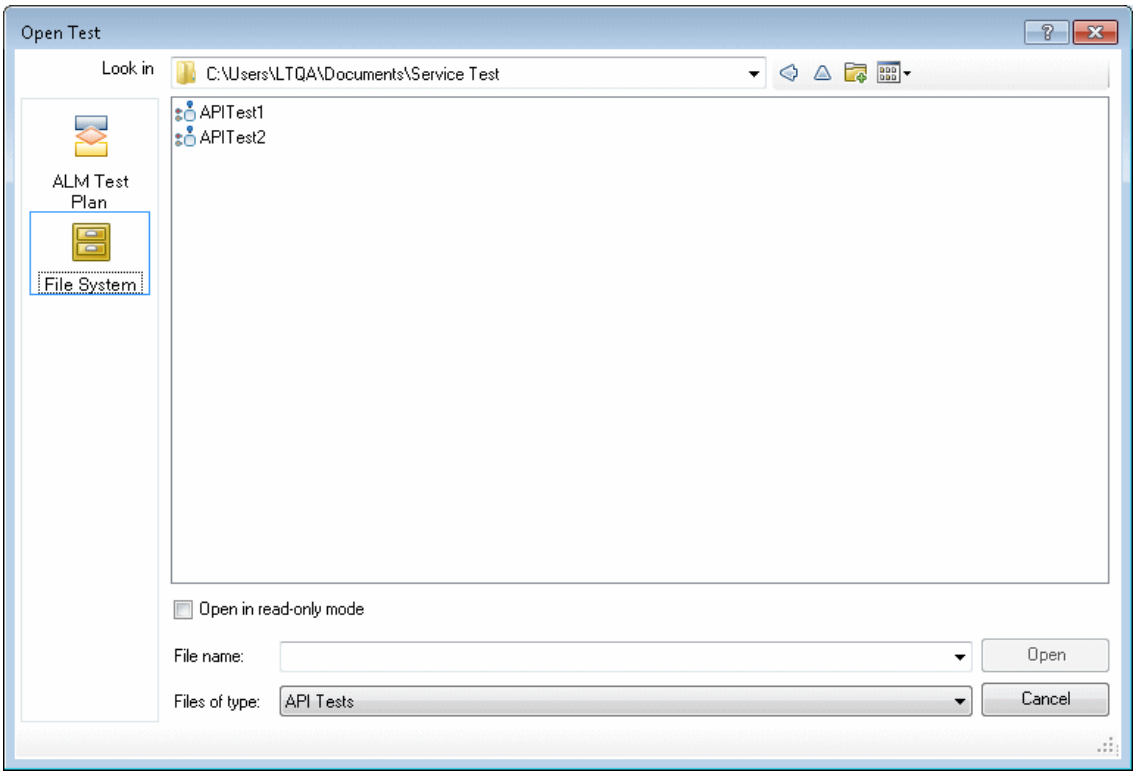
This dialog box enables you to open an existing test from the file system or your ALM project, if Service Test is connected to an ALM project.

Additionally, this dialog box is used when you create a new solution, enabling you to specify its location.

This section describes the following dialog boxes:

- New Solution dialog box
- Open Solution dialog box
- Open Test dialog box
- Open Business Component dialog box

The image below shows a dialog box when opening a test. Similar options are available when opening other types of documents or resources.



To access	1. **Prerequisite for** ALM **users:** Connect to your ALM project if needed. For details, see the "ALM Connection Dialog Box" on page 335. 2. Use one of the following: If you are creating a new solution: • Select **File > New > Solution.** If you are opening a test: • Select **File > Open** and select your test. • Click the **Open** button and select your test.

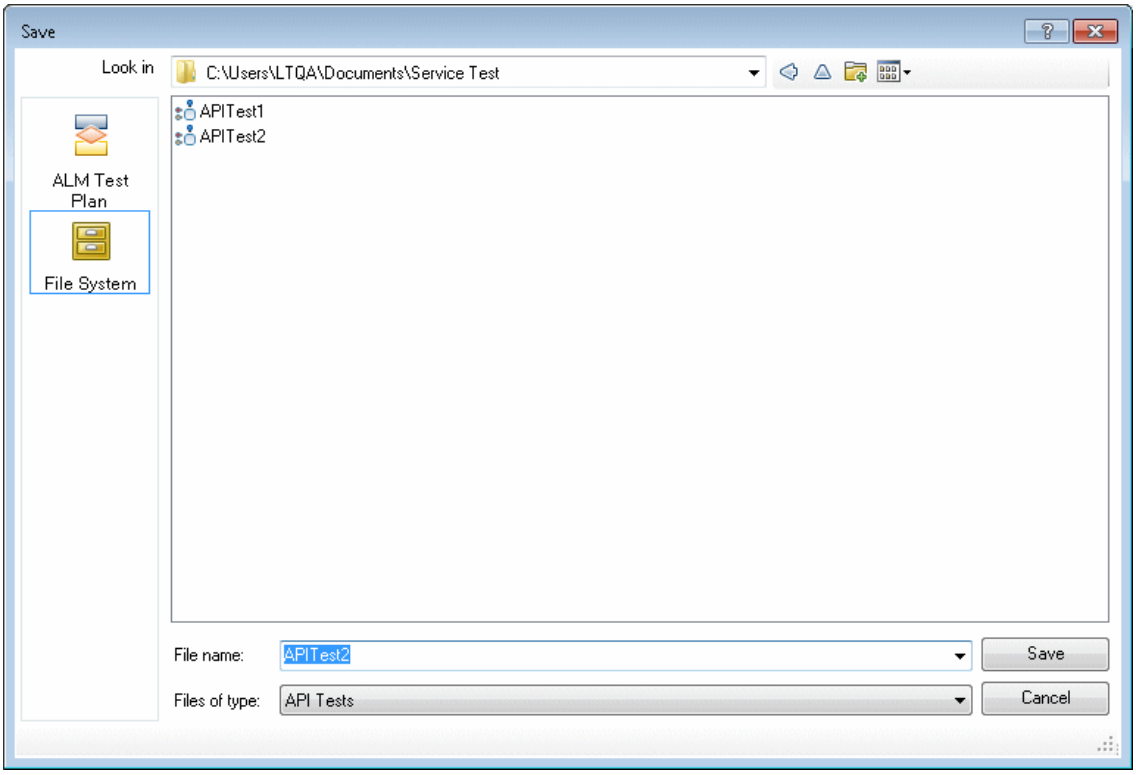
Important information	• When you create a new test in the file system, Service Test suggests a default folder. For all supported operating systems prior to Windows Vista, this folder is located under your Service Test installation folder. For Windows Vista and later operating systems, this folder is located under **My Documents**. • After you create the document, it is automatically saved in the specified location. • When you open a test in read-only mode, the title bar displays **Read Only** and all tabs associated with that test (for example, actions in a test) display a lock icon. • Each time you open a test using **File > Open**, it replaces and closes any tests or solutions currently in use. • You can also open a recently used test or component by selecting from the **Recent** files in the **File** menu. If you select a test or component when you are not connected to an ALM project, or if you select a component that is stored in a different ALM project, Service Test displays a message asking you if you want to connect to that project. • If you try to open a test and there is an error with loading the test (such as ALM connectivity problems or a changed test name), an error describing the problem is displayed in the Errors pane. • **For Business Process Testing**: The first time that you open a business process test or flow from Service Test, you must connect as a user with administrator privileges on the computer on which you are connecting. • When trying to open a test saved in Service Test, version 11.10 or 11.20, Service Test prompts you to upgrade the test using the Batch Upgrader tool (described on page 61). If the test or component was created in Service Test 11.00, you must first open and save it in Service Test 11.10 before you can upgrade it using the Batch Upgrader. • If you try to open a test or component last saved in a version of QuickTest earlier than 9.5, an error message is displayed in the Service Test Error Pane. You must first open it in QuickTest 10.00 or 11.00 to upgrade it. Assets last saved in QuickTest 10.00 or later can be opened in Service Test.
Relevant tasks	"How to Create and Manage Tests" on page 56

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

UI Element	Description
<sidebar>	The location in which the test is stored, for example `File System` or`ALM Test Plan`. **Note:** • You may need to select a different location before browsing to your test. • Business components are always stored in `ALM Components`, and application areas are always stored in `ALM Resources`.
Look in	The path for the test. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the file area to navigate to the required folder.
<file list area>	The folders and/or tests stored in the current path.
File name	The name of the test selected in the file list area.
Files of type	Filters the list of displayed folders or tests to display the selected file type. If the current folder contains other file types, those files are hidden.
Open in read-only mode	Opens the test in read-only mode. This option enables you to view the test but not modify it. **Note: (for ALM users)** A test also opens in read-only mode if: • You opened a test that is currently checked in to the version control database (for projects that support version control). • You opened a test that is currently checked out to another user (for projects that support version control). • You opened a test from an earlier version of ALM and the document has not yet been updated to the current format.
Open	Opens the selected test.

Save <Resource>/Save <Document> As Dialog Box

This dialog box enables you to save a copy of an existing test with another name. You can save the test in the file system or your ALM project, if Service Test is currently connected to an ALM project.



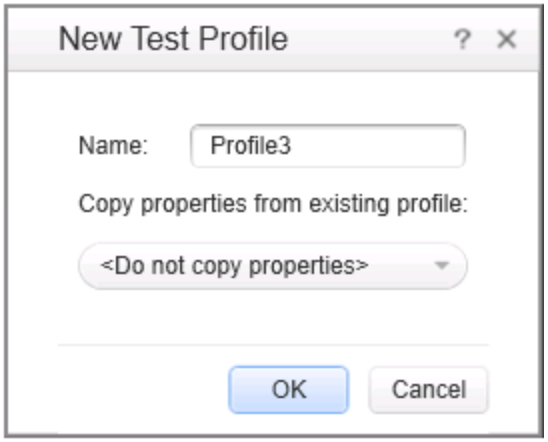
To access	Connect to ALM, if relevant, and then do one of the following: • In the document pane, right-click the document tab, and select **Save <test name> As**. • Select **File > Save <test name> As**. • **File > Save (Other) > Save <component name > As Business Component**.
Important information	• You must use the **Save As** option in Service Test to save a test with another name or create a copy of a test. You cannot copy a test or change its name directly in the file system or in ALM. • If changes are made to an existing test, an asterisk (*) is displayed in the title bar until the test is saved.
Relevant tasks	"How to Create and Manage Tests" on page 56

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

UI Element	Description
<sidebar>	The location in which to save the test, for example, `File System` or `ALM Test Plan`. **Note:** For business components the location is always `ALM Components`, and for and application areas it is always `ALM Resources`.
Look in	The path for the folder in which to save the test. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the file area to navigate to the required folder.
<file list area>	The folders and/or tests stored in the current path.
File name	The name of the test. Use a descriptive name that helps you and others identify the test easily.
Files of type	Filters the file types displayed in the file list area. You can choose to display a specific type of file or all file types.

New Test Profile Dialog Box

This dialog box enables you to create a new profile for your test with pre-defined user variables.



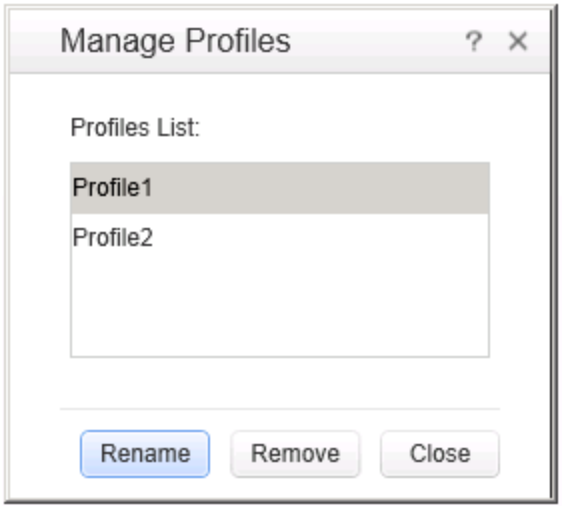
To access	1. Create or open a test or component. 2. Select the **Start** or **End** step in the canvas. 3. In the Properties pane, open the **User Variables** view . 4. Click the **Add Profile** button .
Relevant tasks	"How to Define Test Properties or User/System Variables" on page 59

User interface elements are described below:

UI Element (A-Z)	Description
Name	A name for the profile as it will appear in the profile list drop down list.
Copy properties from existing profile	A drop down list of existing profiles. **Default:** `Do not copy properties`, creates a new empty profile with the user variables of the active profile, without values.

Manage Profiles Dialog Box

This dialog box enables you to remove and rename user variable profiles.



To access	1. Create or open a test or component. 2. Select the **Start** or **End** step in the canvas. 3. In the Properties pane, open the **Test Variables** view. 4. Add additional profiles using **Add Profile**. 5. Click the **Manage Profiles** button.
Relevant tasks	"How to Define Test Properties or User/System Variables" on page 59

User interface elements are described below:

UI Element (A-Z)	Description
Profiles List	A list of all the profiles defined for this test.
Rename	Allows you to rename the selected profile.
Remove	Deletes the selected profile.

Batch Upgrader Command Line Options

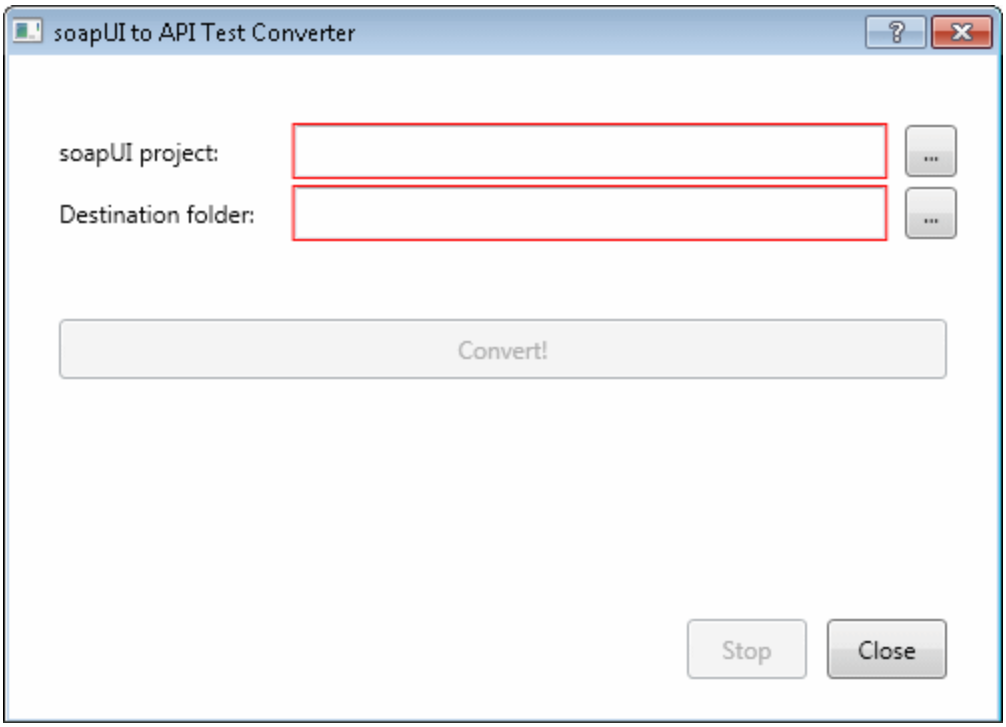
You operate the batch upgrade tool from the command line. For task details, see "How to Upgrade Tests Using the Batch Upgrader " on page 61. The following table describes the command line options:

UI Elements (A-Z)	Description
/ALM	Indicates that ALM connection information will follow.
/log	Indicates that the log will be written to an alternate file. By default, log files are store in the `<Service Test installation>\bin\logs` folder.
/login	Indicates that login information will follow.
/report	Instructs the upgrader tool to generate a summary report.
destination	• For tests on the File system: The full UNC path of the folder in which to store the tests after the upgrade. • For tests in ALM: a target folder path under the Test Plan module, in which to store the upgraded tests. **Note:** • The **destination** value must be a local or remote folder with the same write access permissions as the **source** path. • The destination folder should be an empty folder that does not include any tests. • If you do not specify a destination folder, the tests will be upgraded in the source folder, overwriting the originals.
domain	The name of an ALM domain in which the source tests are stored.
logfile	The full path of the file to which the log will be written.
project	The name of an ALM project in which the source tests are stored.
reportfile	The full path of an existing folder, with the file name including an extension, to where the summary report should be written. For example, `c:\logs\MySUmmary.txt`.
source	• **For tests on the File system:** The full UNC path of the folder containing the tests to be upgraded. • **For tests in ALM:** a source folder path under the Test Plan module).

UI Elements (A-Z)	Description
url	The URL of an ALM instance to which to connect in the following form: `http://{instance_domain}:8080/qcbin`
username, password	For ALM mode, the username and password with which to log in to the ALM project.

soapUI to API Test Converter

Using the soapUI Converter tool, you can convert soapUI tests to Service Testtests.



To access	Do one of the following: • Select **soapUI to API Test Converter** from the product's **Tools** section under the **Start** menu. • Run the `soapUI2APITest.exe` file from the product's `bin` folder.

User interface elements are described below:

UI Element (A-Z)	Description
soapUI project	The absolute path of the `.xml` file representing the soapUI project.
Destination folder	The target folder in which to store the converted test.
Convert	Begins the conversion process.

Command Line Syntax

You can also run the soapUI converter utility from the command line. Run the following command in the command line:

```
soapUI2APITestCMD.exe /<source soapUI_file> /destination
<destination directory>/
logs <log_directory>
```

Command line options are described below:

Command Line Switch	Description
/source	The absolute path to the soapUI file with an `.xml` extension, to be converted.
/destination	The absolute path of the folder to where the created API tests will be written.
/logs (optional)	The absolute path of the folder in which to write the log file. If this option is omitted, the log file is written to the destination folder.
-? or /?	Show the parameters and their usage. For example: `soapUI2APITestCMD.exe -?`

Troubleshooting and Limitations - Service Test Testing Documents

This section describes troubleshooting and limitation for working with tests:

Troubleshooting and Limitations - Opening and Saving Tests

This section describes general troubleshooting and limitation for creating, opening and saving testing documents, as well as opening tests or components last modified in previous versions of QuickTest or Service Test.

Folder names

It is not possible to create a Service Test test in a path containing a **'='** character.

Opening documents stored in ALM

Opening a document from the Recent Files.

You can open components from the **Recent** files list in the **File** menu. If you select a test located in an ALM project, but Service Test is not currently connected to the correct project, the Connect to ALM Project dialog box opens and displays the correct server, project, and the name of the user who most recently opened the test or component on this computer.

This dialog box also opens if you choose to open a test or component that was last edited on your computer using a different ALM user name. You can either log in using the displayed name or you can click **Cancel** to stay logged in with your current user name.

Opening a solution containing test documents stored in ALM.

Testing documents that are part of the same solution cannot be stored in different ALM projects, domains, or servers.

Opening documents last modified with previous versions of Service Test

- When trying to open a test saved in Service Test, version 11.10 or 11.20, Service Test prompts you to upgrade the test using the Batch Upgrader tool (described on page 61). If the test or component was created in Service Test 11.00, you must first open and save it in Service Test 11.10 before you can upgrade it using the Batch Upgrader.
- When upgrading a test from Service Test 11.20, Service Test 11.50 does not retain the Security Settings.

  Workaround: In Service Test 11.20 or earlier, save the Security Scenario to an `.stss` file, and import this file for your service when you upgrade it. For details, see, "Security Scenarios Overview" on page 541.

Opening and saving documents in Windows 7

- The Libraries feature of Windows 7/Windows 8 is not supported. If you attempt to open a test from a library location or save a test to a library location, Service Test may behave unexpectedly.

  Workaround: Browse to your saved tests using standard file paths, even if they are included in a library, such as the Documents library.

  For example, to browse for a test that is stored in the default Service Test folder, use:

  ```
  Computer > C: > Program Files > HP Software > HP Service Test>
  Tests > <TestName>
  ```

  Instead of:

  ```
  Libraries > Documents > HP Service Test > <TestName>
  ```

- If you work with Service Test on a Windows Vista, Windows 7, or Windows 8 operating system with User Account Control (UAC) enabled, and you open a test from a protected location (such as `Program Files`), it is opened in read-only mode and a message is displayed that you do not have permissions to open it in read-write mode.

Unexpected read-only files

Documents that were saved in the file system in system folders like (`%Windir%` or `Program files`) while UAC was disabled can be opened only in read-only mode when the UAC is set to ON.

Workaround: Copy the asset to another location and open the files from the new location.

Multilingual support

Names and paths of tests and resources are not Unicode compliant and therefore should be specified in English or in the language of the operating system.

Troubleshooting and Limitations - Upgrading Service Test Tests

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

- When upgrading a test from version 11.00 (via 11.10), Service Test 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 when you upgrade it. For details, see "Setting Security Overview" on page 540.

- Tests last modified in Service Test 11.20 that contain a `Call QuickTest Professional Test` step, cannot be opened in Service Test 11.50.

- In certain instances, if the upgrader was unable to upgrade the test, you may need to modify the code to make it compatible with the current version:

- 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
        ...
```

- The `args` variable is no longer supported.

  Workaround: Change the `args` variable name to the concrete Activity name as it appears at the top of the Properties pane. For example, change `args.Output.outStr = args.Input.inStr;` to `CodeActivity8.Output.outStr=CodeActivity8.Input.inStr.`

- The soapUI to API Test Converter tool does not support links between steps. If you convert a soapUI test with linked Web service calls, the links will not be transferred to the API/Service Test test.

Part 2: Service Test Panes

Chapter 4: Bookmarks Pane

This chapter includes:

Concepts

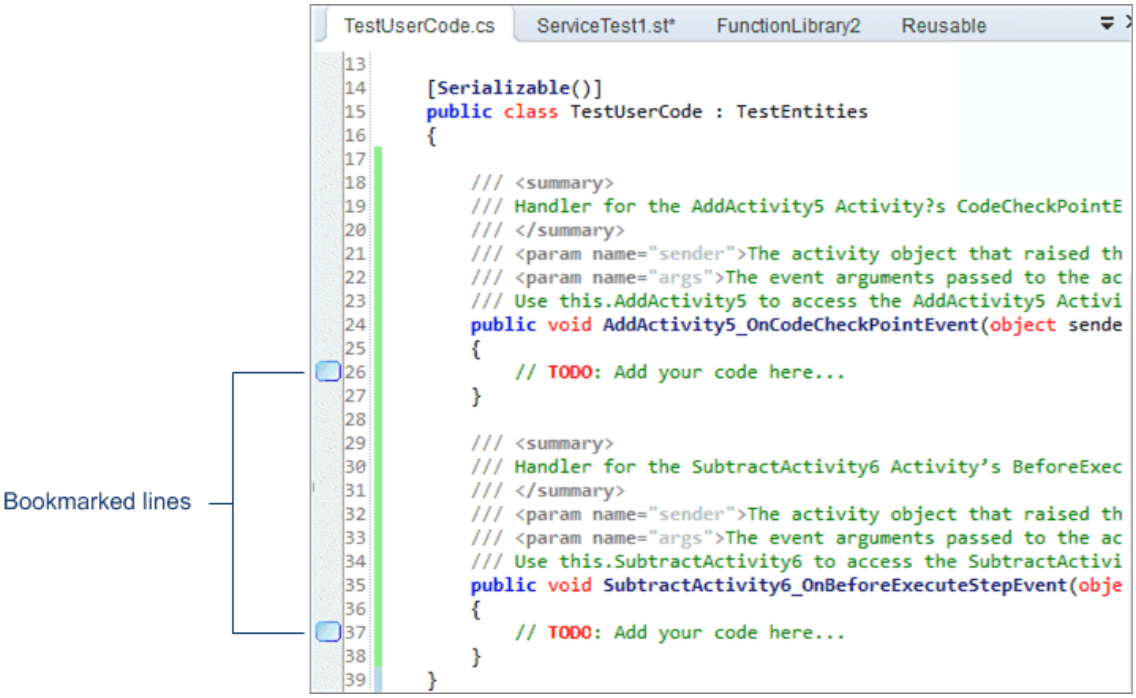
Bookmarks Overview

You can use bookmarks when editing user code files in the Editor. Bookmarks enable you to navigate between sections of your document more easily. The bookmarks are saved on your file system on a per-user basis.

When you assign a bookmark, an icon is added to the left of the selected line of your document.

Example

In the following example, bookmarks have been added to an event handler:



The Bookmarks pane displays a list of all bookmarks added in any document connected with the open test. Using the Bookmarks pane, you can:

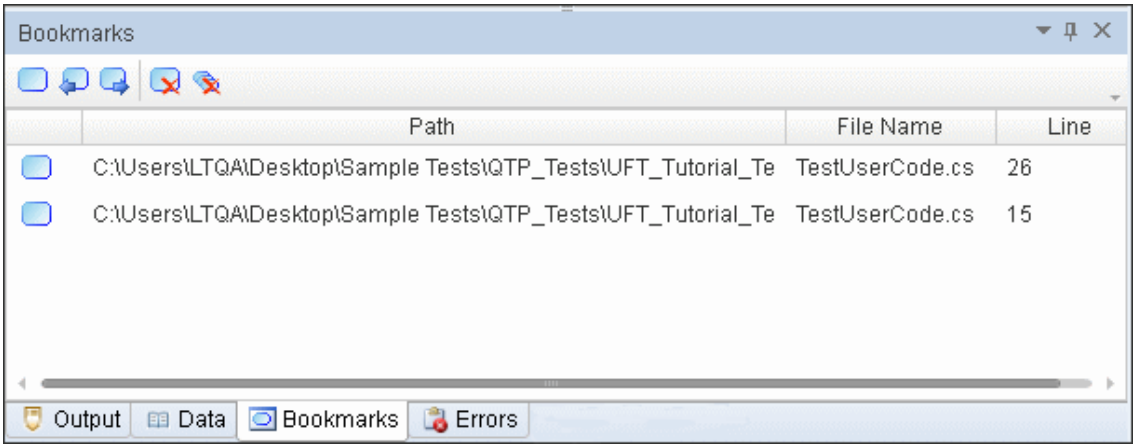
- Add a new bookmark.
- Navigate to the location of the bookmark in your document.
- Navigate between consecutive bookmarks in the pane.
- Delete an individual bookmark.
- Clear all bookmarks.

For task details, see "How to Use Bookmarks in the Editor" on page 138.

Reference

Bookmarks Pane User Interface

The Bookmarks pane displays the location of bookmarks in your user code files and enables you to navigate to these bookmarks.



To access	Select **View > Bookmarks**.
Important information	• All bookmarks added to user code file included in a saved solution are maintained after you close and reopen the pane, document, or Service Test. • Bookmarks are saved per solution, even if multiple users are working on the same tests. • When working with documents saved in ALM, bookmarks are maintained until you close Service Test.
Relevant tasks	"How to Use Bookmarks in the Editor" on page 138
See also	"Bookmarks Overview" on previous page

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

UI Element (A-Z)	Description
<bookmarks list>	All bookmarks connected with the open solution are displayed in the Bookmarks pane. Even if a test is not open in the document pane, its bookmarks are still displayed in the Bookmarks pane. Bookmarks are removed from the pane when the solution is closed. **Example:** When you open a solution, bookmarks in any relevant user code files are displayed. For each bookmark in the list, Service Test displays the location of the bookmark's document, the file name, and the line location of the bookmark. For documents stored on ALM, the location of the local temporary copy is displayed. Bookmarks inserted in a user code file or user code document are kept with the test when it is saved in an ALM project. **Note:** You can double-click the bookmark line to navigate directly to the relevant line in your document. If a bookmark's document is not open, Service Test opens the relevant document.
	**Insert/Remove Bookmark.** Inserts a bookmark in the current event handler or user code file in the line where the cursor is currently located.
	**Previous Bookmark.** Navigates to previous bookmark in the pane.
	**Next Bookmark.** Navigates to next bookmark in the pane.
	**Delete.** Deletes the currently selected bookmark from the relevant document and from the list displayed in the pane.
	**Clear All Bookmarks.** Deletes all bookmarks you have added from all documents and from the list displayed in the pane.

Chapter 5: Data Pane

This chapter includes:

Concepts

Data Pane Overview

Service Test enables you to insert and run steps that are driven by data displayed in the Data pane.

You can designate an Excel spreadsheet, a local data table, an XML file, or results of a database query as data to be used for your test.

Using the Properties pane, you can insert data in your test step's input and output values. Using these parameters enables you to create a **data-driven** test.

Tip: If Service Test is integrating with ALM, you can use the data awareness features to:

- use a different data resource for each run iteration
- use the same data resource in multiple tests
- perform other tasks

For details, see "Data Awareness in ALM" on page 324 and "How to Data Drive a Test Using Data from ALM" on page 328.

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 different 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.

- "Add an Excel data source" below
- "Add an XML data source" on next page
- "Add a database data source" on next page
- "Create a new child relation" on page 94
- "Add a local data table" on page 94
- "Attach a data source to the test flow" on page 94

Add an Excel data source

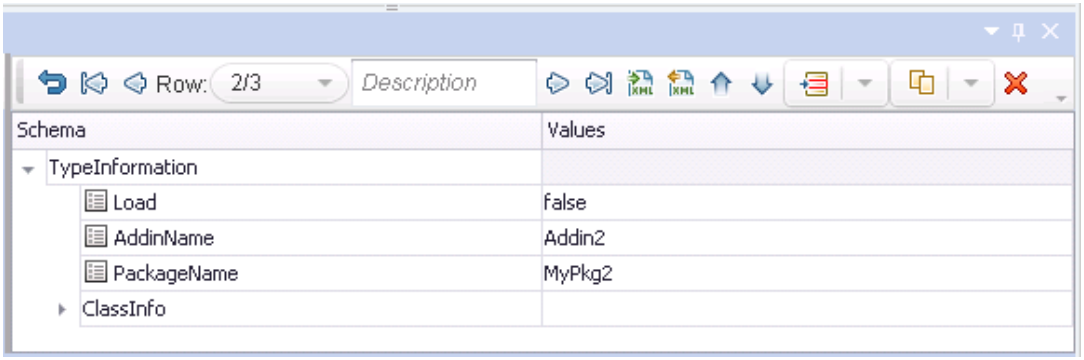
1. Make sure the Data pane is visible. If not, select **View > Data**.
2. In the Data pane, click the **New Data Source** down arrow  and select **Excel**. The "New/Change Excel Data Source Dialog Box" (described on page 98) opens.
3. Browse to the file and indicate if the first row is a header row.
4. Specify a name for the data source. If you do not provide a name, Service Test sets it to the Excel file name after you navigate to the file.
5. Select whether to link to the Excel in its original location or create a local copy. The advantage of making a local copy is that it becomes portable with the test. However, any updates to the data at its original location will not be not reflected in the migrated test.
6. Select **Allow other tools to override the data** option to enable integration with ALM's Data Awareness. This allows you to overwrite the data from the imported Excel file with values from an ALM Data Resource. For details, see "Data Awareness in ALM" on page 324.

   In addition, if you call a test or action with data assigned to its steps, this option allows you to edit the data in the called test or action. For details, see "Actions Using the Data Table" on page 511.
7. Click **OK**.
8. In the Data pane, select the data source under the **Excel** node. In the right pane, you can view and edit the values.

Tip: You can replace an existing Excel data source with a new file. For user interface details, see "New/Change Excel Data Source Dialog Box" on page 98.

Add an XML data source

1. Make sure the Data pane is visible. If not, select **View > Data**.
2. In the Data pane, click the **New Data Source** down arrow and select **XML**. The "New XML Data Source Dialog Box" (described on page 100) opens.
3. Provide a **Data Source name**.
4. Select the entity upon which to base the data source:
   - **Schema file:** Browse to an `.xsd` file.
   - **XML file:** Browse to an `.xml` file.
   - **Use existing**: Browse to an existing XML-based data source from another test.
5. Click **OK**.
6. Select the data source's node in the Data pane.

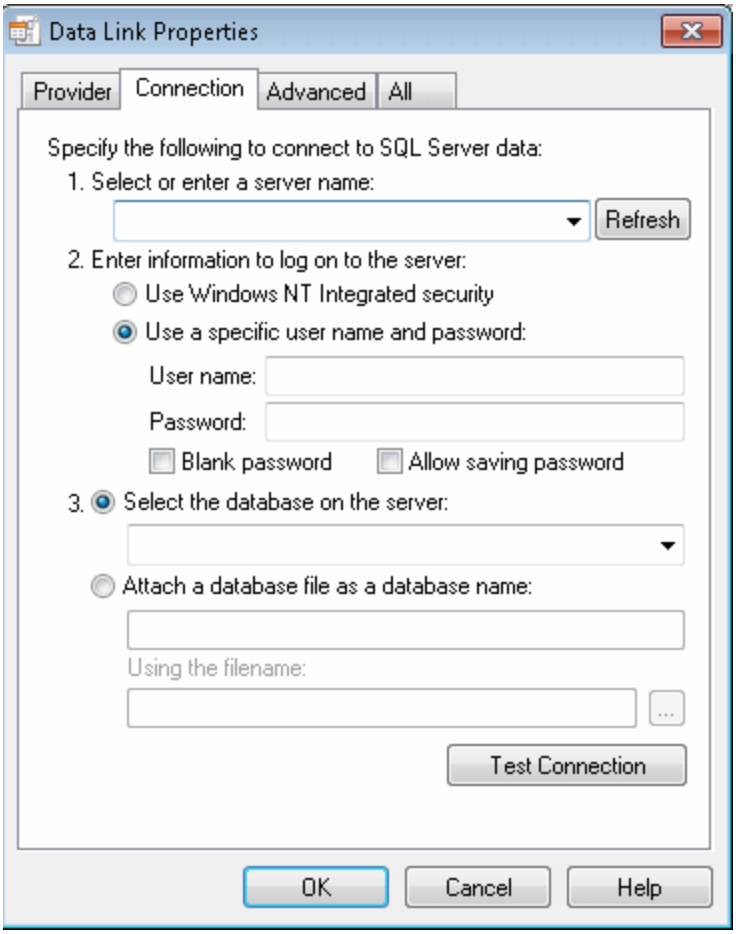


7. Edit the XML data in the grid. Click the **Add rows** button to add row values. For user interface details, see "Data Pane" on page 96.
8. To load XML values, click the **Load data from an XML file** button.

Add a database data source

1. Make sure the Data pane is visible. If not, select **View > Data**.
2. In the Data pane, click the **New Data Source** down arrow and select **Database**. The "Add New Database Data Source Wizard" (described on page 102) opens. The **Set Database Connection** page opens.
3. For an **OleDB** type connection:
   a. Select **OleDB** as the **Connection type**.
   b. Set the connection string in one of the following ways:
      - Paste an OLE DB connection string into the text area.
      - Select a previously defined connection string from the drop down.
      - Click the **Build Connection String** button to use Microsoft's Data Link Properties

dialog box. For details, click the **Help** button in the Data Link Properties dialog box.



c. Click **Test Connection** to verify that the connection is valid.

d. Click **Next**. This button is only available for valid connections.

4. For an ODBC type connection:

a. Select **ODBC** as the **Connection type**.

b. Click the **Build Connection String** button to select a DSN type. For details, see "Select Data Source Name Dialog Box" on page 105.

c. Select a DSN type and provide the necessary credentials. To create or edit a DSN entry, click the **Manage ODBC Data Sources** button. For details, click the **Help** button in the ODBC Data Source Administrator dialog box. After you close the administrator dialog box, click the **Reload DNS list** button to refresh the list.

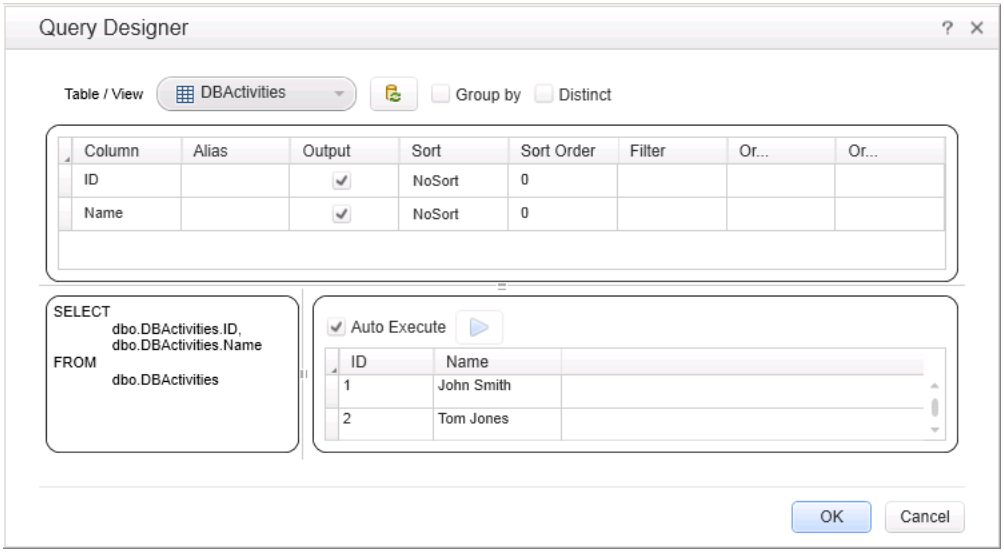
d. Test the connection. After the confirmation, click **OK**.

5. In the **Set SQL Statement** page:

a. Provide a unique name for the data source, not used by another data source. If you do not provide a name, the data source is automatically assigned the table name after you build the query.

b. Provide an SQL statement in one of the following ways:

- Paste an existing statement into the text area.
- Select a previously defined statement from the drop down history.
- Click the **Build Query Statement** button to open the "Query Designer Dialog Box" (described on page 456). The Query Designer allows you to create a basic SQL query, with a single table or view. For complex queries, such as those using advanced SQL features and calculations, create the query in your environment and paste it into the text area.



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. Click the **Reset Query** button to discard the changes.

d. In the wizard's **Set SQL 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 Properties pane's "Database Data Source Properties Tab (Properties Pane)" (described on page 191). By default the

Data pane shows the first ten rows of the query data. To change the number of displayed rows, use the "General Pane (Options Dialog Box > API Testing Tab)" (described on page 271).

f. To update the data at a later stage, click **Refresh** in the Data pane. The data is not automatically refreshed—the displayed data is the result of the query when you created the data source or the last time you clicked the **Refresh** button.

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 of the above data sources containing the foreign and primary keys. This can also be a single Excel file with two worksheets.
2. In the Data pane, select the table that you want to designate as a parent.
3. Click in the Properties pane to open the "Data Source Properties Tab (Properties Pane)" on page 190 (described on page 190).
4. Click the **Add** button. The "Define New/Edit Data Relation Dialog Box" (described on page 637) opens.
5. Specify the details of the new relation. You can specify a relation to a different data source type, provided that it is either an **Excel**, **Local Table**, or **Database** type data source.
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 a local data table

1. Make sure the **Data** pane is visible. If not, select **View > Data**.
2. In the Data pane, click the **New Data Source** down arrow and select **Local Table**. The "New Local Table Data Source Dialog Box" (described on page 106) opens.
3. Click the **Add** button to create a column in the data table.
4. In the columns list, specify a column name and description. Select a data type from the drop down list.
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 the **Remove** button to delete an unwanted column from the data table.
8. Click **OK**.
9. In the Data pane, select the table's node in the left pane and move within the table using the keyboard arrows. Type within the cells to manually set values.

Attach a data source to the test flow

This step lets you attach a data source to the Test Flow or current loop, even without linking to a

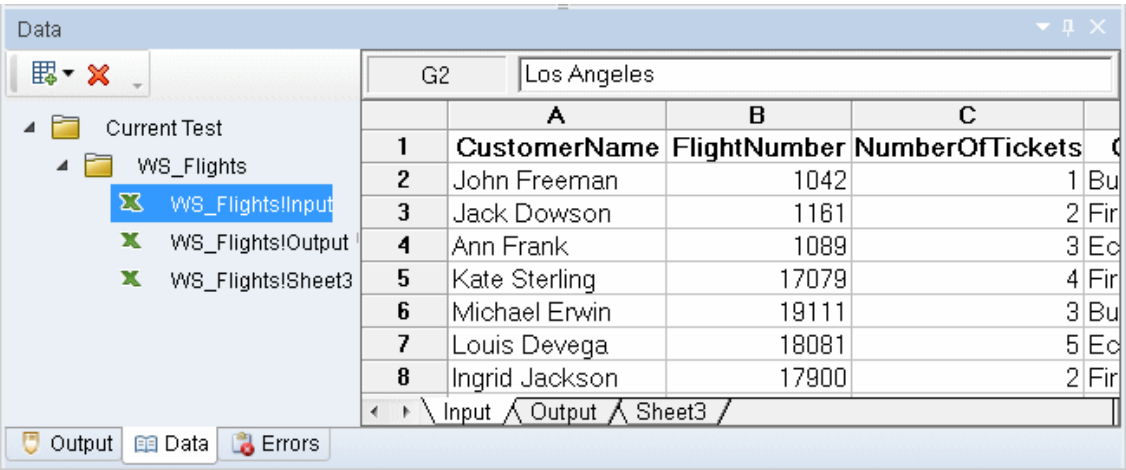
property. This will allow you to easily access the data source and use it with an event handler.

1. In the Data pane, select the data source.
2. In the canvas, select the Test Flow or loop frame.
3. Select the "Data Sources Tab (Properties Pane)"  in the Properties pane (described on page 189).
4. Click **Add**. The "Attach Data Source to Loop Dialog Box " (described on page 629) 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 Properties pane.

Reference

Data Pane

The Data pane enables you to add, view, and edit data sources for use as property values when running tests.



To access	1. Create or open a test or component 2. Select **View > Data** or select the Data pane tab.
Important information	In order to see the contents of a data source, click its node in the left pane.
Relevant tasks	"How to Create Data Sources" on page 90
See also	• "How to Assign Data to Test Steps" on page 621 • "How to Set the Navigation Properties" on page 622

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

UI Element	Description
<data source tree>	A hierarchy of the data sources and their subnodes, displayed in the left pane. There are separate nodes for the test and action data sources: Current Test, <action1>, <action1>, and so forth. For Excel files, each sheet represents another set of properties.

UI Element	Description
<data source content>	The data associated with the selected node, displayed in the right pane. You can edit values directly from this pane. If the current test calls another test or action with its own data source, this area shows the data, provided that you enabled the data to be viewed as described below. **Note:** To enable this data to be viewed and edited, click the node in the original location of the data, and enable the **Allow other tools to override the data** option. For details, see "Data Source Properties Tab (Properties Pane)" on page 190.
	**New Data Source.** Creates a new a data source. Expand the drop down to select a data source type: **Excel**, **XML**, **Database**, or **Local Table**.
	**Remove.** Removes the selected data source from the Data pane.

XML Data Source - Toolbar

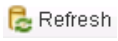
The following controls are available when selecting an XML data source in the Data pane.

UI Elements	Description
	**Go back.** Returns to the previous view.
	**Go to start.** Returns to the first row.
	**Go to the previous row.** Navigates to the previous row (one row up).
Row	The current row number out of the total rows, such as `1/3` and so forth. Use the drop down to move to the desired row.
Description	A title for the row currently displayed.
	**Go to the next row.** Navigates to the next row (one row down).
	**Go to end.** Navigates 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 forwards.
	**Move down.** Moves the current row backwards.

UI Elements	Description
	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.

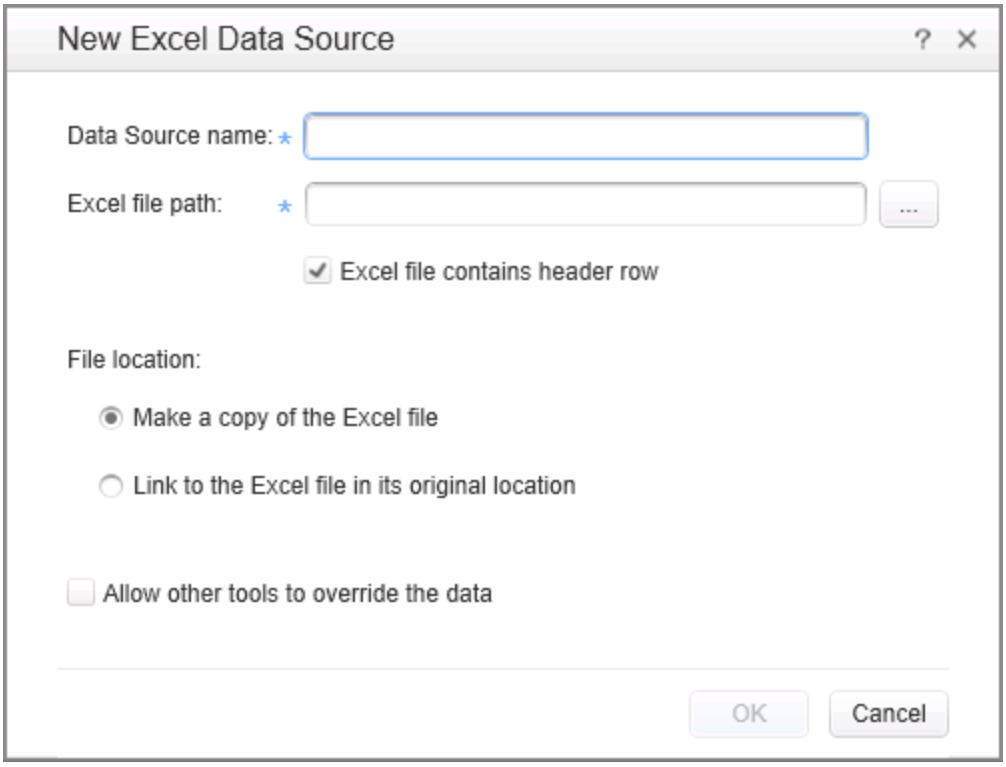
Database Data Source Nodes

The following controls are available when selecting a database data source in the Data pane.

UI Elements	Description
	**Refresh.** Updates the data from the database.
	**Count.** Opens a message box with a row count. By default the Data pane shows the first ten rows of the query data. To change the number of displayed rows, use the "General Pane (Options Dialog Box > API Testing Tab)"(described on 271).

New/Change Excel Data Source Dialog Box

This dialog box enables you to add a new Excel data source to the Data pane or change the underlying file for existing Excel data sources.



To access	Before you open the dialog boxes, make sure that a test or component is in focus in the document pane. **To open the New Excel Data Source dialog box:** 1. Create or open a test or component. 2. Select **View > Data** 3. Click on the **New Data Source** down arrow and select **Excel**. **To open the Change Excel Data Source dialog box:** 1. Select the main node of an existing Excel data source. 2. In the Properties pane, click **Change Excel File**.
Relevant tasks	"Add an Excel data source" on page 90
See also	"Data Link/Relation Message Box" on page 627

User interface elements are described below:

UI Element	Description
Data source name	A name for the data source as it will appear in the Data pane. If no value is specified, it uses the file name. **Note:** This field is read-only in the Change Excel Data Source dialog box.
Excel file path	The complete path of an Excel file on the file system. Use the **Browse** button to locate the file. **Note:** You must use an absolute path for the Excel file path.
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.
File location	The location for storing the imported Excel data: • **Link to the Excel file in its original location.** If the data changes at its source, it affects your test. In addition, if you modify the data in your test, it affects the data at its source. • **Make a copy of the Excel file.** Copies the Excel file locally. If the data changes at its source, it does not affect your test. In addition, the data is portable together with the test.
Allow other tools to override the data	Enables you to overwrite data from the imported Excel file with values from other tools, such as an ALM Data Resource (for versions ALM 11.00 and higher), and GUI or Service Test tests. For details about ALM data resources, see "Data Awareness in ALM" on page 324. **Tip:** If you did not enable this option when you imported the Excel file, you can enable it separately for each data source in the Properties pane. For details, see "Data Source Properties Tab (Properties Pane)" on page 190.

New XML Data Source Dialog Box

This dialog enables you to add new XSD or XML data sources to the Data pane.

To access	1. Do one of the following: ▪ Ensure that a test or component is in focus in the document pane. ▪ In the solution explorer, select a test or component. 2. Create or open a test or component. 3. Select **View > Data**. 4. Click on the **New Data Source** down arrow and select **XML**.
Relevant tasks	"Add an XML data source" on page 91
See also	"Data Pane" on page 96

User interface elements are described below:

UI Element	Description
Data source name	The display name of the data source. If no name is specified, it uses the file name.
Based on	The file on which to base the data table: • **Schema file.** A file with an XSD extension. • **XML file.** A file with an XML extension. • **Use existing.** An existing data resource created within another test. Since this data source is referenced, changes made to it are reflected in all tests using the data source. Use the **Browse** button to navigate to the desired location.

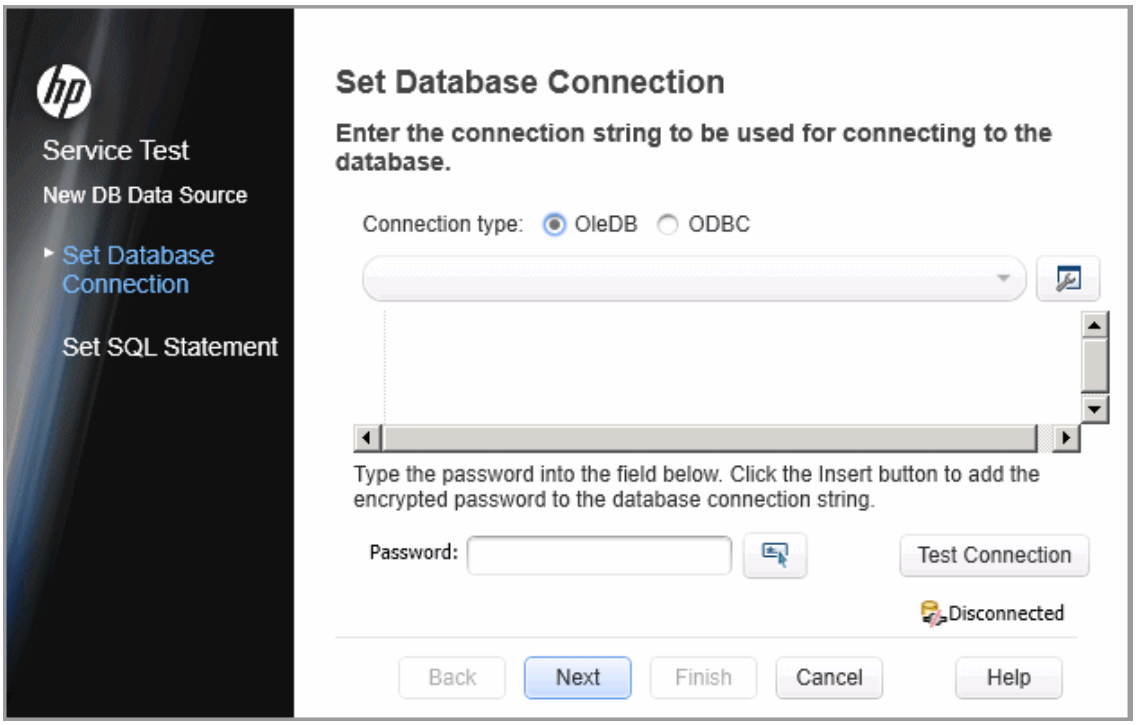
Add New Database Data Source Wizard

This wizard enables you to create a data source by retrieving data rows from a database table. You specify a connection string and an SQL statement to retrieve the data.

To access	1. Do one of the following: ▪ Ensure that a test or component is in focus in the document pane. ▪ In the solution explorer, select a test or component. 2. Create or open a test or component. 3. Select **View > Data**. 4. Click on the **New Data Source** down arrow and select **Database**.
Relevant tasks	"Add a database data source" on page 91
Wizard map	This wizard contains: "Set Database Connection Page" > "Set SQL Statement Page "
See also	"Query Builder Dialog Box" on page 455

Set Database Connection Page

This wizard page enables you to define a connection string for connecting to the database.



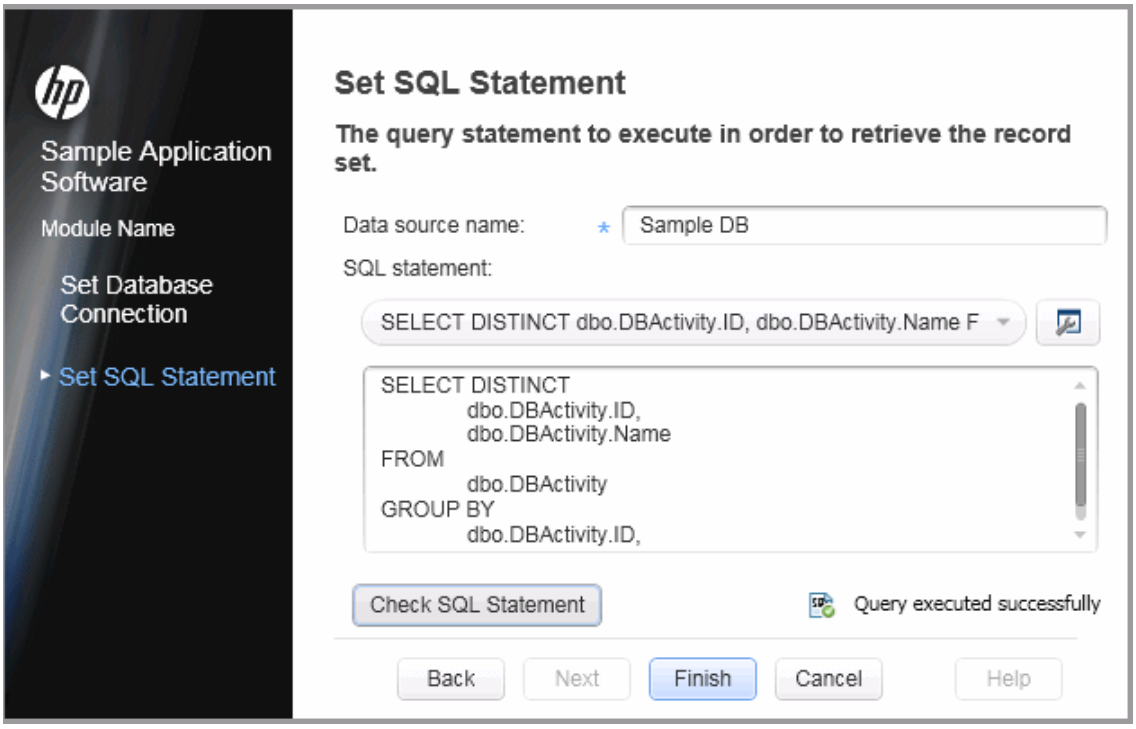
Wizard map	This wizard contains: **Set Database Connection Page** > "Set SQL Statement Page "

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

UI Element	Description
Connection type	The type of database connection: **OleDB** or **ODBC**.
<connection list>	A drop down list of previously defined connection strings.
	**Build Connection String.** • For **OleDB** connections: Opens the Microsoft Data Link Properties dialog box. • For **ODBC** connections: Opens the Select Data Source Name 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**.
Password	The password with which to access the database.
	**Insert.** Inserts the encoded password into the displayed connection string.
Test Connection	Sends the current connection string to the database server to check its validity.
<connection status>	Displays the status of the database connection. If it is a valid connection, the wizard adds the connection string to the list of Connection strings and changes the status to `Connected`  Connected .

Set SQL Statement Page

This wizard page enables you to prepare an SQL statement for retrieving your data.



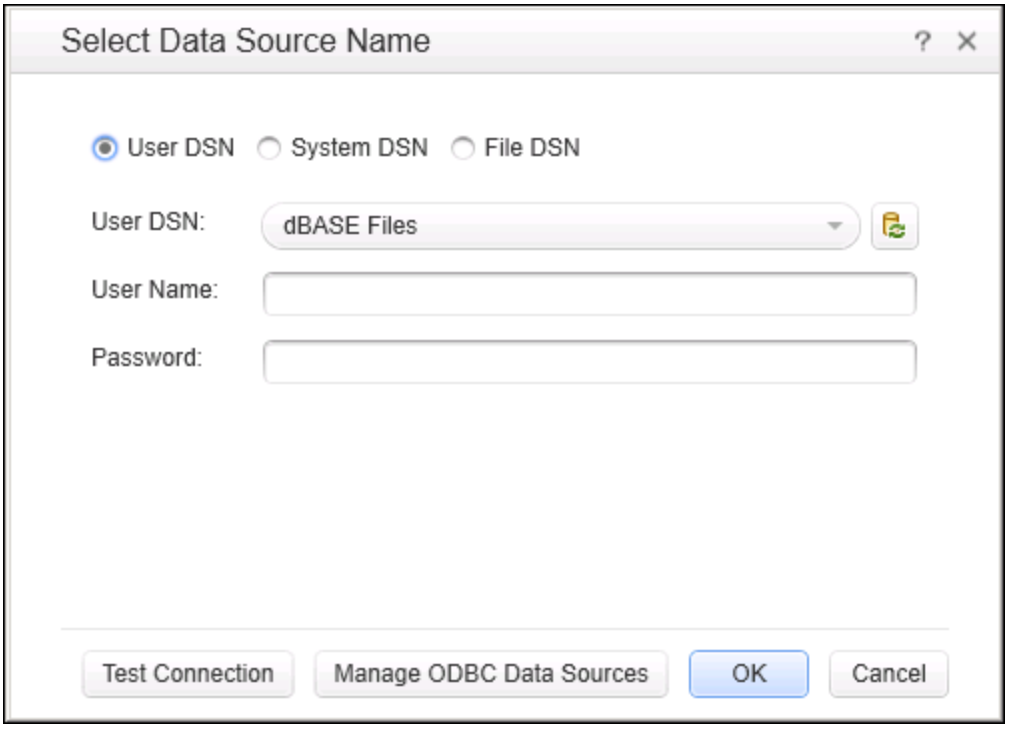
Wizard map	This wizard contains: "Set Database Connection Page" > **Set SQL Statement Page**

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

UI Element	Description
Data source name	The name of the data source as it is be referenced by your test. If no value is specified, it uses the name of the table or view.
SQL statement	A drop down list of the available SQL statements.
	Opens the "Query Builder Dialog Box" (described on 455).
<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**.
Check SQL Statement	Executes the SQL statement shown in the text area and opens the results in a **Query Preview** window.
<query status>	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`.

Select Data Source Name Dialog Box

This dialog helps you build a connection string for an ODBC data source.



To access	Do one of the following: **From the Database wizard:** 1. Choose **Database** in the Data pane to open the wizard. For details, see "Add New Database Data Source Wizard" on page 102. 2. Select **ODBC** as a connection type. 3. Click on the **Build Connection String** button adjacent to the Connection list drop down list. **From the Properties pane:** 1. Add a Database **Open Connection** activity to the canvas. 2. Click the **Input/Checkpoints** tab in the **Properties** pane and click the browse button in the **Connection string** property row. 3. In the "Connection Builder Dialog Box" (described on page 453), select **ODBC** as a connection type. 4. Click on the **Build Connection String** button adjacent to the Connection list drop down.

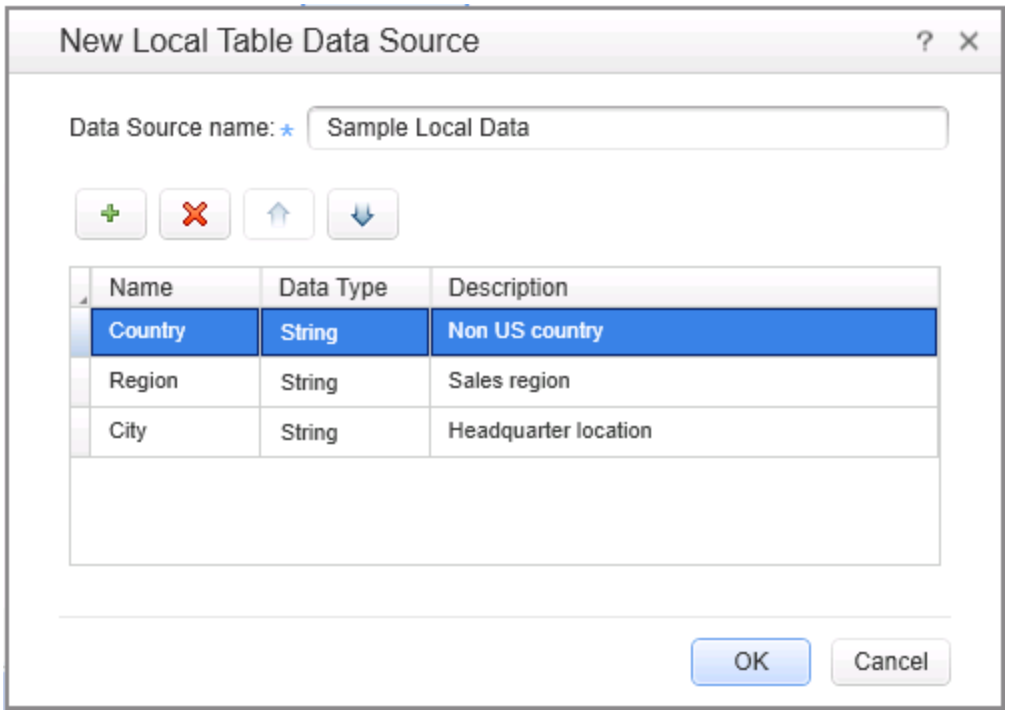
Relevant tasks	"Add a database data source" on page 91
See also	"Add New Database Data Source Wizard" on page 102

User interface elements depend on the type of DSN selected:

UI Element	Description
	**Reload DSN List**. Reloads the items that will appear in the drop down list.
User DSN	Allows you to select a User DSN and provide credentials: • User Name • Password
System DSN	Allows you to select a System DSN and provide credentials: • User Name • Password
File DSN	Allows you to provide the following information for the File DSN or select a file with the required settings. • User Name • Password • File Content
Test Connection	Sends the current DSN information to the database server to check its validity.
Manage ODBC Data Sources	Opens the Microsoft ODBC Data Source Administrator dialog box. This dialog box lets you add, edit or delete data sources at the Windows level. **Tip:** After you create or modify a data source on the Windows level, click the **Reload DSN List** button to refresh the DSN list.

New Local Table Data Source Dialog Box

This dialog box enables to create local data tables in the Data pane for use with your test.



To access	1. Do one of the following: ▪ Ensure that a test or component is in focus in the document pane. ▪ In the solution explorer, select a test or component. 2. Create or open a test or component 3. Select **View > Data**. 4. Click on the **New Data Source** down arrow and select **Local Table**.
Relevant tasks	"Add a local data table" on page 94

User interface elements are described below:

UI Element	Description
	Adds a new column to the local data table. New entries are added below the last column.
	Deletes the selected data table column.
	Raises or lowers the selection in the column list.

UI Element	Description
Columns	A list of the columns in the local data table. Click inside a table row to set or edit the following values: • **Name.** The name of the column. • **Data Type.** A data type: `String`, `Integer`, `Double`, or `Date`. • **Description.** A description of the data in the column.

Troubleshooting and Limitations - Data Pane

This section describes troubleshooting and limitations for using the Data pane.

- Excel data with column headers beginning with digits or containing symbols (such as **!**, **@**, or **#**) are not supported.
- XML data sources: `.xml` files that use XSL are not supported.

Chapter 6: Debug Panes

This chapter includes:

Concepts

Debug Panes Overview

After creating an user code file, you can check to see if it runs properly. Using the debug panes, you can then debug your user code files using Service Test's debugging capabilities.

The debug panes are the primary interface for viewing information about your application during run sessions.

For a conceptual overview of Service Test's debugging capabilities, see "Debugging Overview" on page 308.

For a task related to debugging, see "How to Debug Your User Code File" on page 311.

For an exercise to practice using Service Test's debugging capabilities, see "How to Debug a User Code File - Exercise" on page 314.

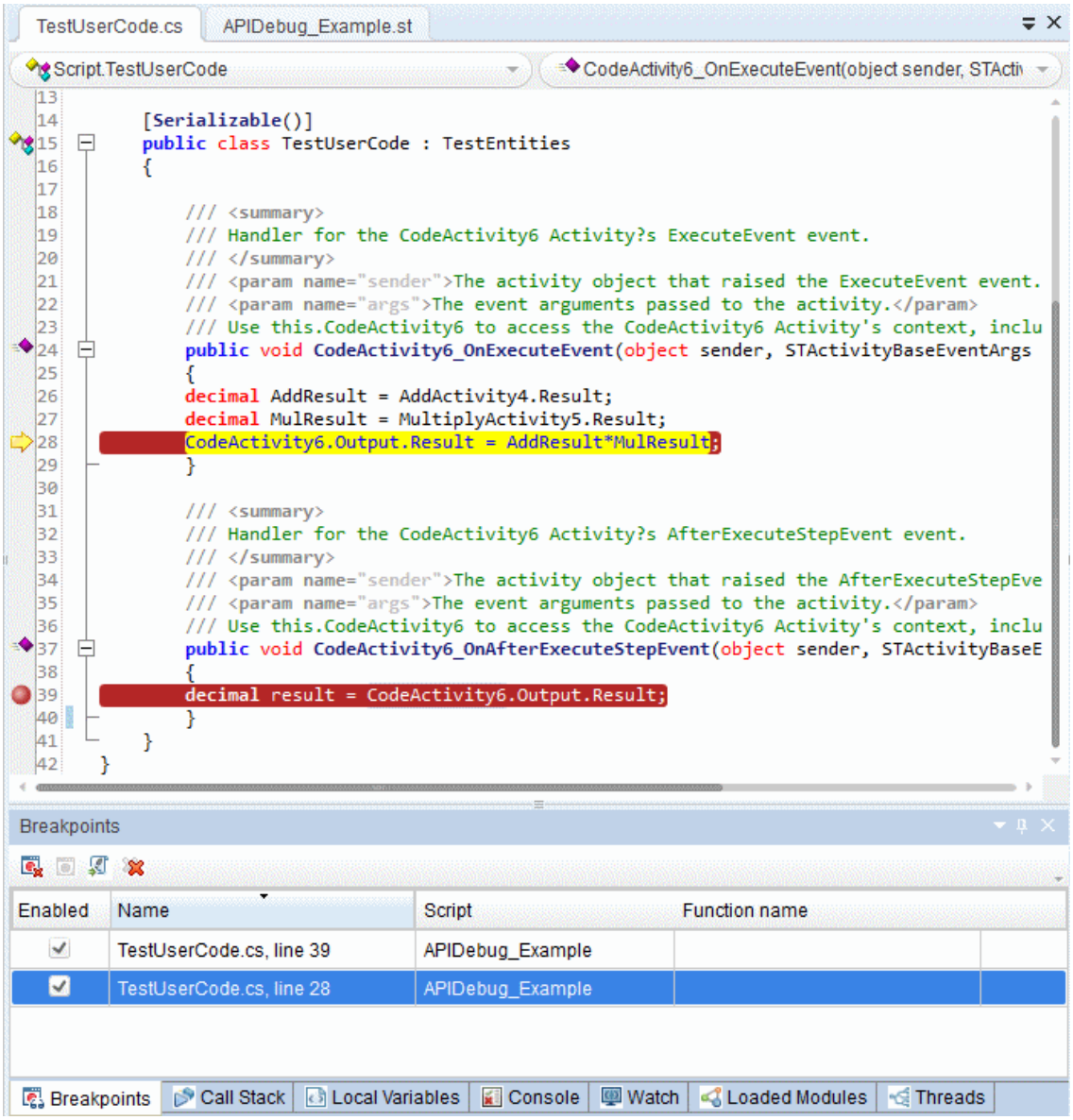
Reference

Debug Pane Interface

The Debug panes enable you to perform different debugging activities when a run session is suspended, such as:

- View, set, or modify the current value of objects or variables in your user code file
- Run C# commands in your paused run session
- Viewing information about the current call stacks of your test
- View information about the associated **.dll** files and threads being used in your run session

The image below shows all the Debug panes open as tabs. The default debug pane shown is the Debug Breakpoints tab.



To access	1. Ensure that a user code file is in focus in the document pane. 2. Select **View > Debug** and choose the relevant debug pane.
Important information	Debug panes display information only when a run session is paused. A run session can be suspended in the following situations: • The run session stops at a breakpoint. • You use **Run** menu commands or context-menu options (such as **Pause**) to suspend a run session.
Relevant tasks	"How to Debug Your User Code File" on page 311

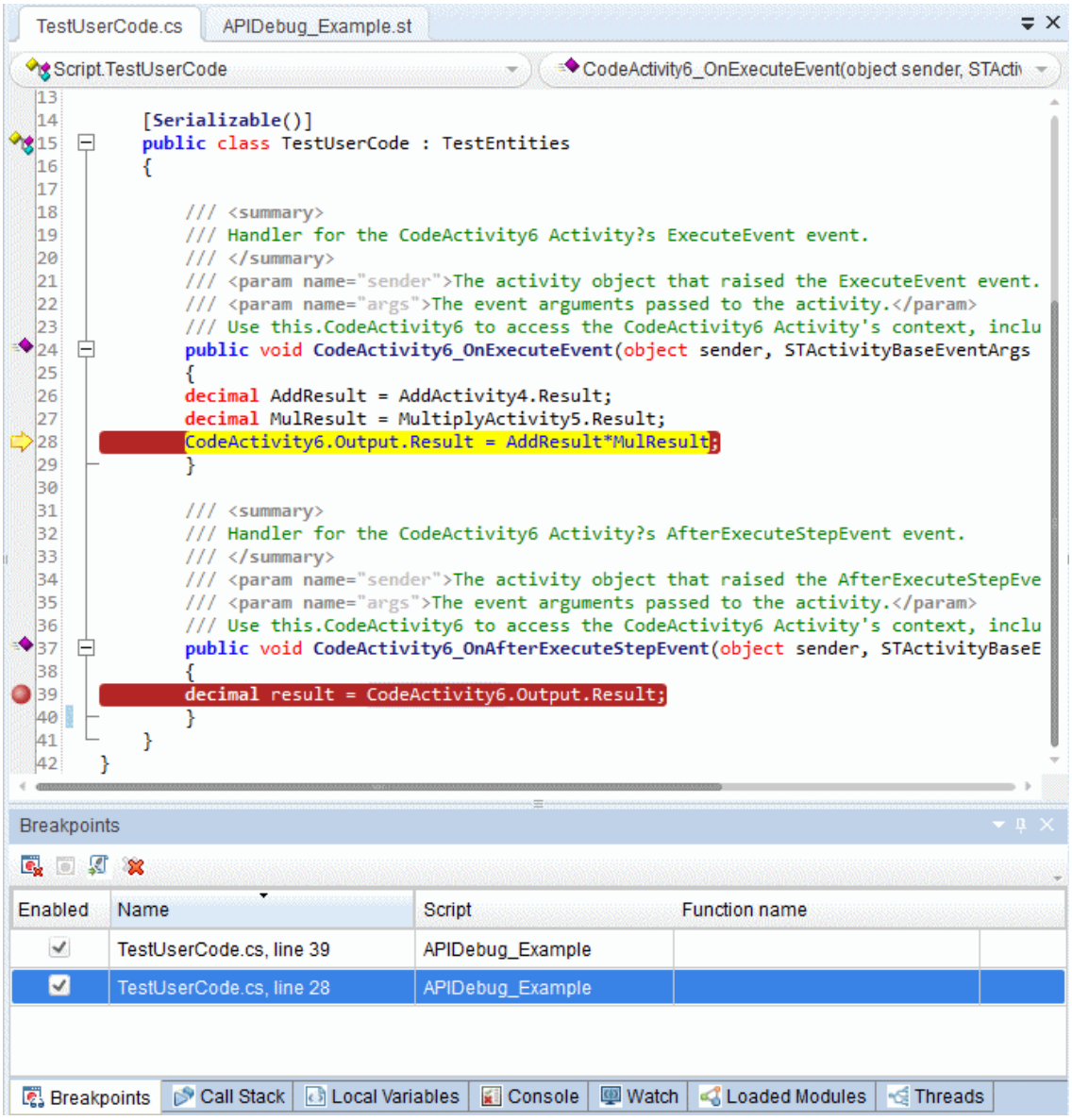
See also	• "Debugging Overview" on page 308 • "Considerations for Debugging User Code Files" on page 308 • "How to Debug a User Code File - Exercise" on page 314

The following debug panes are available:

UI Element	Description
Breakpoints pane	Displays the information about all breakpoints inserted into your user code file and enables you to enable or disable any or all breakpoints in a run session. For details, see " Breakpoints Pane" below.
Call Stack pane	Displays information about the functions, methods, or context currently relevant to your tests. For details, see "Call Stack Pane" on page 116.
Loaded Modules pane	Displays information about the **.dll** files associated with your run session and provides the path to find and debug them. For details, see "Loaded Modules Pane " on page 119.
Threads pane	Displays information about the threads running in your current run session. For details, see "Threads Pane " on page 120.
Local Variables pane	Displays the current values and types of all variables in the current context of the test. For details, see "Local Variables Pane" on page 122.
Console pane	Enables you to run C# commands in your paused run session. For details, see "Console Pane" on page 124.
Watch pane	Displays the current values and types of variables and C# expressions that you add to the Watch pane. For details, see "Watch Pane" on page 126.

Breakpoints Pane

This debug pane enables you to view information about breakpoints inserted into your user code files and navigate directly to the breakpoint location in the relevant test.



To access	1. Do one of the following: ▪ Ensure that a user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, or user code node. 2. Select **View > Debug > Breakpoints** **Note:** This pane displays information only when a breakpoint is inserted into a user code file.

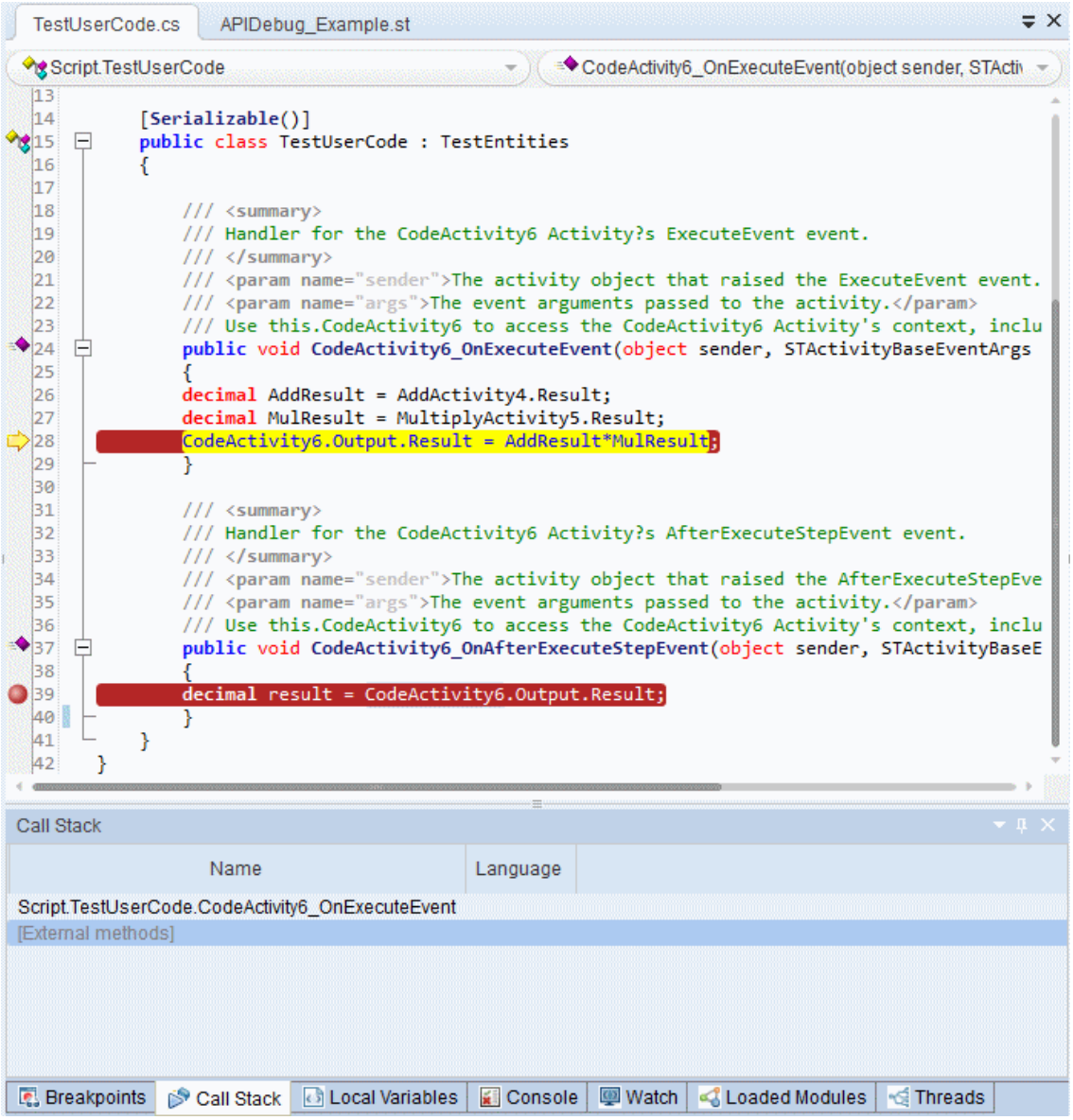
Important information	• The pane displays the breakpoints for the tests that are part of the current solution. • You can double-click on a breakpoint to navigate directly to the relevant line. • Breakpoints are saved with your test or user code file and are maintained even after you close Service Test.
Relevant tasks	"How to Use Breakpoints " on page 312
See also	"Breakpoints " on page 309

User interface elements are described below:

UI Element	Description
	**Remove Current Breakpoint.** Removes the current breakpoint from your test.
	Disable/Enable Current Breakpoint. Disables or enables the selected breakpoint, which instructs Service Test to use or ignore it during a run session.
	**Go to Source.** Navigates directly to the line containing the selected breakpoint in your test.
	**Remove All.** Removes all breakpoints listed in the pane from the tests that contain them.
Enabled	A check box that specifies whether the breakpoint is enabled or disabled.
Name	The file name of the document that contains the breakpoint, and the number of the line in the file that contains the breakpoint.
Script	The name of the user code file that contains the breakpoint.

Call Stack Pane

This debug pane enables you to view information about the methods and functions that are currently on the call stack of your test, component, or user code file or the context in which the run session was paused.



To access	1. Do one of the following: ▪ Ensure that a test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, or user code node. 2. Select **View > Debug > Call Stack**. **Note:** This pane displays information only when a run session is paused.

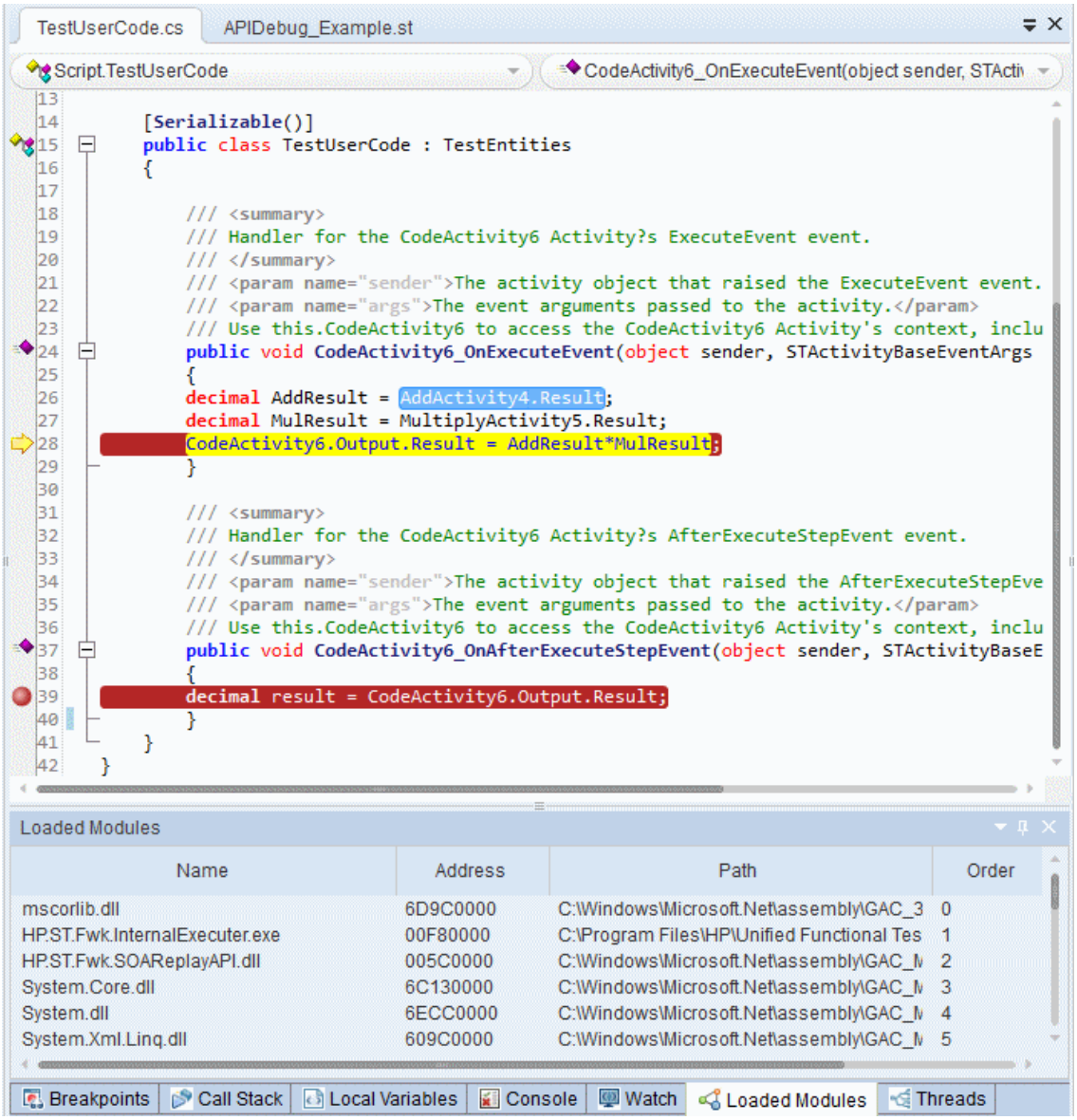
Important information	• This pane is read-only. • You can navigate directly to the beginning line of a function, method, or context in your test by clicking on its row in the pane. If the relevant test is not open, Service Test opens it. • The Watch pane and Local Variables pane display information relevant to the context that you select in the Call Stack pane. For example, if you select the row for a specific function in the Call Stack pane, the variables from this function and expressions selected to watch from this function are displayed in their respective panes. • **Caution:** If you navigate to a stack that is not defined in a user code file, Service Test opens other coding files. Do not edit this code. Doing so can cause unexpected behavior in your run sessions.

User interface elements are described below:

UI Element	Description
Function name	The name of the function, method, or context currently relevant. **Note:** If a run session is suspended before running a step or event or there is no specific function or event being run, then this pane displays the string `Script.TestUserCode.<event name>`. The expressions displayed in the Local Variables and Watch panes are evaluated within the context of the suspended step or event.
File name	The name of the file containing the called function, method, or context. This is the location in the file system or an ALM project.
Line #	The line number on which the function, method, or context definition begins.
Context menu options	The following context menu options are available: • **Show external methods.** Displays all methods running in the background during a run session. **Note:** The Call Stack pane displays the external methods for informational purposes. You cannot navigate to the stack displayed in these method calls. • **Show module names.** Displays the `.dll` file in which the method call is found. • **Show argument values.** • **Show line number.** Displays the line number on which the method call begins for accessible internal methods.

Loaded Modules Pane

This debug pane enables you to view information about the **.dll** files loaded and executed as a test runs.



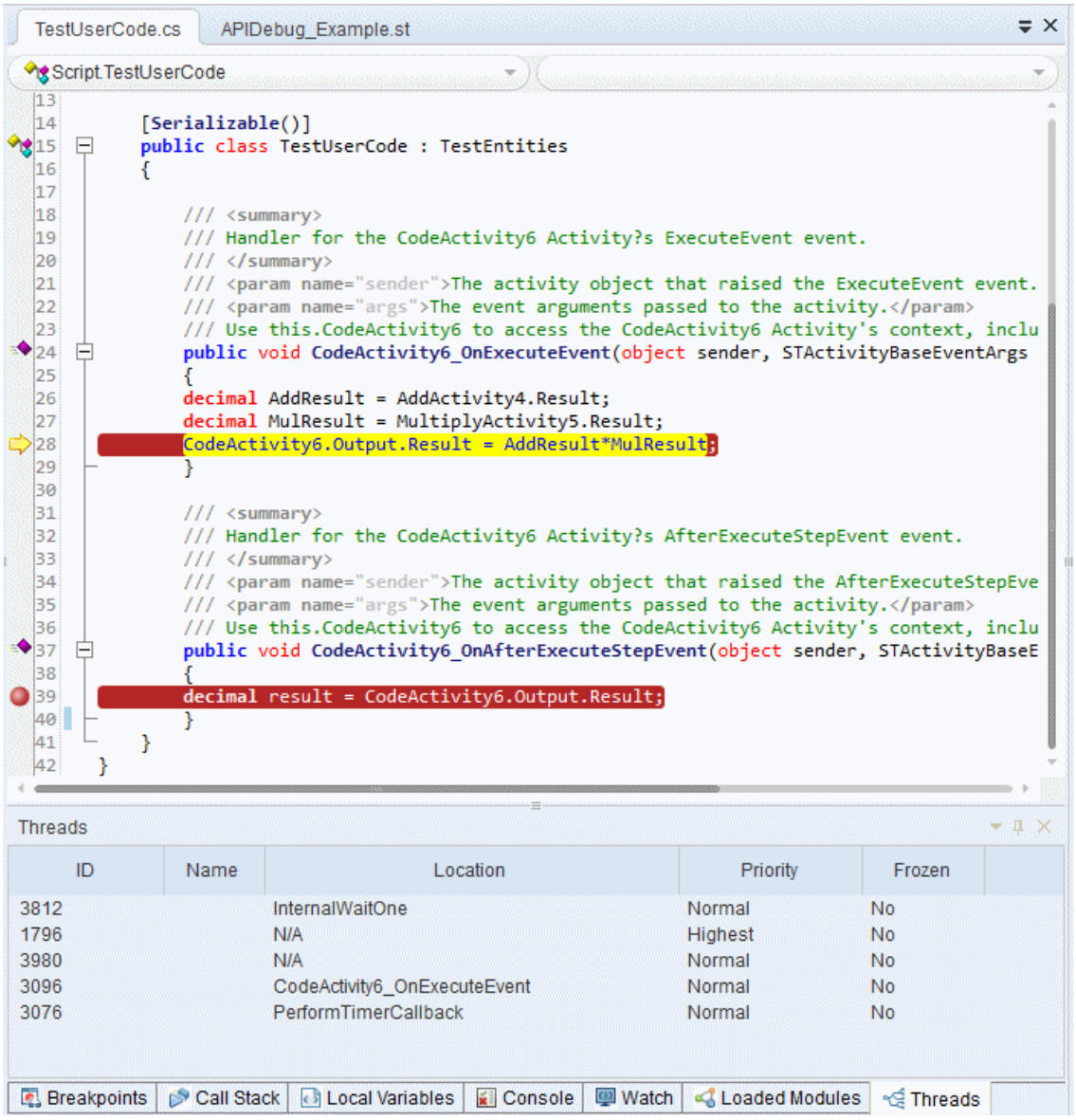
To access	1. Do one of the following: ▪ Ensure that a test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, or user code node. 2. Select **View > Debug > Loaded Modules** **Note:** This pane displays information only when a run session is paused.
Important information	The list of `.dll` files is displayed only when you pause a run session.

User interface elements are described below:

UI Element	Description
Name	The name of the `.dll` file.
Address	The numeric address of the `.dll` file.
Path	The location of the `.dll` file in the file system.
Order	The order in which the `.dll` files are used in your run session.
Symbols	The symbols included with each `.dll` file, which allow for debugging of loaded modules. **Note:** • A `.dll` file's symbols must be loaded to enable you to debug a `.dll` file. • Many of the `.dll` files listed in this pane have no symbols loaded, as they are files called by Service Test during a run session. • Any `.dll` files you add to your test can be debugged if there is a run error.

Threads Pane

This pane enables you to view information about the threads currently running as part of the run session.



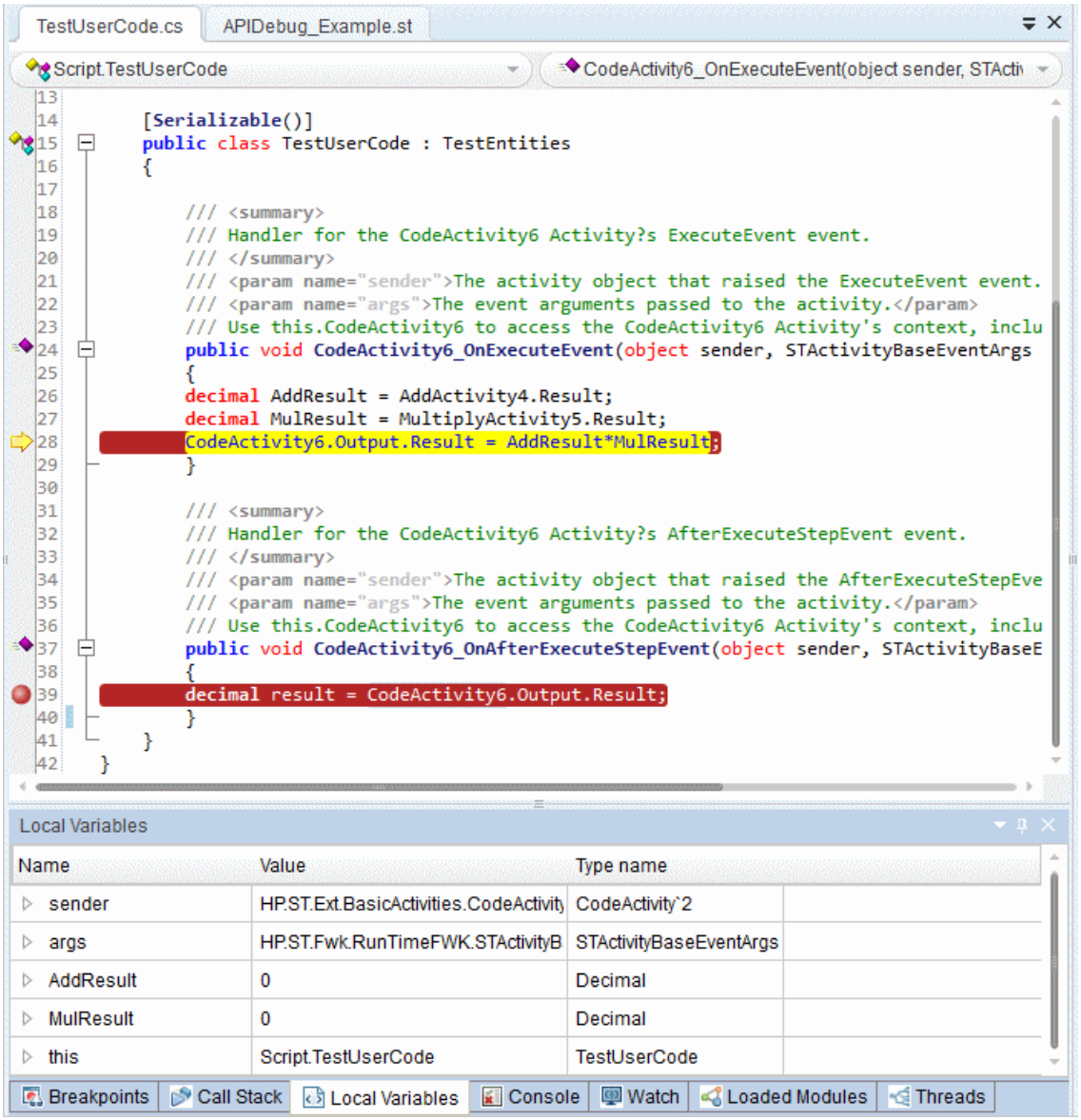
To access	1. Do one of the following: ▪ Ensure that a test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, or user code node. 2. Select **View > Debug > Threads** **Note:** This pane displays information only when a run session is paused.
Important information	• The list of threads is displayed only when you pause a run session. • If your test contains only one thread, then this pane is not available. • If the thread is contained within a user code file, you can double-click on the thread to navigate directly to the beginning of the thread.

User interface elements are described below:

UI Element	Description
ID	The numeric identification of the thread.
Name	The name given to a particular thread.
Location	The location in which the thread is found. **Note:** Some threads are not found in your user code, but represent background threads that are part of your test. These threads are not accessible.
Priority	The order in which the threads can be used when running a multi-thread operation. Higher priority threads are called first.
Frozen	Indicates whether a thread is frozen or unfrozen. Right-click a thread to freeze or unfreeze it.

Local Variables Pane

This debug pane displays the current values and types of all variables in current the context of your test.



To access	1. Do one of the following: ▪ Ensure that a test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, or user code node. 2. Select **View > Debug > Local Variables** **Note:** This pane displays information only when a run session is paused.

Important information	• The information displayed in this pane is read-only. • You cannot edit the value of expressions in the pane. To change the value of an expression, enter a command in the "Console Pane". • Only variables that were recognized up to the last event that occurred are displayed in the Local Variables pane. As you continue stepping through the subsequent steps in your test, Service Test adds any additional variables that it recognizes and updates the values displayed in the Local Variables pane.
Relevant tasks	"How to Debug Your User Code File" on page 311
See also	• "Debugging Overview" on page 308 • "Considerations for Debugging User Code Files" on page 308 • "Watching the Values of Variables and Properties of Objects During a Run Session" on page 309 • "How to Debug a User Code File - Exercise" on page 314

User interface elements are described below:

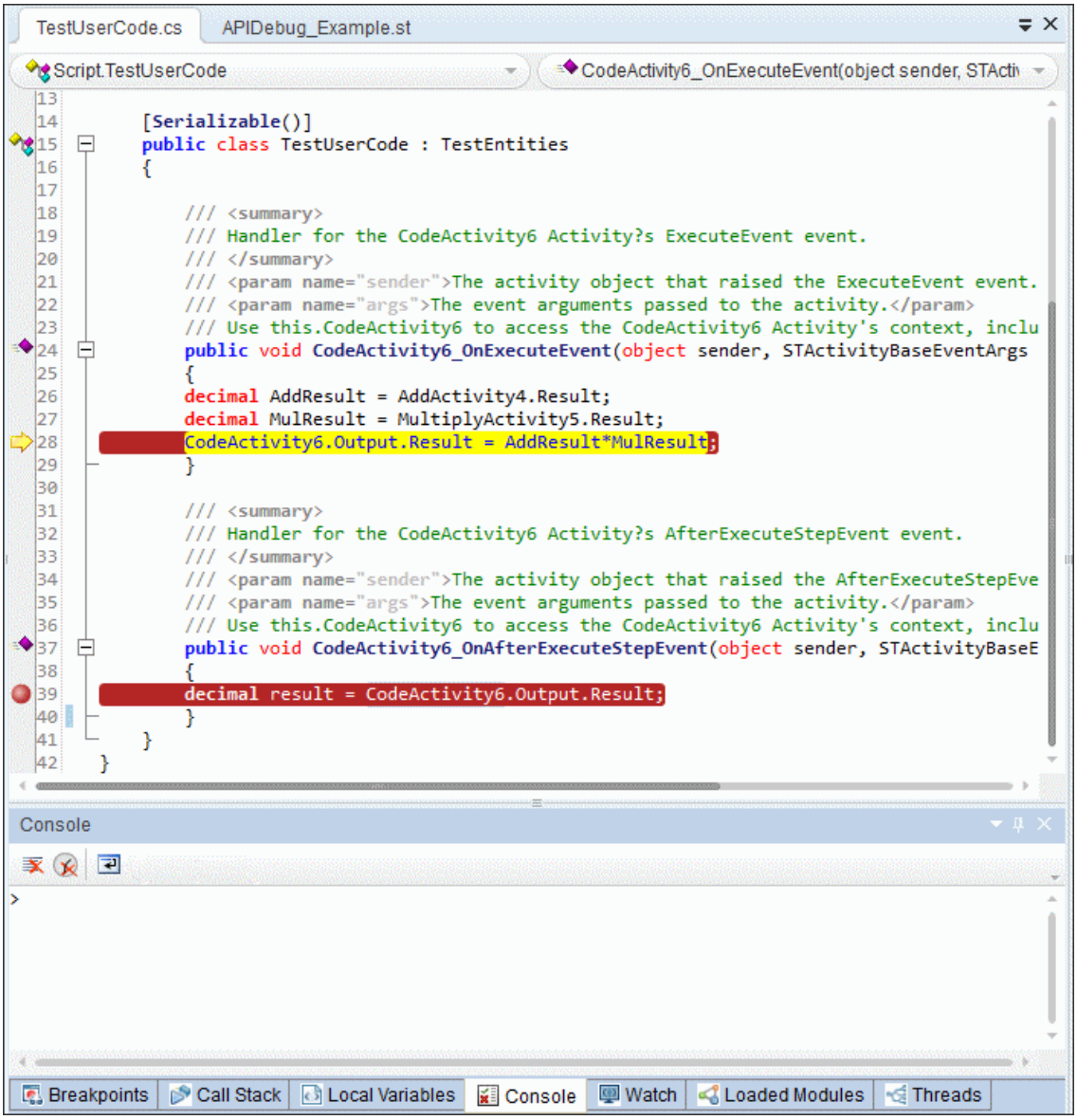
UI Element	Description
Name	The name of the variable, or an expandable node whose child elements contain details about the variable. For example, if a variable is assigned to an object, you can expand the variable in the pane and view its methods and properties.
Value	The current value of the variable. **Note:** If a variable node contains child elements, the value column indicates the number of child elements.
Type Name	The type of the variable's value (for example, **Integer** or **String**).

Console Pane

This debug pane enables you to run lines of C# code in your suspended run session.

For example, you can run code that performs any of the following activities before you resume the run session:

- Modifies the input and output properties for a step
- Sets or modifies a variable



To access	1. Do one of the following: ▪ Ensure that a test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component,, or user code node. 2. Select **View > Debug > Console** **Note:** This pane displays information only when a run session is paused.
Important information	• You can enter lines of code in the Console pane only when a run session is suspended. When no run session is suspended, you can view the command history, or use the **Clear All** command.

Relevant tasks	"How to Debug Your User Code File" on page 311
See also	• "Debugging Overview" on page 308 • "Considerations for Debugging User Code Files" on page 308 • "Watching the Values of Variables and Properties of Objects During a Run Session" on page 309 • "How to Debug a User Code File - Exercise" on page 314

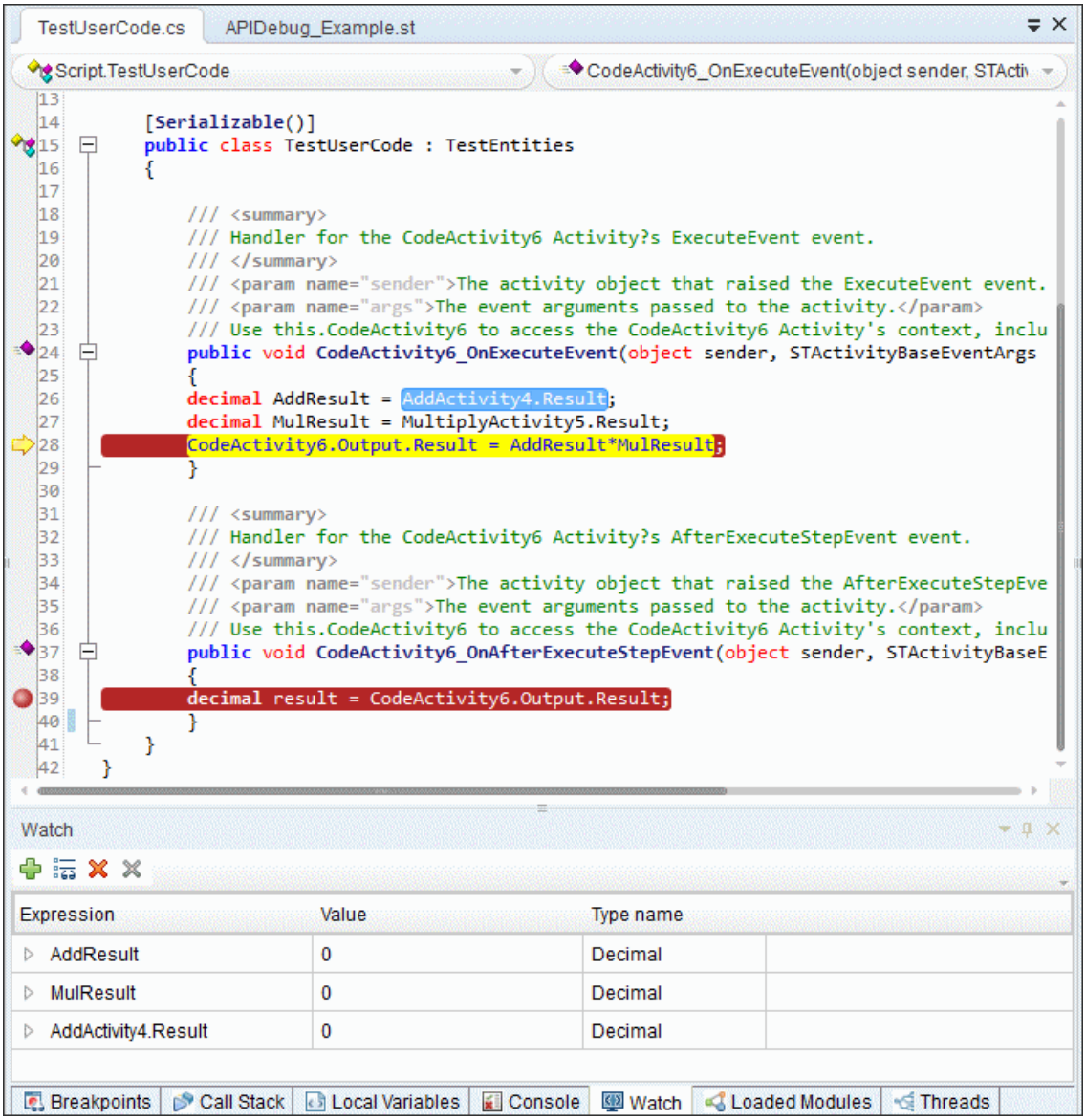
UI elements

Console pane user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
	**Clear Console.** Enables you to erase all of the commands entered in the command line area.
	**Delete History.** Clears all previously saved commands. By default, all previous commands are saved in the pane and can be selected for a new command by using the up or down arrows.
	**Toggle Word Wrap.** Enables or disables word wrap behavior for commands given in the command line area.
<command line area>	Enables you to enter a line of code to be run in the context of your suspended run session. Type or paste the line of code in the window and press ENTER to run the code. This area also displays the command line history, including the lines of code that you ran. **Notes:** • You cannot make any changes to these lines, but you can select and copy text from them. • You can use the UP and DOWN arrow keys to browse through the command history. Service Test copies the commands to the active command line, enabling you to repeat or reuse commands that you entered earlier.

Watch Pane

This debug pane enables you to view the current values and types of selected variables, properties, and C# expressions in your suspended run session.



To access	1. Do one of the following: ▪ Ensure that a test, component, or user code file is in focus in the document pane. ▪ In the solution explorer, select a test, component, or user code node. 2. Select **View > Debug > Watch**. **Note:** This pane is only relevant when a run session is paused.

Important information	• You can add, edit, or remove expressions within the Watch pane using the pane's toolbar buttons in addition to the **Run** menu command (**Run > Add to Watch**). • You cannot edit the value of expressions in the pane. To change the value of a watched expression, enter a command in the "Console Pane". • You can expand Watch pane items if the property selected contains multiple values. • You can sort the columns in the watch pane by expression, value, or type name by clicking the column headers. • Watch expressions are saved with the test and maintained between testing sessions and as you switch between open user code files. • If you added an item to this pane from an user code file that already ran, the pane will show it as undefined.
Relevant tasks	"How to Debug Your User Code File" on page 311
See also	• "Debugging Overview" on page 308 • "Considerations for Debugging User Code Files" on page 308 • "Watching the Values of Variables and Properties of Objects During a Run Session" on page 309 • "How to Debug a User Code File - Exercise" on page 314

User interface elements are described below:

UI Element	Description
	**Add New Watch Expression.** Opens the Add New Watch dialog box, which enables you to add a new expression to the Watch pane. After you enter the expression and click **OK**, Service Test displays the value and type of the expression in the pane.
	**Edit Watch Expression.** Enables you to edit the selected expression during your debug session.
	**Remove Watch Expression.** Removes the currently selected expression from the Watch pane.
	**Remove All.** Removes all the expressions currently displayed in the Watch pane.

UI Element	Description
Expression	The expression whose value you want to watch. For details on adding and removing expressions from the Watch pane, see "Watching the Values of Variables and Properties of Objects During a Run Session" on page 309. **Caution:** Service Testruns the expressions in the Watch pane to evaluate them. Therefore, do not enter any expression whose evaluation could affect the state of the test, as this can lead to unexpected behavior of your test.
Value	The current value of the expression. The evaluated value is displayed only when a run session is suspended.
Type name	The type of the expression's value after it is evaluated (for example, **Integer** or **String**). If an expression cannot be evaluated in the current context, the type displayed is **Incorrect expression**.

Troubleshooting and Limitations - Debug Panes

This section describes troubleshooting and limitations for the Debug Pane.

- You cannot edit variables or properties from the Local Variables or Watch Panes.

  Workaround: Use the Console pane to edit and modify your variable and property values by running a command line to assign the new value.

Chapter 7: Document Pane

This chapter includes:

Concepts

Document Pane Overview

The document pane is the main design area in Service Test and displays all open documents as separate tabs. Each tab can also be individually moved, docked, or floated as an independent pane, and you can drag and drop or pin any of the Service Test panes within the document pane. You can also restore the tabs or panes to the default location from the **View** menu.

The document pane displays the following types of documents:

- **Service Test documents:**
  - Tests
  - Actions
  - C# files containing event handler code (`TestUserCode.cs` files)
  - C# user code files
- **Business Process Testing documents:**
  - Business process tests
  - Business process flows
  - API components
- **Non-testing documents:**
  - Text files

The features available for viewing and editing documents depend on the document type, as described in the following table.

Document Type	Opens in the:
• Tests • Actions • Business components	**Canvas**. Graphically displays and enables you to edit the flow of your test, action, or component. For details, see: "The Canvas" on page 373
• C# code files • Text files	**Editor.** Provides text and code editing features that facilitate the design of your text, script, and code documents. For details: • "Editing Text and Code Documents" on next page • "How to Open a Window for Writing Custom Code" on page 644

Multiple Tests in the Document Pane

Using the document pane, you can open and work with multiple tests simultaneously. To open multiple tests, they must all be included in the same solution.

You can navigate between multiple tests by clicking the tab name of the test you want to view. You can also use the document view drop-down arrow to display a list of open tests.

For details on managing tests, see "How to Create and Manage Tests" on page 56.

Editing Text and Code Documents

You can write customized code for your tests by modifying user code files in the Editor.

The Editor supports common text and code editing features. For details, see:

- "Statement Completion in User Code Files" below
- "Automatic Code Completion" on page 135
- "Bookmarks Overview" on page 84
- "Searching and Replacing in the Editor " on page 135
- For API testing: "Coding Service Test Events" on page 641

Other notable Editor features include expanding and collapsing code, zooming in and out using the mouse, and code templates defined in the Code Templates pane of the Options dialog box (**Tools > Options > Coding** tab **> Code Templates** node). For details about defining code templates, see "Code Templates Pane (Options Dialog Box > Coding Tab)" on page 277.

To define other preferences related to viewing and editing documents in the Editor, see "Text Editor Tab (Options Dialog Box)" on page 279.

For a user interface description of the Editor, see "Editor User Interface" on page 147.

Statement Completion in User Code Files

Statement completion, which is similar to Microsoft's IntelliSense functionality, enables you to increase programming speed and accuracy by providing dynamic lists of items, in the form of tooltips, drop-down lists, or popup windows, while writing statements in the Editor.

As you type in the Editor,Service Test displays items you might want to add to your statement, as well as the syntax relevant to what you are typing. Service Test provides this type of statement completion information, when available, for:

- Activity-specific properties
- Function and method syntax
- Operations
- Variable definitions and methods

For details about statement completion in the Editor, see:

- "Statement Completion Options" below
- "Statement Completion Considerations" on next page

Statement Completion Options

The following table summarizes some statement completion options available when you enter specific items and keystrokes.

If you enter:	Followed by:	Service Test displays:
An operation name	SPACE or Open parenthesis" ("	The operation syntax, including its mandatory and optional arguments. When you add a step that uses an operation, you must define a value for each mandatory argument associated with the operation.
An argument	Comma (,)	The operation syntax, bolding the next argument for which you need to enter a value. Relevant if you enter a comma after any argument value other than the last one in a step. **Note:** For certain operations, when you type the space or comma before an argument that has a predefined list of values, Service Test displays the list of possible values.
An operation or function name	CTRL+SHIFT+SPACE	The statement completion (argument syntax) tooltip for that item.
CTRL+SPACE		A dynamic list of the relevant: • operations • properties • user-defined functions • constants and local variables relevant to the current programming scope • functions and methods relevant to the current programming scope **Note:** If there is only one relevant item defined, its name is automatically entered in the step, without opening the list. For example, if you typed the beginning of the item name before pressing CTRL+SPACE, and only one item matches the text you typed.

If you enter:	Followed by:	Service Test displays:
The beginning of an item in the statement completion list		The list of items, highlighting the first item (alphabetically) that matches the text you typed. Pressing ENTER or SPACE adds the highlighted word to the step.

Statement Completion Considerations

When working with Service Test's statement completion feature, consider the following:

- To close the statement completion drop-down list without selecting from it, press ESC.
- If you resize the frame in which the statement completion drop-down list is displayed, Service Test subsequently uses the new size when it displays statement completion drop-down lists.
- Service Test might not display statement completion information if the statement is typed incorrectly and contains syntax errors. In many cases, you can view such errors in the "Errors Pane" (described on page 172) when you save your changes.

Automatic Code Completion

Service Test provides automatic code completion features, to facilitate coding in the Editor. This includes templates that you can use to insert code snippets by typing specific keywords.

You can also modify the templates provided, or build your own customized templates as needed, and define the keywords used to invoke the use of each template.

For example, you might repeat a complicated **If...Then** statement many times your document. You can use an existing template for the **If...Then** statement to create your own customized template with your more complicated code. You can also create templates from scratch, such as a comment block template, which might include information such as programmer identification, date added, or other details you want included in all comments.

Code templates are defined in the "Code Templates Pane (Options Dialog Box > Coding Tab)" (described on page 277), and are supported for `.cs` files and `.txt` files.

For details, see "How to Use Code Snippets and Templates" on page 139.

Searching and Replacing in the Editor

You can search in the Editor for text strings, as well as references, derived classes, base classes, or overriding methods, for the current method, function or class.

When searching for text, you can use standard text or regular expressions in your search strings, and you can perform string replacements. You can also search in documents that are closed but accessible by the search functionality, either by searching an entire solution, or specifying a search folder.

For details, see "How to Find or Replace Strings in Files" on page 141 and "How to Search for References or Classes in Tests in the Editor" on page 140.

Note: Search and replace functionality is not available in the canvas.

For details, see:

- " Regular Expressions in the Find and Replace Dialog Boxes" on page 165
- "File and Item Types Included in String Searches" below

File and Item Types Included in String Searches

When performing text searches or replacements you can search throughout an entire test or folder. However, the specific file and item types searched within the test or folder are defined by the search algorithm and cannot be modified by users.

Note: The search is limited to the text in the file at the time that the Find or Replace dialog box was opened. Any changes made after opening the dialog box are not included the search.

Searches for strings in tests

When you search for text strings in tests, you can search in actions, `*.cs` files, or `*.txt` files. The search is performed in each source code module and in the test flow.

When you search in a specific C# source code file, the search is performed throughout all user code in the test, as a single text file.

You can search for the following types of items:

- Activity or event display names or event handles
- Global environment variable display names and values
- Link expressions
- Loaded XML or schema files
- Test setting definitions
- Visible checkpoint or property display names and values
- X-paths

Regular Expressions Overview

A **regular expression** is a string that specifies a complex search phrase. By using special characters, such as a period (**.**), asterisk (*****), caret (**^**), and brackets (**[ ]**), you can define the conditions of a search.

Regular expressions are used to identify objects and text strings with varying values. You can use regular expressions to instruct Service Test to find a value that matches a particular pattern or condition instead of a specific hard-coded value.

Whenever a Service Test feature supports regular expressions, the relevant dialog box includes a **Regular Expression** check box. Selecting this check box instructs Service Test to treat the provided value as a regular expression. Some dialog boxes that contain a **Regular Expression**

check box, also contain a right arrow adjacent to the text box for the value. Clicking this arrow enables you to select regular expression characters from a drop-down list, and to test your regular expression to make sure it suits your needs. For more details, see "Smart Regular Expression List" on page 168.

You can use regular expressions only for values of type **string**.

For details on defining regular expressions, including regular expression syntax, see "Regular Expression Characters and Usage Options" on page 161.

Tasks

How to Use Bookmarks in the Editor

This task describes how to use bookmarks in tests and includes the following steps:

- "Insert bookmarks into a document" below
- "Navigate between bookmarks and to specific bookmarks" below
- "Clear bookmarks" below

Insert bookmarks into a document

1. Create or open a user code file. For details, see "How to Create and Manage Tests" on page 56.
2. Click in the line to which you want to assign a bookmark.
3. Select **Search > Bookmarks > Toggle Bookmarks**, or press the Bookmarks button in the Bookmarks pane. A bookmark icon is added to the left of the selected line and is also displayed in the "Bookmarks Pane" (described on page 83).

Navigate between bookmarks and to specific bookmarks

1. To navigate between bookmarks listed in the "Bookmarks Pane", do one of the following:
   - Select **Search > Bookmarks > Next Bookmark** or **Search > Bookmarks > Previous Bookmark** to navigate forward and backward, respectively.
   - In the "Bookmarks Pane", click the **Next Bookmark** and **Previous Bookmark** buttons.
2. To navigate to a specific bookmark, double-click the bookmark's line in the "Bookmarks Pane".

Clear bookmarks

Do one of the following:

- Click a bookmark icon to the left of the selected line to delete the bookmark.
- In the "Bookmarks Pane", click the **Delete** button to delete the selected bookmark.
- In the "Bookmarks Pane", select **Search > Bookmarks > Clear All Bookmarks** or click the **Delete All** button in the "Bookmarks Pane" to delete all bookmarks.

How to Use the Go To Dialog Box in the Editor

This task describes how to go to a specific line of code, class, or function, in a user code, or to navigate to any file.

1. Create or open auser code file. For details, see "How to Create and Manage Tests" on page 56.
2. In the Editor, select **Search > Go To > Location**. The "Go To Dialog Box" (described on page 158) opens.

Tip: By default, line numbers are displayed in the Editor. If they are not displayed, you can select the **Show line numbers** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node). For details on the Text Editor options, see "General Pane (Options Dialog Box > Text Editor Tab)" on page 279.

How to Use Code Snippets and Templates

This task describes how to insert pre-designed code snippets or blocks of text into your document, as well as how to manage templates for such snippets in the "Code Templates Pane (Options Dialog Box > Coding Tab)" (described on page 277). For details on code snippets and templates, see "Automatic Code Completion" on page 135.

This task includes the following steps:

- "Insert code snippets into your test in the Editor" below
- "Modify an existing list of code templates" below
- "Remove an existing list of code templates" on next page
- "Add a new list of code templates" on next page

Insert code snippets into your test in the Editor

1. Create or open a user code file. For details, see "How to Create and Manage Tests" on page 56.
2. Place your cursor at the point in the file that you want to insert the code snippet, type the keyword for a code template defined in the Code Templates pane of the Options dialog box, and press TAB.

Modify an existing list of code templates

1. Open the "Code Templates Pane (Options Dialog Box > Coding Tab)" (described on 277).
2. From the **File Types** drop-down list, select the item associated with the list of templates you want to modify. The table lists all code templates defined for the selected file type or types.
3. To edit the file types associated with the selected list, click **Edit List**. In the Edit List dialog box, enter the file type or types that should be supported by the selected list, separated by semi-colons (;).
4. To edit the list of code templates, select the table row for the code template you want to modify and do one of the following:
   - **Add a new template in the list:** Click the empty space below the last row in the table, above the syntax area.
   - **Edit the template name:** Double-click the cell in the Template column, and update the name.
   - **Edit the keyword:** Double-click the cell in the Keyword column, and update the keyword. The keyword is the text that you enter in the Editor to insert the template.
   - **Edit the code template description:** Double-click the cell in the Description column and update the description text.

- **Edit the code syntax inserted into your code:** Click inside the code syntax area below the table and update the code.

  Note: Note that changes made here do not affect the code snippets available from the **Edit > Code Snippet** menu, which are static and cannot be modified.

Remove an existing list of code templates

1. Open the "Code Templates Pane (Options Dialog Box > Coding Tab)", described on page 277.
2. From the **File Types** drop-down list, select the item associated with the list of templates you want to remove. The table lists all code templates defined for the selected file type or types.
3. Click **Remove List**. All code templates associated with the selected file types are removed, as well as the **File Types** item.

   Caution: This action is irreversible.

Add a new list of code templates

1. Open the "Code Templates Pane (Options Dialog Box > Coding Tab)", described on page 277.
2. Click **Add List**.
3. In the Add List dialog box, enter the file types you want to associate with your new list of templates, separated by semi-colons (;). A blank list is added to the table.
4. In each of the cells in the rows, enter text to add template names, keywords to be entered in the code, and descriptions of each template. In the syntax area below the table, enter the code template to be inserted in your code.
5. To add a new row in the list, click the empty space below the last row in the table, above the syntax area.

How to Search for References or Classes in Tests in the Editor

This task describes how to search for references to functions or method definitions, or base or descending classes, and includes the following steps:

- "Search for references to the currently selected function or method" below
- "Search for classes derived from the currently selected class" on next page
- "Search for methods that override a virtual method" on next page
- "Search for the base class of the current class" on next page

Search for references to the currently selected function or method

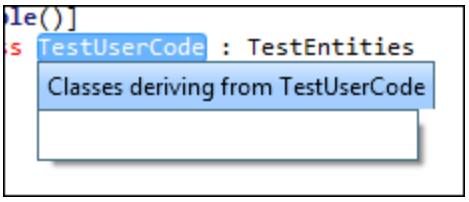
1. Create a new user code file, or open an existing one. For details, see "Specify an event handler - optional " on page 385 or "How to Open a Window for Writing Custom Code" on page 644.
2. Select a function or method definition, and then select **Search > Find References**.

   The search results found are displayed in the "Search Results Pane" (described on page 231).

Search for classes derived from the currently selected class

1. Create a new user code file, or open an existing one. For details, see "Specify an event handler - optional " on page 385 or "How to Open a Window for Writing Custom Code" on page 644.
2. Select a class and then select **Search > Find Derived Symbols**.

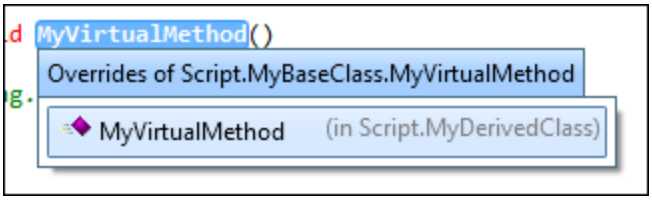
   The derived classes are displayed in a small drop-down box under the selected class. For example:



Search for methods that override a virtual method

1. Create a new user code file, or open an existing one. For details, see "Specify an event handler - optional " on page 385 or "How to Open a Window for Writing Custom Code" on page 644.
2. Select a class and then select **Search > Find Derived Symbols**.

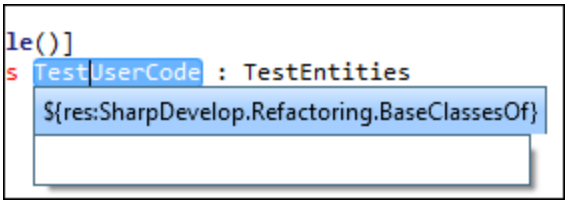
   The overriding methods are displayed in a small drop-down box under the selected class. For example:



Search for the base class of the current class

1. Create a new action or user code file, or open an existing one. For details, see "Specify an event handler - optional " on page 385, "How to Open a Window for Writing Custom Code" on page 644, or "How to Use Actions in a Test" on page 513.
2. Select a class and then select **Search > Find Base Classes**.

   The base classes are displayed in a small drop-down box under the selected class. For example:



How to Find or Replace Strings in Files

This task describes how to search for strings in files that are open in the document pane or supported by the search functionality and includes the following steps:

- "Define the search or replace criteria" below
- "Search for strings in files" below
- "Replace strings in files" on next page

You can find individual occurrences, display all search results, or replace occurrences of the search string with a different string. If a search generates multiple results, all results are displayed in the "Search Results Pane" (described on page 231).

Note: You cannot modify the file and item types included in the search, which are defined by the search algorithm. For details, see "File and Item Types Included in String Searches" on page 136.

Define the search or replace criteria

1. Create or open a user code file. For details, see "How to Create and Manage Tests" on page 56.
2. Do one of the following:
   - To only find strings, open the "Find Dialog Box", described on page 152.
   - To find and replace strings, open the "Replace Dialog Box", described on page 155.
3. In the **Find text** field, enter a search string using plain text or regular expressions. For details, see " Regular Expressions in the Find and Replace Dialog Boxes" on page 165.
4. If you entered a regular expression manually, select the **Regular Expression** check box. (If you selected a regular expression from the **Expression Builder** drop-down list, the **Regular Expressions** check box is automatically selected.)
5. **For replacements only**: In the **Replace with** field, enter the replacement string.
6. From the **Look in** drop-down list, select a location in which to search, or search and replace. Supported locations include the current test, document, or any specified folder, except for folders stored in ALM. If you want to search inside a document stored in ALM, you must open the document first.
7. Select any of the other search options as needed, including matching the text case, matching the whole word, searching upwards in the file instead of downwards, or including subfolders if you selected a folder to search.

Search for strings in files

To search for occurrences of the string one by one:

Do one of the following:

- In the "Find Dialog Box" (described on page 152), click **Find Next.**
- To search again for the most recently defined search criteria without opening the Find dialog box, select **Search > Find Next** or press F3.

The cursor jumps to the next occurrence of the search string and highlights the string. If the search string is found in a closed file, the file automatically opens in the document pane.

To search for all occurrences of the search string:

In the Find dialog box, click **Find All.** The search results are displayed in the "Search Results Pane" (described on page 231).

To perform an incremental search as you type:

1. Do one of the following:
   - To search from the cursor location towards the end of the file, select **Search > Incremental Search** or press CTRL+E.
   - To search from the cursor location towards the top of the file, select **Search > Reverse Incremental Search** or press CTRL+SHIFT+E.

   The cursor changes to a binoculars icon  with an arrow pointing in the search direction.
2. Start typing the string you want to find. Service Test highlights the next matching string.
3. Click anywhere in the file to change the cursor back to the regular cursor.

> **Note:** When performing incremental searches, you can find only one search result at a time. Repeat this step to find the next occurrence of the search string.

Replace strings in files

To replace occurrences of the string one by one:

1. In the "Replace Dialog Box" (described on page 155), click **Find Next** until you reach the search result you want to replace.
2. Click **Replace**. The selected string is replaced with the defined replacement text. The next occurrence is automatically highlighted.

To replace all occurrences of the search string:

In the Replace dialog box, click **Replace All**. A confirmation message lists the number of occurrences replaced.

Reference

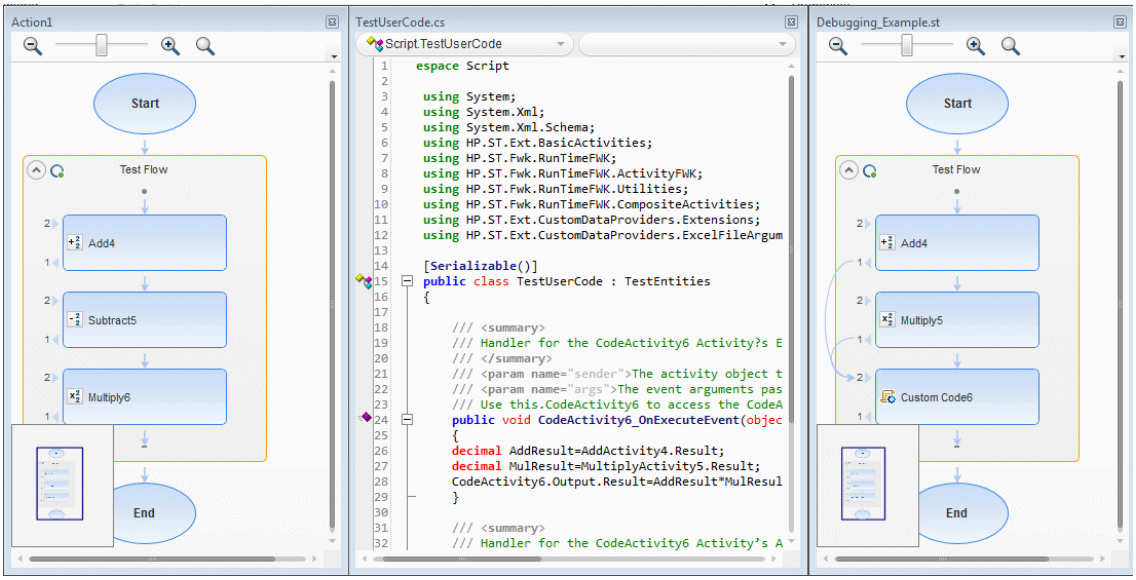
Document Pane User Interface

This pane enables you to view and edit Service Test documents. For details on the types of available documents, see "Document Pane Overview" on page 132.

The image below shows the various document types available in Service Test, including (from left to right):

- An action (canvas)
- An event handler (Editor)
- A test (canvas)

Note: The documents shown in the image below have been undocked from the document pane. By default, these views are displayed as tabs in the document pane.



To access	Do one of the following: • Create or open a test. • Double-click a node in the Solution Explorer. A test opens as a tab in the document pane.

Important information	• If you select a test in the Solution Explorer, but do not double-click it to bring it into focus in the document pane, some other panes (such as the Properties pane), automatically display content relevant for that test. Therefore, the test displayed in the document pane may not correspond to the content of other open panes. • When you bring a test into focus, the corresponding node in the Solution Explorer is highlighted, and other panes are updated accordingly.
Relevant tasks	"How to Create and Manage Tests" on page 56
See also	• "Document Pane Overview" on page 132 • "Document Pane Context Menu Options" below • "Editor User Interface" on page 147

User interface elements for the document when documents are displayed as tabs are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<tab name>	The name of the test. If you hover over the name of the tab, you can view information about the selected document: • **For actions:** the test in which the action is located • **For user code files:** the path on the file system in which the document is saved
	Displays a drop-down list of all tests currently open in the document pane.
	Closes the active test.

User interface elements for the document when documents are displayed as floating documents are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<window name>	The name of the test.
	Closes the active test.

Document Pane Context Menu Options

This section describes the context menu options available for documents displayed as tabs or document windows in the documents pane.

To access	Right-click the test tab or title bar (for floating documents).
Important information	This information describes the context menu options for general document pane tabs and floating documents. For details on the context menu options for a specific type of document, see the relevant section in this guide.

The context menu options for documents displayed as tabs are described below:

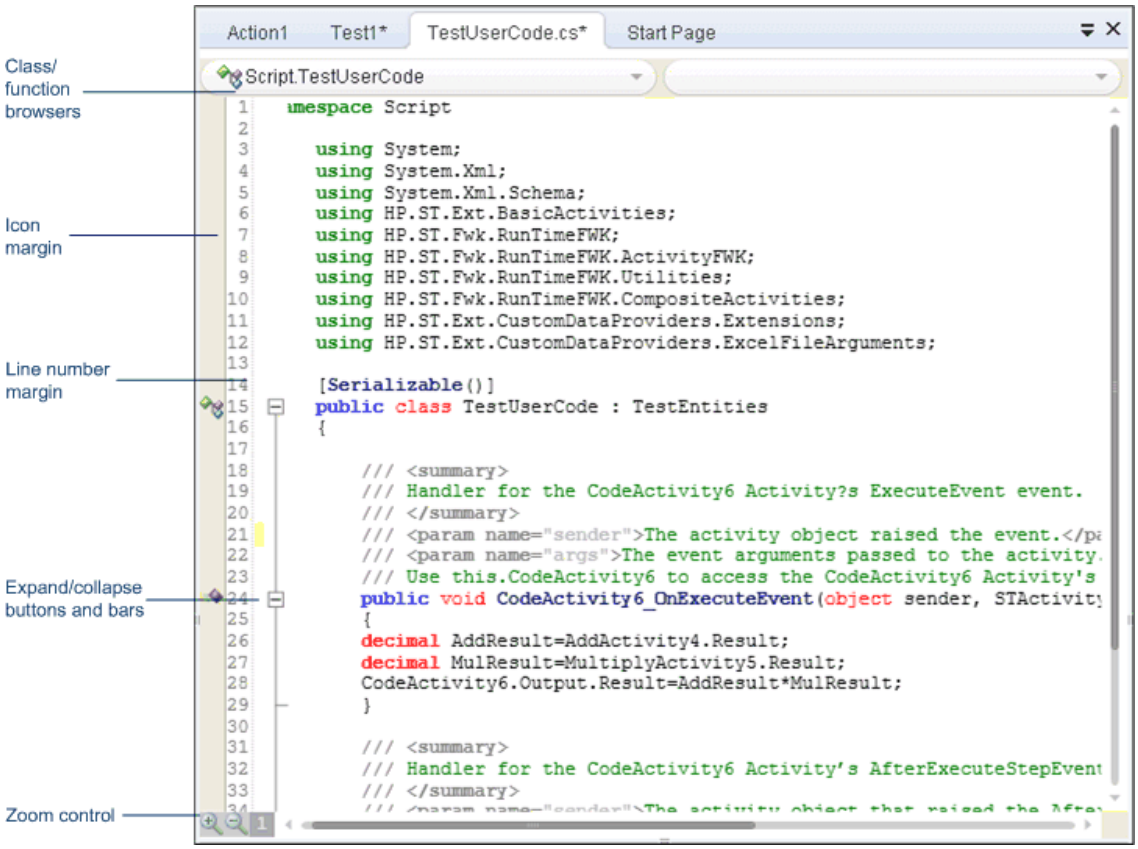
Menu Command	Keyboard Shortcut	Description
Close	CTRL+F4	Closes the current test.
Close All Documents	CTRL+SHIFT+F4	Closes all tests.
Close All but This		Closes all tests except the current test.
Save	CTRL+S	Saves the current test. **Note:** This command is enabled only if you have modified a test.
Save As		Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 72) so you can save a copy of the current document. with another name or in another location. **Note:** This option is enabled only if a test is able to be saved under a different name. For example, you can perform **Save as** for a test but not a test action. This option is only available if the open document is an editable file (such as a user code file.)
Copy file/path name		Copies the full path of the document in the file system or an ALM project to the Clipboard. **Note:** This option is only available if the open document is an editable file (such as a user code file.)
Open Containing Folder in Explorer		Opens the folder containing the document in Windows Explorer. **Note:** This option is only available if the open document is an editable file (such as a user code file.)

The context menu options for documents displayed as floating windows are described below:

UI Element	Description
Close	Closes the test.
Dock as Tabbed Document	Attaches the test as a tab in the document pane.

Editor User Interface

This view enables you to edit text and code for user code files, and text files.



To access	Create a new user code file, or open an existing one.
Important information	To open the help topic for a specific object or method from the Editor, do the following: 1. Verify that you do not have any text highlighted. 2. Place the cursor in the middle of the word and press **F1**. **Note:** You can also use these steps to find VBScript items referenced in the Microsoft VBScript help, included in the Service Test Help. If you place the cursor in an item that is not part of the Service Test test object model and press **F1**, the Service Test Help displays the item as found in the index, or if the exact item is not found, the closest alphabetical item in the index.

Relevant tasks	"How to Create and Manage Tests" on page 56 "How to Create an API Test" on page 383 "How to Use Bookmarks in the Editor" on page 138 "How to Use the Go To Dialog Box in the Editor" on page 138 "How to Use Code Snippets and Templates" on page 139 "How to Search for References or Classes in Tests in the Editor" on page 140 "How to Find or Replace Strings in Files" on page 141
See also	• "Editing Text and Code Documents" on page 133 • "Coding Service Test Events" on page 641 • "Text Editor Tab (Options Dialog Box)" on page 279 • "Find Dialog Box" on page 152 • "Replace Dialog Box" on page 155 • "Go To Dialog Box" on page 158 • "Bookmarks Pane User Interface" on page 85

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

UI Element	Description
<class and function browsers>	Drop-down menus that enable you to quickly browse to a class and function by selecting from those available in the current document. These drop-down menus are most helpful in documents that include many function definitions, such as function libraries. To navigate to a class or function definition, first select the class from the left drop-down menu, and then select the function from the right drop-down menu. **Note:** If the class and function browsers are not displayed, you can select the **Show class/function browser** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node).
<Icon margin>	Display icons for lines with specific types of code, such as classes, events, or fields. For details, see "Editor Icons" on page 150.

UI Element	Description
<Line number margin>	Displays the line numbers for each line of code. You can use the "Go To Dialog Box" (described on page 158) to navigate to a specific line in the code. **Note:** If the line numbers are not displayed, you can select the **Show line numbers** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node.
	**<Expand and collapse>.** Expands or collapses folded code. Folded code is indicated by an ellipsis at the end of the first line of the folded code. **Tip:** Hover over the ellipses to display a snapshot of the hidden code. **Note:** If the expand and collapse buttons are not displayed, you can select the **Enable folding** option in the General pane of the Text Editor pane (**Tools > Options > Text Editor** tab **> General** node.
	**<Zoom control bar>.** Displayed in the lower left corner of the Editor, only after having zoomed in or out of the text for the first time in a session. • Click the **Zoom in** icon to zoom in. • Click the **Zoom out** to zoom out. • Click the **Default zoom** icon to return the text to the default size.

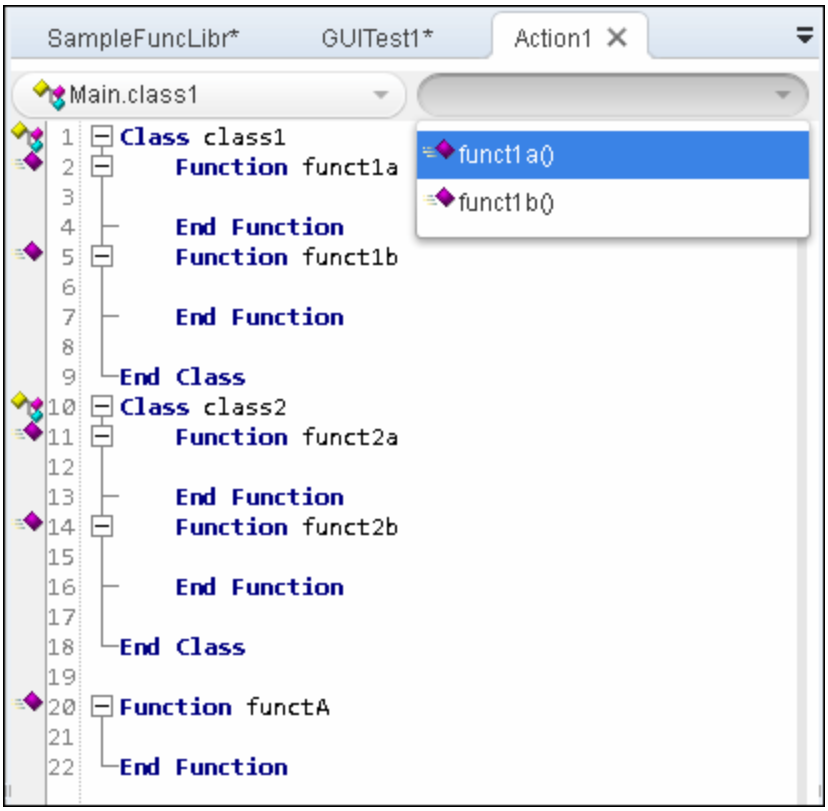
Class and Function Browser Examples

The following image shows an example of an action containing two classes and a function. Each of the classes contains two functions.

The functions defined in a specific class are displayed in the function browser only when you select that class in the class browser. If you select one of these functions, the cursor jumps to that function definition. If you select a different class from the class browser, the cursor jumps to that class definition.

In this image, the **Main.class1** class is selected in the class browser, and the function browser is expanded, displaying the functions defined in the **class1** class (**fucnt_1a** and **fucnt_1b**).

In the class browser, the **Main** item represents the context of the document. In this image, if you select **Main** from the class browser, the function browser displays only **funct_A**.



Editor Icons

Icons used in the Editor to indicate the different types of code are described below:

Type	Icon	Description
Class		Class
		Internal class
		Private class
		Protected class
Delegate		Delegate
		Internal delegate
		Private delegate
		Protected delegate

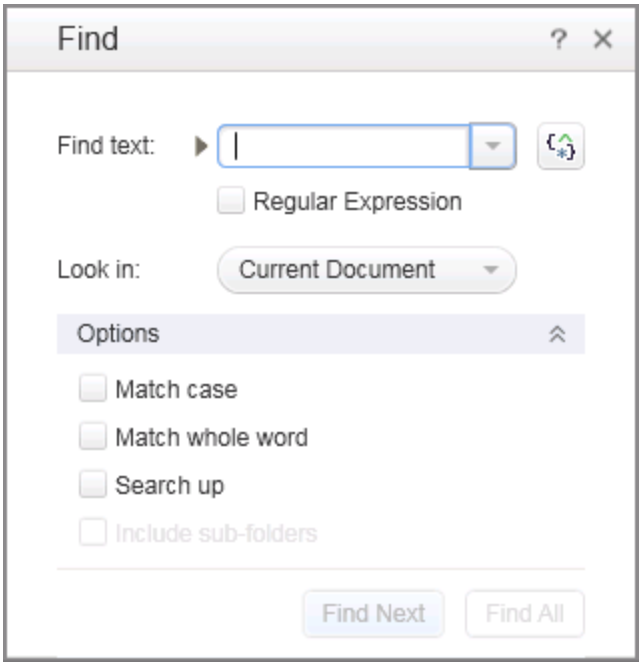
Type	Icon	Description
Enumeration definition		Enumeration definition
		Internal enumeration definition
		Private enumeration definition
		Protected enumeration definition
Event		Event
		Internal event
		Private event
		Protected event
Extension method		Extension method
		Internal extension method
		Private extension method
		Protected extension method
Field		Field
		Internal field
		Private field
		Protected field
Indexer		Indexer
		Internal indexer
		Private indexer
		Protected indexer
Interface		Interface
		Internal interface
		Private interface
		Protected interface
Keyword		Keyword
Literal		Literal

Type	Icon	Description
Local		Local
Method		Method
		Internal method
		Private method
		Protected method
Namespace	{}	Namespace
Operator		Operator
Parameter		Parameter
Property		Property
		Internal property
		Private property
		Protected property
Reference		Reference
Structure		Structure
		Internal structure
		Private structure
		Protected structure

Find Dialog Box

This dialog box enables you to search for strings in files displayed in the Editor or located in a specific folder.

You can either find literal text or use regular expressions. You can also use other options to further fine-tune your search.



To access	1. Create or open the document in which you want to search, or, to search in multiple files, open any document. 2. Select **Search > Find** or press CTRL+F.
Important information	• You cannot use the **Find** dialog box to search in the canvas or application areas. • The search is limited to the text in the file at the time that the **Find** dialog box was opened. Any changes made after opening the **Find** dialog box are not included the search.
Relevant tasks	"How to Find or Replace Strings in Files" on page 141
See also	"Searching and Replacing in the Editor " on page 135

User interface elements are described below:

UI Elements	Description
Find text	The text string you want to locate. For details on searching with a regular expression, see the **Expression Builder** button description.
	**Recent searches**. Click to select a recently used search string. The list includes the last ten search strings.

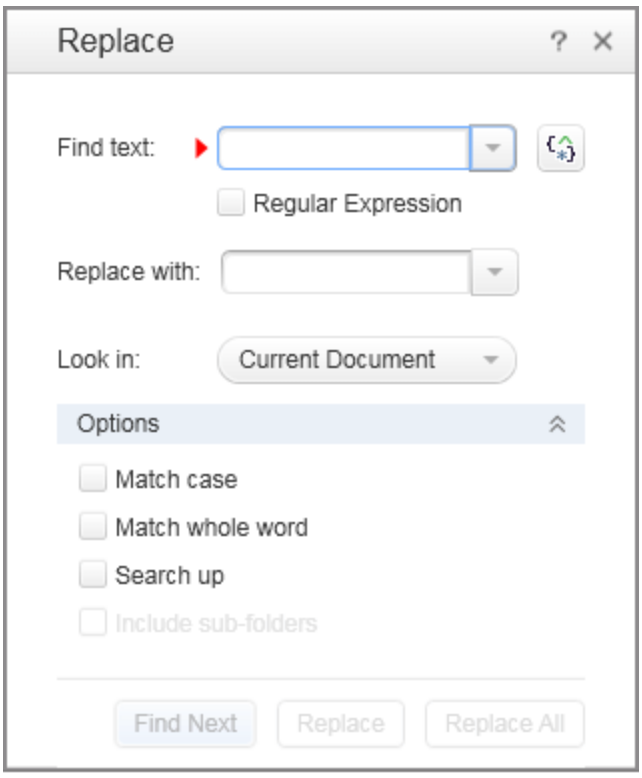
UI Elements	Description
	**Expression Builder**. Click to select a predefined regular expression and insert it into your search string at the cursor location. When you click this button, the **Regular expression** check box is automatically selected. For details on regular expressions in search strings, see " Regular Expressions in the Find and Replace Dialog Boxes" on page 165.
Regular Expression	Treats the specified search string as a regular expression. This option is automatically selected when you select a regular expression from the Smart Regular Expression drop-down list.
Look in	The files or folder you want to search. Select one of the following from the drop-down list: • **Current Document.** Searches within the document currently open in the Editor. This is the default option. • **Current Test.** Searches within the currently open test. • **<Files...>.** Enables you to browse to a select a specific folder in the file system. **Note: (for ALM users)** You cannot browse to an ALM path. The drop-down list also displays the last ten folders you selected, with the most recent folder listed first. Select one to search in the same folder again. For details, see "File and Item Types Included in String Searches" on page 136.
Match case	Distinguishes between upper-case and lower-case characters in the search, and searches only for occurrences with exact matches to the casing in the search string.
Match whole word	Searches only for occurrences that are whole words and not part of longer words.
Search up	Searches from the cursor location towards the top of the file, and then wraps around to the current cursor location from the bottom of the file. By default, the search is performed downwards in the file.
Include sub-folders	Searches through any sub-folders found in the selected folder. **Note:** This option is available only when you select a specific folder from the **Look in** drop-down list.

UI Elements	Description
Find Next	Searches for the next occurrence of the defined search criteria.
Find All	Searches for all occurrences of the defined search criteria. Search results are listed in the Search Results pane. For details, see "Search Results Pane User Interface" on page 231.

Replace Dialog Box

This dialog box enables you to search for strings in files displayed in the Editor or located in a specific folder, and replace them with a different string.

You can either find and replace literal text or use regular expressions. You can also use other options to further fine-tune your find and replace process.



To access	1. Create or open the document in which you want to search, or, to search in multiple files, open any document. 2. Select **Search > Replace** or press **CTRL+H**.

Important information	• You cannot use the **Replace** dialog box to search in the canvas or application areas. • The search is limited to the text in the file at the time that the **Replace** dialog box was opened. Any changes made after opening the **Replace** dialog box are not included the search.
Relevant tasks	"How to Find or Replace Strings in Files" on page 141
See also	"Searching and Replacing in the Editor " on page 135

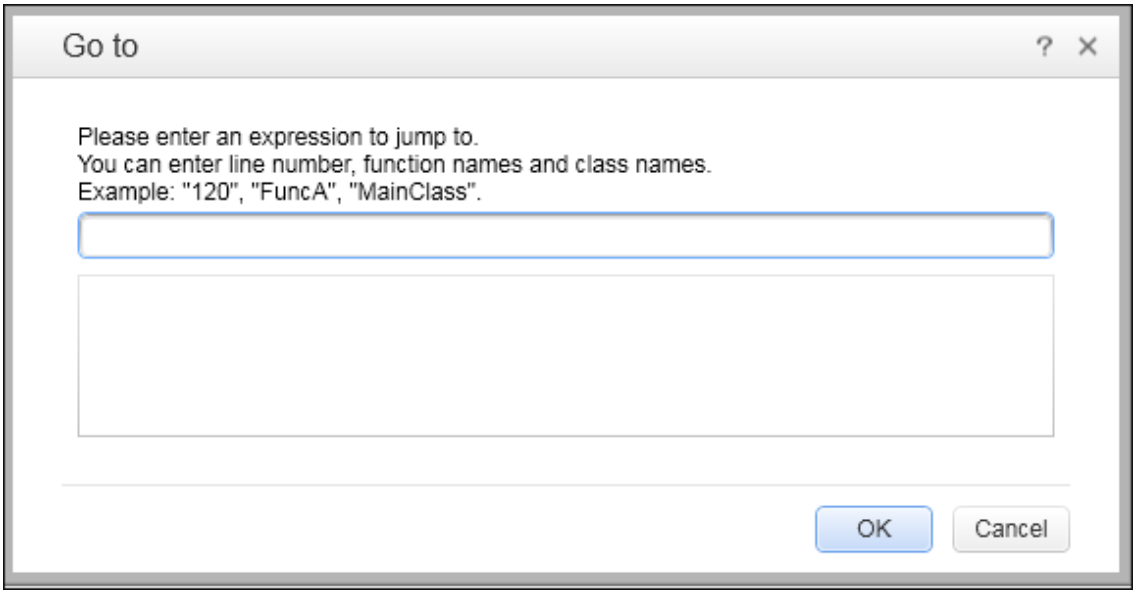
User interface elements are described below:

UI Elements	Description
Find text	The text string you want to locate. For details on searching with a regular expression, see the **Expression Builder** button description.
	**Recent searches and replacements.** Click to select a recently used string. The list includes the last ten search strings.
	**Expression Builder.** Click to select a predefined regular expression and insert it into your search string at the cursor location. When you click this button, the **Regular expression** check box is automatically selected. For details on regular expressions in search strings, see " Regular Expressions in the Find and Replace Dialog Boxes" on page 165.
Regular Expression	Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.
Replace with	The text string you want to use as the replacement string. The text is replaced using the same casing as defined in the replacement string.

UI Elements	Description
Look in	The files or folder you want to search. Select one of the following from the drop-down list: • **Current Selection.** Searches within the currently highlighted text. You must highlight a specific area in the current file before selecting this option. • **Current Document.** Searches within the document currently open in the Editor. This is the default option. • **Current Test.** Searches within the currently open test. • **<Files...>.** Enables you to browse to a select a specific folder in the file system. **Note: (for ALMusers)** You cannot browse to an ALM path. If you want to search in a document stored in ALM, you must open the document first. The drop-down list also displays the last ten folders you selected, with the most recent folder listed first. Select one to search in the same folder again. For details, see "File and Item Types Included in String Searches" on page 136.
Match case	Distinguishes between upper-case and lower-case characters in the search, and searches only for occurrences with exact matches to the casing in the search string.
Match whole word	Searches only for occurrences that are whole words and not part of longer words.
Search up	Searches from the cursor location towards the top of the file, and then wraps around to the current cursor location from the bottom of the file.
Include sub-folders	Searches and replaces through any sub-folders found in the selected folder. **Note:** This option is available only when you select a specific folder from the **Look in** drop-down list.
Find Next	Searches for the next occurrence of the defined search criteria.
Replace	Replaces the string defined in the **Find text** field with the string defined in the **Replace with** field.
Replace All	Replaces all occurrences of the string defined in the **Find text** field with the string defined in the **Replace with field**, throughout the location defined in the **Look in** drop-down list. All strings are found and replaced in the same action.

Go To Dialog Box

This dialog box enables you to navigate to a specific line,class, or function in a user code file, or to navigate to any file.



To access	1. Create a new solution or open an existing solution. 2. Open the necessary tests or user code files. For details on how to open tests or user code files included in a solution, see "How to Manage Items in the Solution Explorer Pane" on page 237. 3. Select **Search > Go To**.
Important information	**Search Limitations:** • When you navigate to an class or function name, Service Test searches all files within the open solution for occurrences of the search terms. • When you navigate to line numbers, Service Test searches only the currently selected document. • You can navigate to any file name contained within the current solution.
Relevant tasks	"How to Use the Go To Dialog Box in the Editor" on page 138
See also	"Bookmarks Overview" on page 84

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

UI Elements	Description
<search text>	The location to which you want to go. Enter one of the following: • A line number • A class name • A function name • Any file name
<search results>	The locations that match the text you entered. As you enter a navigation search string, all locations that match the current string are displayed. Double-click a specific location to navigate directly to it. If a test or user code file containing the location is closed, Service Test opens the relevant documenttest or user code file.

Editor Context Menu Options

This section describes the context menu options available for the different document types displayed in the Editor.

To access	1. Create or open auser code file. 2. Right-click in the document.
Important information	• Some options are also available through menu commands or toolbar buttons. • Some of the context-menu options may be hidden depending on the view of the document or the content of the document.
Relevant tasks	"How to Open a Window for Writing Custom Code" on page 644
See also	"Coding Service Test Events" on page 641

Context menu options are described below:

Note: The menu commands described below may not be displayed in the exact order presented here. The options available differ depending where in your document you right-click.

Menu Command	Keyboard Shortcut	Description
Cut		Removes the currently selected step or text from your action or component.
Copy	CTRL+C	Copies the currently selected step or text.

Menu Command	Keyboard Shortcut	Description
Paste	CTRL+V	Pastes the currently copied step or text.
Delete		Deletes the selected step or text from your action or component.
Comment	CTRL+M	Adds a comment to the selected line.
Uncomment	CTRL+ SHIFT+M	Removes a comment from the selected line.
Indent	TAB	Indents the current line.
Outdent	Shift+Tab	Removes an indent from the selected line.
Go to Definition	Ctrl+ Enter	Navigates directly to the definition of the function that is called in the currently selected step.
Insert/Remove Breakpoint	F9	Sets or clears a breakpoint in the test or component.
Enable/Disable Breakpoint	Ctrl+F9	Enables or disables the currently selected breakpoint.
Add to Watch	Ctrl+T	Adds the selected item to the Watch pane.
Select All	Ctrl+A	Selects all steps or text.

Regular Expression User Interface

By default, Service Test treats all characters in a regular expression literally, except for the period (`.`), hyphen (`-`), asterisk (`*`), caret (`^`), brackets (`[ ]`), parentheses (`( )`), dollar sign (`$`), vertical line (`|`), plus sign (`+`), question mark (`?`), and backslash (`\`). When one of these special characters is preceded by a backslash (`\`), Service Test treats it as a literal character.

If you enter a special character in the value box of any dialog box that contains the **Regular Expression** check box, and then you select the **Regular Expression** check box, Service Test asks you if you want to add a backslash (`\`) before each special character. If you click **Yes**, a backslash (`\`) is added before the special character to instruct Service Test to treat the character literally. If you click **No**, Service Test treats the special character as a regular expression character.

For details, see:

Regular Expression Characters and Usage Options

This section describes some of the more common options that can be used to create regular expressions:

Using the Backslash Character (\ )

A backslash (\) can serve two purposes. It can be used in conjunction with a special character to indicate that the next character be treated as a literal character. For example, `\.` would be treated as period (.) instead of a wildcard. Alternatively, if the backslash (\) is used in conjunction with some characters that would otherwise be treated as literal characters, such as the letters `n`, `t`, `w`, or `d`, the combination indicates a special character. For example, `\n` stands for the newline character.

If the backslash character is not used for either of these purposes, it is ignored.

For example:

- `w` matches the character `w`
- `\w` is a special character that matches any word character including underscore
- `\\` matches the literal character `\`
- `\(` matches the literal character `(`
- `one\two` matches the string `onetwo`

For example, if you were looking for a Web site called:

```
newtours.demoaut.com
```

the period would be mistaken as an indication of a regular expression. To indicate that the period is not part of a regular expression, you would enter it as follows:

```
newtours\.demoaut\.com
```

Matching Any Single Character (.)

A period `(.)` instructs Service Test to search for any single character (except for \n). For example:

```
welcome.
```

matches `welcomes`, `welcomed`, or `welcome` followed by a space or any other single character. A series of periods indicates the same number of unspecified characters.

To match any single character including \n, enter:

```
(.|\n)
```

For more details on the **( )** regular expression characters, see "Regular Expression Characters and Usage Options" above. For more details on the **|** regular expression character, see "Matching One of Several Regular Expressions (|)" on page 163.

Matching Any Single Character in a List ([xy])

Square brackets instruct Service Test to search for any single character within a list of characters. For example, to search for the date 1967, 1968, or 1969, enter:

196[789]

Matching Any Single Character Not in a List ([^xy])

When a caret `(^)` is the first character inside square brackets, it instructs Service Test to match any character in the list except for the ones specified in the string. For example:

[^ab]

matches any character except `a` or `b`

Note: The caret has this special meaning only when it is displayed first within the brackets.

Matching Any Single Character within a Range ([x-y])

To match a single character within a range, you can use square brackets `([ ])` with the hyphen `(-)` character. For instance, to match any year in the 1960s, enter:

```
196[0-9]
```

A hyphen does not signify a range if it is displayed as the first or last character within brackets, or after a caret `(^)`.

For example, `[-a-z]` matches a hyphen or any lowercase letter.

Note: Within brackets, the characters ".", "*", "[" and "\" are literal. For example, `[.*]` matches `.` or `*`. If the right bracket is the first character in the range, it is also literal.

Matching Zero or More Specific Characters (*)

An asterisk (*) instructs Service Test to match zero or more occurrences of the preceding character. For example:

```
ca*r
```

matches `car`, `caaaaaar`, and `cr`

Matching One or More Specific Characters (+)

A plus sign `(+)` instructs Service Test to match one or more occurrences of the preceding character. For example:

```
ca+r
```

matches `car` and `caaaaaar`, but not `cr`

Matching Zero or One Specific Character (?)

A question mark `(?)` instructs Service Test to match zero or one occurrences of the preceding character. For example:

```
ca?r
```

matches `car` and `cr`, but nothing else

Grouping Regular Expressions (( ))

Parentheses `(())` instruct Service Test to treat the contained sequence as a unit, just as in mathematics and programming languages.

Using groups is especially useful for delimiting the arguments to an alternation operator (|) or a repetition operator: (* , + , ? , { })

Matching One of Several Regular Expressions (|)

A vertical line (|) instructs Service Test to match one of a choice of expressions. For example:

```
foo|bar
```

causes Service Test to match either `foo` or `bar`

```
fo(o|b)ar
```

causes Service Test to match either `fooar` or `fobar`

Matching the Beginning of a Line (^)

A caret (^) instructs Service Test to match the expression only at the start of a line, or after a newline character.

For example:

```
book
```

matches `book` within the lines—`book, my book`, and `book list`, while

```
^book
```

matches `book` only in the lines—`book` and `book list`

Matching the End of a Line ($)

A dollar sign ($) instructs Service Test to match the expression only at the end of a line.

For example:

```
book
```

matches `book` within the lines—`my book`, and `book list`, while a string that is followed by (\n), (\r), or ($), matches only lines ending in that string. For example:

```
book$
```

matches `book` only in the line—`my book`

Matching a Newline or Carriage Return Character (\n) or (\r)

\n or \r instruct Service Test to match the expression only when followed by a newline or carriage return character.

- \n instructs Service Test to match any newline characters.
- \r instructs Service Test to match any carriage return characters.

For example:

```
book
```

matches `book` within the lines—`my book`, and `book list`, while a string that is followed by (\n) or (\r) matches only lines that are followed by a newline or carriage return character. For example:

```
book\r
```

matches `book` only when `book` is followed by a carriage return

Matching Any AlphaNumeric Character Including the Underscore (\w)

`\w` instructs Service Test to match any alphanumeric character and the underscore (A-Z, a-z, 0-9, _).

For example:

`\w*` causes Service Test to match zero or more occurrences of the alphanumeric characters—`A-Z`, `a-z`, `0-9`, and the underscore (_). It matches `Ab`, `r9Cj`, or `12_uYLgeu_435.`

For example:

`\w{3}` causes Service Test to match 3 occurrences of the alphanumeric characters `A-Z`, `a-z`, `0-9`, and the underscore (_). It matches `Ab4`, `r9_`, or `z_M`.

Matching Any Non-AlphaNumeric Character (\W)

\W instructs Service Test to match any character other than alphanumeric characters and underscores.

For example:

`\W`

matches `&`, `*`, `^`, `%`, `$`, and `#`

Matching a Decimal Digit (\d)

`\d` instructs Service Test to match any decimal digit.

For example:

`\d`

matches `1`, `2`, `4`, and `5`

Matching an Integer (\D)

`\D` instructs Service Test to match any whole integer.

For example:

`\D`

matches `145643`, `20`, `3426767`, `4`, and `5`

Combining Regular Expression Operators

You can combine regular expression operators in a single expression to achieve the exact search criteria you need.

For example, you can combine the `.' and `*' characters to find zero or more occurrences of any character (except \n).

For example,

`start.*`

matches `start`, `started`, `starting`, `starter`

You can use a combination of brackets and an asterisk to limit the search to a combination of non-numeric characters. For example:

```
[a-zA-Z]*
```

To match any number between 0 and 1200, you need to match numbers with 1 digit, 2 digits, 3 digits, or 4 digits between 1000-1200.

The regular expression below matches any number between 0 and 1200.

```
([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)
```

Note: For a complete list and explanation of supported regular expressions characters, see the Regular Expressions section in the Microsoft VBScript documentation (select **Help > HP Service Test Help** to open the Service Test Help. Then select **VBScript Reference > VBScript > VBScript User's Guide > Introduction to Regular Expressions**).

Regular Expressions in the Find and Replace Dialog Boxes

You can use regular expressions in the **Find text** fields to enhance your search in both the "Find Dialog Box" (described on 152) and "Replace Dialog Box" (described on 155).

You can insert a regular expression into your search string by selecting one from a predefined list, or by entering one manually. For a general understanding of regular expressions in Service Test, see "Regular Expressions Overview" on page 136.

For more details about find and replace functionality, see "Searching and Replacing in the Editor " on page 135.

Note: Not all the regular expressions listed for other parts of Service Test are supported for the search strings in the Find and Replace dialog boxes.

The following table lists the regular expressions supported for search strings, as well as descriptions of each regular expression. For details, see "Regular Expression Characters and Usage Options" on page 161.

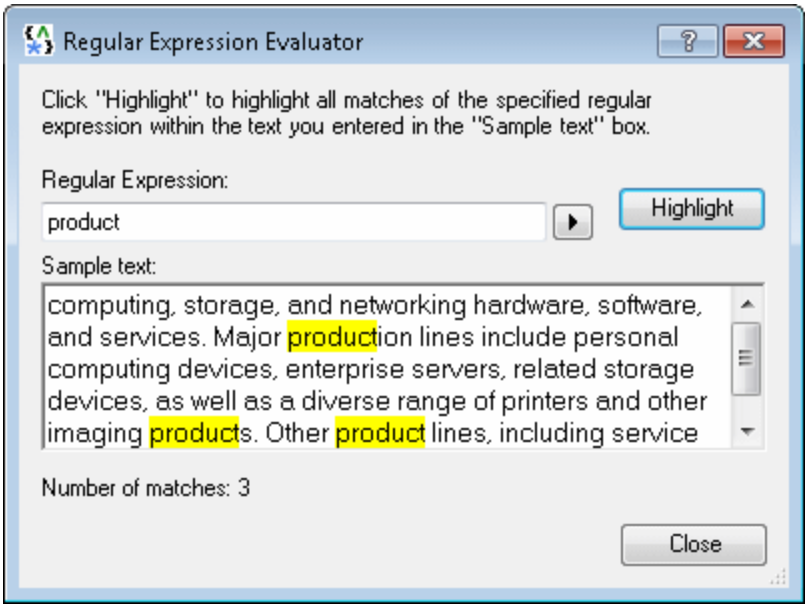
Character	Description	Reference
. (period)	Matches any single character	"Matching Any Single Character (.) " on page 161
* (asterisk)	Matches zero or more specific characters	"Matching Zero or More Specific Characters (*)" on page 162
+ (plus sign)	Matches one or more specific characters	"Matching One or More Specific Characters (+)" on page 162
? (question mark)	Matches zero or one specific character	"Matching Zero or One Specific Character (?)" on page 162

Character	Description	Reference
^ (caret)	Matches the beginning of a line	"Matching the Beginning of a Line (^)" on page 163
$ (dollar sign)	Matches the end of a line	"Matching the End of a Line ($)" on page 163
\n	Matches a line break	"Matching a Newline or Carriage Return Character (\n) or (\r)" on page 163
\r	Matches a carriage return	"Matching a Newline or Carriage Return Character (\n) or (\r)" on page 163
[]	Matches any single character in a list	"Matching Any Single Character in a List ([xy])" on page 161
[^]	Matches any single character not in a list	"Matching Any Single Character Not in a List ([^xy])" on page 162
\w	Matches any alphanumeric character including the underscore	"Matching Any AlphaNumeric Character Including the Underscore (\w)" on page 164
\W	Matches any non-alphanumeric character	"Matching Any Non-AlphaNumeric Character (\W)" on page 164
\d	Matches a decimal digit	"Matching a Decimal Digit (\d)" on page 164
\D	Matches an integer	"Matching an Integer (\D)" on page 164
|	Matches one of several regular expressions	"Matching One of Several Regular Expressions (|)" on page 163
\	Matches a special character treated as a literal character, or a literal character treated as a special character	"Using the Backslash Character (\ ) " on page 161

Regular Expression Evaluator

This dialog box enables you to create and test a regular expression to determine whether it suits your needs. You can enter a regular expression and sample text to test. When you click **Highlight**, Service Test searches the sample text for matches, highlights these matches, and displays the number of matches.

The following example searches for the word `product` followed by a space or other character because the regular expression includes the period (**.**) character.



To access	1. Do one of the following: ▪ Ensure that a test, action, or component is in focus in the document pane. ▪ In the Solution Explorer, select a test or component node. 2. Use one of the following: ▪ Select the **Tools > Regular Expression Evaluator** menu command. ▪ Click the **Smart Regular Expression** button to the right of a value box in any dialog box (except the Find and Replace dialog boxes) in which the **Regular Expression** check box is selected. Select the **Open Regular Expression Evaluator** list item.
Important information	The list of selectable regular expression characters available from this dialog box is also available from other Service Test dialog boxes that contain a selected **Regular Expression** check box and the **Smart Regular Expression** button . **Note:** The Regular Expression Evaluator is not supported for regular expressions in the Find and Replace dialog boxes.
See also	**Conceptual overview:** "Regular Expression Characters and Usage Options" on page 161 **Additional related topics:** "Smart Regular Expression List" on next page

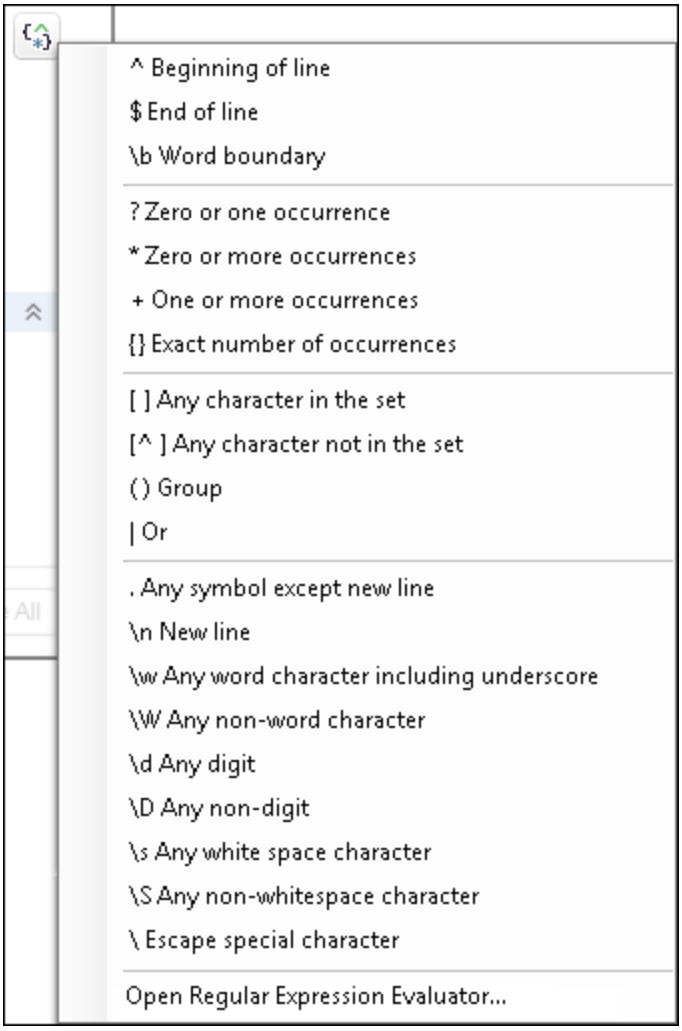
User interface elements are described below:

UI Elements	Description
	**Smart Regular Expression.** Click to display a list of regular expression characters that you can select. For details on each of the regular expression characters, see "Smart Regular Expression List" below.
Regular Expression	The regular expression to test. Enter the regular expression in this box. **Tip:** Click the right arrow  to display a list of regular expression characters that you can insert.
Highlight	Searches for the regular expression in the **Sample text** area and highlights all matches. **Note:** Before you click **Highlight**, make sure that the text to search is displayed in the **Sample text** area.
Sample text	The text to use when testing the regular expression. **Note:** Text may already be displayed in the **Sample text** area when the dialog box opens, depending on the steps you performed to open this dialog box. The **Sample text** area is editable. You can insert any text to test your regular expression.
Number of matches	The number of matches for the regular expression within the sample text. These matches are highlighted in the **Sample text** area.

Smart Regular Expression List

This list:

- Displays a list of commonly used regular expression characters.
- Enables you to select a regular expression character from the list and insert it in a value.
- Enables you to access the "Regular Expression Evaluator" (described on page 166).



To access	Click the **Expression Builder** button to the right of a value box in which regular expressions can be used, such as the Find dialog box.
See also	"Regular Expressions Overview" on page 136 and "Regular Expression Characters and Usage Options" on page 161

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

UI Elements	Description
<Regular expression options>	The list of regular expression characters that you can select when creating a regular expression. When you select a regular expression character from the list, Service Test inserts it in the currently open dialog box and closes the list. Depending on its type, a regular expression character can be inserted: • At the cursor location • Around selected text • In place of selected text **Example 1:** If you select `text` and then select the **[ ] Any character in the set** option, the brackets are inserted on either side of the selected text, as follows: `[text]` **Example 2:** If you select text and then select the **\d Any digit** option, the selected content is replaced with **\d**. For example, if you select `1` in `Document1` and then select the **\d Any digit** option, the selected content is replaced with **\d**, as follows: `Document1` becomes `Document\d` **Note:** The **\n New line** option is available only if applicable. For details on working with regular expression characters, see "Regular Expression Characters and Usage Options" on page 161.

Chapter 8: Errors Pane

This chapter includes:

Concepts

Errors Pane Overview

The Errors pane displays a list of errors generated when opening, working with, saving, and running tests, components, and user code files. The Errors pane also displays the error severity levels: `Message`, `Warning`, or `Error`. For user interface details, see "Errors Pane User Interface" on page 177.

The following types of errors can occur:

- "Code syntax errors" below
- "Missing References " below
- "Missing Property Values " below

Code syntax errors

Service Test checks for syntax errors whenever you save a test, component, or user code file. If a syntax error is present in your code, the error description is displayed as an `Error` in the Errors pane.

You can view a description of code syntax to help you resolve code errors that are displayed in the Errors pane in the Microsoft C# Reference. For details, see http://msdn.microsoft.com/en-us/library/ms228296(v=vs.90).aspx.

For details, see "How to Manage Errors in the Errors Pane" on page 175.

Missing References

In the Solution Explorer, each Service Test test contains numerous reference files used in the run session of the test. In addition, you can add additional references to a test. For details on the **References** node in the Solution Explorer, see "References Node" on page 243.

When aService Test test is run, Service Test checks that all necessary reference files are present. If a reference file needed for the run session is not present, then this missing reference is displayed in the Errors pane, along with the location where Service Test expected to find the reference.

> **Note:** If a reference file is located on a password-protected network host, the reference is listed as missing in the Errors pane unless you add the resource prior to opening the test.

The Errors pane enables you to find the name of the missing reference file and add it to the test. After you save and run the test again, the error disappears from the Errors pane.

For details on adding references to a test, see "Add Reference Dialog Box" on page 245.

Missing Property Values

When you add certain Service Test test steps to the canvas, you are also required to provide input values and property definitions in the "Input/Checkpoints/Output Properties Tab (Properties Pane)" of the Properties pane (described on page 201) After you add these steps to the canvas, an error message is displayed in the Errors pane explaining the necessary property value to enter.

You can double-click on the error description in the Errors pane to navigate to the field necessary to resolve the error. After entering the requested values, the error is longer listed in the Errors pane.

For details on input properties for Service Test test steps, see "Standard Activities" on page 396.

Tasks

How to Manage Errors in the Errors Pane

This task describes the different operations that can be performed in the Errors Pane and includes the following steps:

- "Select the errors to display" below
- "Locate syntax errors" below
- "Locate a missing reference file" below
- "Locate a missing test step property value" below

Select the errors to display

Use the drop-down list at the left side of the pane to select the errors to display in the pane. You can choose to display all errors in the current solution, or sort by individual tests.

Locate syntax errors

In the Errors pane, do one of the following:

- Double-click the error description.
- Right-click the error description and select **Locate**.
- Select the line containing the syntax error and click the **Locate** button.

In the document pane, the cursor jumps to the source of the syntax error. For user code files, the characters containing syntax error are also underlined.

Locate a missing reference file

1. Locate the source of the missing reference files, as described in the step"Locate syntax errors" (described above).

   The relevant test code file opens and the cursor flashes at the place in which the missing reference file is called during a test run displaying the reference file's name.

2. In the Solution Explorer, right-click the **References** node located under an Service Test test and select **Add Reference**.

3. In the "Add Reference Dialog Box" (described on page 245), navigate to the missing reference file and associate it with your test.

Locate a missing test step property value

1. Locate the source of the missing reference files, as described in the step (described above).

   The field requiring the missing property value is highlighted in the "Input/Checkpoints/Output Properties Tab (Properties Pane)" in the Properties pane (described on 201).

2. Enter the required information for the property value.

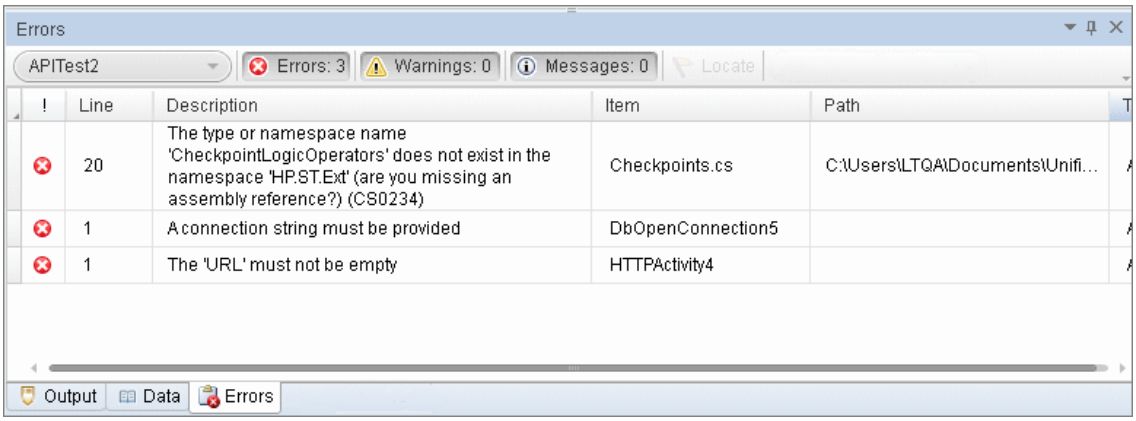
   Note: An alert is also displayed in the step in the canvas. For details, see "The Canvas"

on page 373.

Reference

Errors Pane User Interface

The Errors pane lists the errors found in your test or user code document, and enables you to locate each syntax error so that you can correct it. It also provides a list of the resources that are referenced in your test or component but cannot be found.



To access	Do one of the following: • Select **View > Errors**. • Click the **Errors** tab in the bottom pane.
Important information	• The Errors pane displays the errors for all tests contained in your solution. You can also sort to display errors for each individual document contained within the solution. • Click on a row to jump to the line in the code that generated the error. • Click any column header, for example **Line** or **Item**, to sort by that criteria.
Relevant tasks	• "How to Manage Errors in the Errors Pane" on page 175 • "How to Run an API Test" on page 288
See also	"The Canvas" on page 373

The following sections describe:

- "Main User Interface Elements" below
- "Context Menu Items" on page 179

Main User Interface Elements

The main user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Show errors from>	Filters the message list by the source of the error. **Default:** Solution
Errors: 0	Shows or hides syntax errors, property value errors and missing resources detected during the test run.
Warnings: 0	Shows or hides warnings detected during the run.
Messages: 0	Shows or hides informational messages detected during the run.
Locate	• **Syntax errors:** Jumps to the line in the Editor that contains the error. • **Missing references.** Opens the relevant testing code file to highlight the name of the missing resource. • **Missing property value.** Highlights the input field in the **Input/Checkpoints** tab in the Properties pane. Available also by right-clicking an error line. For details, see "Context Menu Items" on next page.
! <Exclamation Point>	**Message type:** • Error • Warning • Informational message
Line	The line containing the error. The lines are numbered from the beginning of the `TestUserCode.cs` file.
Description	Description of the error, warning or message and advice on how to fix the problem. For example, a syntax error is displayed if you opened a block of code with a `{` but did not close it with an `}`, the description is `Expected end of statement`. **Note:** In certain cases, Service Test is unable to identify the exact error and displays a number of possible error conditions, for example: `Expected 'End Sub'`, or `'End Function'`, or `'End Property'`. Check the statement at the specified line to clarify which error is relevant in your case.

UI Elements	Description
Item	• **Syntax errors.** The name of the user code file containing the problematic statement. • **Missing references.** The name of the test user code that contains the call to the missing reference. • **Missing property values.** The name of the activity containing the missing property value.
Path	The full path of the file that generated the error.
Test	The name of the relevant test or component containing the error.

Context Menu Items

The user interface elements described below are available when you right-click an error in the Errors pane.

UI Elements	Description
Copy	Copies the content of the selected error to the clipboard.
Locate	• **Syntax errors.** Jumps to the location that contains the error. You can double-click a syntax error to locate the error in the relevant user code file, and then correct it. • **Missing references.** Opens the test code file that contains the call to the reference file and highlights the name of the missing reference. • **Missing property value.** Opens the relevant field in the **Input/Checkpoints** tab in the Properties pane. **Note:** Double-clicking a missing resource item has the same effect as selecting the **Locate** option from the right-click menu or from the toolbar.

Chapter 9: Output Pane

This chapter includes:

Concepts

Output Pane Overview

The Output pane displays the following information:

- Details about assets that cannot be located or loaded during a run session.
- Output messages generated in the Output log while compiling and running the test.

For user interface details, see "Output Pane User Interface" on next page.

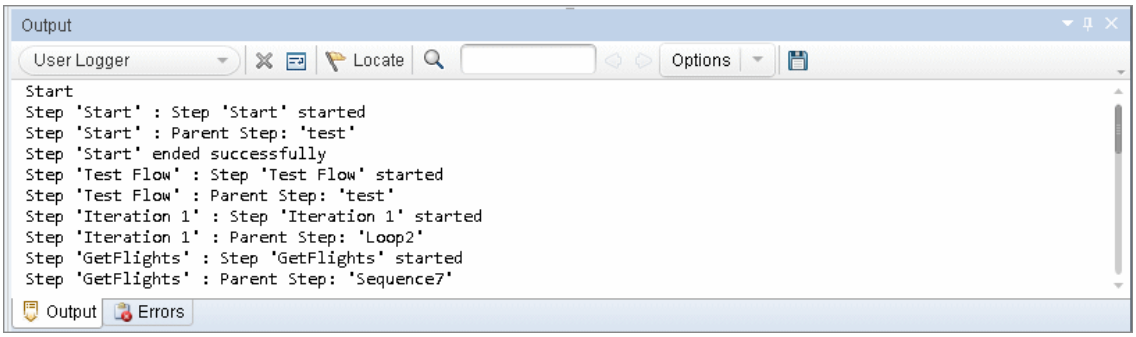
Reference

Output Pane User Interface

The Output pane displays

output messages generated in the Output log while compiling and running the test.

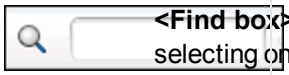
The image below shows the Output pane during an API test run session.



To Access	Do one of the following: • Select **View > Output**. • During a paused run session, click the **Output Pane** toolbar button .
Important information	Click on a row to navigate to the code that generated the output message.
Relevant tasks	"How to Run an API Test" on page 288

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

UI Element	Description
<Show output from>	The type of output to display. The following types are available: • **Build** Displays all test build information. • **Debug:** Includes debug information, such as all **Print** command (print log) outputs and details about tests called from GUI tests.
	Clear All Lines. Clears all of the messages from the message list.
	Toggle Word Wrap. When selected, wraps the text of each message onto the next line.

UI Element	Description
Locate	Jumps to the location in the source document relevant to the selected output message.
	**<Find box>**.The text string you want to find. You can refine your search by selecting one of the "Options" described below. Press ENTER to begin the search.
	**Find Previous / Find Next.** Highlights the next or previous string that matches the text that you entered in the **Find** box. Available only after you enter text in the **Find** box.
Options	Enables you to refine your search with the following options: • **Match Case.** Distinguishes between upper-case and lower-case characters in the search. • **Match Whole Word.** Searches for occurrences that are only whole words and not part of longer words. • **Use Regular Expression.** Treats the specified text string as a regular expression. **Note:** Extended regular expressions and multi-line searches are not supported.
	**Save Output to a Text File.** Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 72), enabling you to save the contents of the message list as a text file.
<Message list>	All messages sent during the test run session. Each message corresponds to a message generated in the Output log.
Copy	Shortcut (right-click) menu option that copies the selected content to the clipboard.
Locate	Shortcut (right-click) menu option that jumps to the relevant line in the source document for the selected output message.

Troubleshooting and Limitations - Output Pane

- The Output tab may be unable to display the run results for very large tests, exceeding a thousand steps.
- The compilation information for a test is not localized.

Chapter 10: Properties Pane

This chapter includes:

Concepts

Properties Pane Overview

The properties pane is used for viewing and editing global and specific properties of your tests and components.

Each step or test activity may contain several customizable properties or parameters. The Properties pane provides several views that allow you to define or assign values to the properties and define handler events.

The Properties pane's toolbar also provides action buttons that allow you to data drive properties and load files.

For details, see "Properties Pane User Interface" on next page.

Reference

Properties Pane User Interface

The Properties pane displays a series of tabs showing the properties, schemas, and events for the step selected in the canvas.

To access	1. Create or open a test or component. 2. Select **View > Properties** or click on the Properties pane tab.
Important information	• Only active when the canvas is visible. • The toolbar displays different buttons based on the type of activity or view. • The top row of buttons are **View** buttons, such as **General** or **Events** views. • The bottom row of buttons are buttons that perform actions, such as **Data-Drive** and **Load XML**.
Relevant tasks	• "How to Create an API Test" on page 383 • "How to Set Security for a Web Service on the Port Level" on page 547

This section includes:

Properties Pane Tabs

The following section describes each of the Properties pane tabs.

Asynchronous Tab (Properties Pane)

This tab enables you to indicate whether the response for a Web Service is asynchronous.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **Asynchronous** tab  .

Relevant tasks	"How to Create a Test for an Asynchronous Web Service" on page 605
See also	"Asynchronous Services" on page 602

The **Asynchronous** tab's user interface elements are described below.

UI Elements	Description
This is an asynchronous call	Indicates whether the call is asynchronous and enables you to specify a listener.
Listen for response on	The port upon which to listen for a response. This property is visible only if you enable the above option.

Attachments Tab (Properties Pane)

This tab enables you to view input and output attachments associated with the current test step.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **Attachments** tab  .
Relevant tasks	"Add attachments to the test - optional" on page 384

The **Attachments** tab'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 name of the file to send as an attachment. Click the **Browse** button to navigate to the directory in which the attachment is saved. **Note:** You can use the ouptut from a different test step as an attachment. Click the **Link to data source** button and choose the test property in the "Select Link Source Dialog Box" (described on 630). ▪ **Content Type.** The type of file and format. Possible file types include: ◦ `application/zip` ◦ `image/gif` ◦ `image/ief` ◦ `image/jpeg` ◦ `image/png` ◦ `image/tiff` ◦ `text/plain` ◦ `text/richtext` ◦ `text/html` ◦ `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 button in the parent node.

UI Elements	Description
<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 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.

Data Sources Tab (Properties Pane)

This tab enables you to set the data navigation instructions for the data sources associated with your test flow.

To access	1. Open the Properties pane. 2. Select the Test Flow in the canvas. 3. In the Properties Pane, select the **Data Sources** tab .
Important information	You can have more data sources associated with a test in the Data Pane than are associated with a particular test flow.
Relevant tasks	"How to Set the Navigation Properties" on page 622
See also	• "Navigating within the Data " on page 615 • "Data Navigation Dialog Box" on page 634

The **Data Sources** tab's user interface elements are described below.

UI Element	Description
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 Pane. • **Policy.** A summary of the data navigation policy, for example: **Start at first row, forward one row, wrap around.**
Add	Opens the "Attach Data Source to Loop Dialog Box " (described on page 629) for selecting data sources from the Data Pane. **Note:** In order to add a define navigation properties for a Data Source, you must associate it with your test in the Data pane. For details, see "How to Create Data Sources" on page 90.
Edit	Opens the "Data Navigation Dialog Box" (described on page 634) for setting the data policy—the way to use the data from the table.
Remove	Deletes the selected data source from the list—not from the Data Pane.

Data Source Properties Tab (Properties Pane)

This tab enables you to define the usage properties for your test's data sources.

To access	1. Open the Properties pane. 2. In the Data Pane, select a data source. This tab appears by default and all other properties pane tabs are hidden..
Relevant tasks	"Create a new child relation" on page 94

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

UI Elements	Description
Name	The name of the data source node.
Child 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. To add, edit, or remove a child relation, use the corresponding buttons. For details, see the see "Define New/Edit Data Relation Dialog Box" on page 637.

UI Elements	Description
Allow other tools to override the data	Allows other tools or tests to override Excel data. Enabling this option allows you to overwrite the data table values with ALM Test Resources. For details, see "Data Awareness in ALM" on page 324. This option also allows action data to be edited when called by another test. For details, see "Data Pane" on page 96. **Note:** This only applies to Excel data sources. **Caution:** For Excel data sources with multiple sheets, you must enable this option for each data sheet if you did not enable it when importing the Excel file.
Data source policy	The policy by which to handle the data (only relevant for steps that call actions from an external test, to which data was assigned). • **Use data stored with original action (read-only)**. Use the data that is associated with the action, as is. • **Use a local, editable copy**. Create of local copy of the action's data, to allow you to edit the values in the Data Pane. **Note:** This only applies to Excel data sources.

Database Data Source Properties Tab (Properties Pane)

This tab enables you to view the properties for your database data sources.

To access	1. Open the Properties pane. 2. Select a Database data source in the Data pane.
Relevant tasks	"Add a database data source" on page 91
See also	"Add New Database Data Source Wizard" on page 102

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

UI Elements	Description
Name	The name of the data source (read-only).
Connection type	The type of connection: **OleDB** or **ODBC**.

UI Elements	Description
Connection string area	The **Connection string** section contains the following elements: • **Connection string.** A drop down list of connection strings that were defined. • **Build Connection String** . ▪ For **OleDB** types, it opens Microsoft's Data Links Properties dialog box. ▪ For **ODBC** types, it opens the "Select Data Source Name Dialog Box", described on 105. • **Edit** . 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 there is an open connection to the database.
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. • **Build Query Statement.** Opens the **Query Designer**. For details, see the "Query Designer Dialog Box" on page 456. • **Edit** . 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 the query was executed successfully.

Dependencies Tab (Properties Pane)

This tab shows a list of all external resources (Web Services, REST services, and .NET assemblies) used in your test flow.

To access	1. Open the Properties pane. 2. Select the **Start** or **End** steps in the canvas. 3. In the Properties Pane, select the **Dependencies** tab .

The **Dependencies** tab's user interface elements are described below.

UI Element	Description
Resource Name	The name of the resource as imported into the test
Type	The type of resource, for example, WSDL.
Location	The location in the test in which the resource is used.

Events Tab (Properties Pane)

This tab enables you to enter custom code to run special custom code in conjunction with the steps of your test.

To access	1. Open the Properties pane. 2. Select a test step in the canvas. 3. In the Properties Pane, select the **Events** tab .
Relevant tasks	"How to Open a Window for Writing Custom Code" on page 644
See also	"Writing Code for Events Overview" on page 642

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. • **ExecuteEvent.** Triggered when the activity isn the step is run. **Tip:** For details about an event, select it and select **Create a default handler**. Refer to comments in the template.
<Events list> - Conditional. Loop steps	**Condition.** Defines the conditions under which the code should be run.
<Events list> - HTTP Receiver	• **OnFilter.** The code to execute when the messages are filtered. • **ReceiveRequest.** The code to execute when a request is received. • **SendResponse.** The code to execute when a response is sent.

UI Elements	Description
<Events list> - IBM Websphere MQ	• **BeforeCreateQueueManager.** Code to execute before creating a Queue Manager. • **BeforeMQGet.** Code to execute before getting the messages from the MQ queue through browsing. • **AfterMQGet.** Code to execute after getting the messages from the MQ queue through browsing. • **BeforeMQPutMessage.** Code to execute before putting a message from the MQ queue. • **AfterMQPutMessage.** Code to execute after putting a message from the MQ queue. • **BeforeMQGetMessage.** Code to execute before getting a message from the MQ queue. • **AfterMQGetMessage.** Code to execute after getting a message from the MQ queue. • **BeforeMQPublishMessage.** Code to execute before publishing a message to an MQ topic. • **AfterMQPublishMessage.** Code to execute after publishing a message to an MQ topic. • **BeforeMQReceiveMessage.** Code to execute before receiving a message published on an MQ topic. • **AfterMQReceiveMessage.** Code to execute after receiving a message published on an MQ topic. For task details, see "How to Retrieve Messages from an MQ Queue" on page 408.
<Events list> - Wait	• **TimeoutReached.** The code to execute when the timeout (Input properties) is reached.

Excel File Properties Tab (Properties Pane)

This tab enables you to change the Excel file associated with your Excel data sources.

To access	In the Data pane, select the parent node for an Excel data source. The other Properties pane tabs are hidden.
Relevant tasks	Add list of cross references using bullets for multiple entries. Do not use periods.

The user interface elements are described below.

UI Elements	Description
Name	The name of the data source as it appears in the Data pane (read-only).
File Path	The current path to the Excel file (read-only).
Change Excel File	Opens the "New/Change Excel Data Source Dialog Box" (described on 98) which enables you to select a new file for the selected data sources.

Filter Settings Tab (Properties Pane)

This tab enables you to set a filter for received messages. This tab is available for **HTTP Receive** steps and operations from Web Service calls imported as server response steps.

To access	1. Open the Properties pane. 2. Select an HTTP Recieve or Web Service step in the canvas. 3. In the Properties Pane, select the **Filter** tab .
See also	"Asynchronous Services" on page 602

The **Filter Settings** tab's user interface elements are described below.

UI Elements (A-Z)	Description
	Clears the text in the filter area.
	Opens the "Select Link Source Dialog Box" (described on page 630), to allow you to link to existing data.
<filter text>	A text area containing the filtering expression. You can enter free text or a regular expression and use the Select Link Source dialog box to create a data expression. **Note:** When filtering request headers, you can only filter requests that contain both a key and value. Headers that have a key, but no value, cannot be filtered.
Filter Type	The type of filtering: **Exact match**, **Starts with**, **Ends with**, **Contains**, and **Regular expression**. **Note:** If you specify **Exact Match** and you are trying to match a key-value pair, you must specify both the key and value strings.

General Tab (Properties Pane)

This tab displays general information about the selected test step.

To access	1. Open the Properties pane. 2. Select any step in the canvas. 3. In the Properties Pane, select the **General** tab  .

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

For details about activity-specific general properties, see the specify activity in "Standard Activities" on page 396.

UI Elements (A-Z)	Description
Comment	An editable note describing the purpose of the step. Service Test also places the comment in the test report.
Properties - default	• **Step ID.** A unique ID for the step. • **Name.** Step name as it should appear in the canvas.
Properties - HP Automated Testing Tools	• **Test path.** The path of the API or GUI test or VuGen script (LoadRunner's Virtual User Generator) to run. • **Action name.** The name of the action in the test (not relevant for Virtual User Generator scripts). • **Description.** A description of the step. • **Result directory.** The path of the results relative to the solution directory (For **Call API Action or Test** only).

General Properties - Web Services and SOAP Request

The following table lists the General properties specific for Web services.

Property	Description
Transport	The transport mode: **HTTP** or **JMS**.
HTTP	Use the HTTP transport method for this call with the following properties: • **Endpoint Address.** The endpoint location of the service as derived from the WSDL file. • **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. • **Timeout.** The maximum time in milliseconds to wait for the response. Enter -1 to disable the timeout and instruct the client to wait for the response as long as required.

Property	Description
JMS	Use the JMS transport method for this call with the following properties: • **Send queue name.** The name of the queue from which to send the message. • **Receive queue name.** The name of the queue from which to receive the message. • **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 details on the syntax for defining selectors, see the **Message Properties** section in http://download.oracle.com/javaee/1.3/api/javax/jms/Message.html. For example, a key `JMSCorrelationID`, with a value of `4566636`, receives only messages whose correlation ID is `4566636`.
IsOneWay	Indicates whether the service is a one way service: `true` or `false`. A one way service only sends a request. A non-one way service sends a request and receives a response.

General Properties - REST Services/ HTTP Request

The following properties are available in the Add REST Service dialog box and the Properties pane of HTTP Request steps. For details, see "Add/Edit REST Service Dialog Box" on page 495 or "Network Activities" on page 426.

Property (A-Z)	Description
Allow redirections	Indicates whether to allow the step to redirect to another URL. **Default:** `true`
Authentication	The user credentials settings for obtaining a service contract: **Username** and **Password**.
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. **Note:** To use a certificate, make sure to set the **Use Client Certificate** property to `true`.

Property (A-Z)	Description
Connection type	The type of connection: **Keep-Alive** or **Close**. **Default:** Keep-Alive
Maximum automatic redirections	The number of times to attempt accessing a page, including redirection. **Default:** 3
Proxy	The settings for the proxy server hosting the service contract: **Server** (URL and port when required), **Username**, and **Password**.
Reuse cookies	Enables the reusing of cookies for the current step. **Default:** `false`.
Timeout	The HTTP timeout in milliseconds.
Use Client Certificate	Enables the use of a client certificate. **Default:** `false`

General Properties - SAP

The following properties are available in the Properties pane for SAP IDOC or RFC steps. For details, see "Import from SAP/Update Dialog Box" on page 502.

Property (A-Z)	Description
Timeout	The maximum time allowed for the function to respond in milliseconds. **Default:** `100000`
ConnectionInfo	The SAP Connection information for the selected step: • **Server.** The name or IP address of the server. • **System number.** The server's System Number. • **Client.** The client for this SAP connection. • **Username.** The username for this SAP connection. • **Password.** The password for this SAP connection.
Function name	The name of the RFC (for RFCs only).
Expect Exception	Indicates whether or not to expect an exception when running the function: `true` or `false` (for RFCs only).
IDocController	IDoc Controller properties (for IDocs only): • **Message Type**, **Basic Type**, and **Extension**. • **Sender:** `Port`, `Partner number`, and `Partner type`. • **Receiver:** `Port`, `Partner number`, and `Partner type`.
Transaction	Indicates whether or not the RFC is part of a transaction or not: `true` or `false` (for RFCs only).

HTTP Tab (Properties Pane)

This tab enables you to define request and response data for your test steps that perform an HTTP Request to a Web Service or REST service.

To access	1. Open the Properties pane. 2. Select an HTTP Request or REST service step in the canvas. 3. In the Properties Pane, select the **HTTP** tab .
Relevant tasks	"How to Create a REST Method" on page 475
See also	"Network Activities" on page 426

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

UI Elements	Pane Body type	Description
	Request/Response: • XML • Text • File • JSON • Post Form (Request only)	**Remove.** Deletes the contents of the: • Body text • Entered path • Selected value in grid
...	Request/Response: • File	**Browse.** Enables you to load a non-XML file containing the request/response body.
	Request/Response: • XML • Text (Request only) • File • JSON	**Link Body.** Opens the "Select Link Source Dialog Box" (described on page 630) for selecting a data source.

UI Elements	Pane Body type	Description
<checkpoint list>	Response pane	A list of the checkpoints with the following information: • **Name.** The name of the checkpoint as it will appear in the results. • **Regular Expression.** A regular expression representing the expected value. • **Validate.** Compares the actual value with the expected value.
<message body>	Request/Response panes • XML • Text • JSON	The body of the request or response loaded through a file, pasted from a clipboard (**Edit > Paste**), or manually entered (Post form).
<post form list>	Request pane: • Post Form	A list of the Post form elements: • **Name.** The name of the element. • **Value.** The element's value.
Clear	Request/Response: • XML	Clears the contents of the **Request/Response Body** areas.
Import Schema	Request/Response: • XML	Imports an `.xsd` file containing the schema of the request or response.
Load File	Request pane: • Text	Loads a non-`.xml` file containing the request.
Load JSON	Request/Response: • JSON	Loads a `.json` file containing the request or response.
Load XML	Request/Response: • XML	Loads an `.xml` file with the request or response body.
Request Body	Request pane	The format of the request body: **XML**, **Text**, **File**, **JSON,** or **PostForm.**
Response Body	Response pane	The format of the response body: **XML**, **Text**, **File,** or **JSON.**

HTTP Receiver Tab (Properties Pane)

This tab enables you to define response data for steps that receive a response from a HTTP or SOAP-based Web Service.

To access	1. Open the Properties pane. 2. Select an HTTP Receiver step in the canvas. 3. In the Properties Pane, select the **HTTP Receiver** tab  .
See also	"Network Activities" on page 426

User interface elements are described below (unlabeled elements are shown in angle brackets).\

UI Elements (A-Z)	Response Body type	Description
	**Text**	**Clear.** Clears the contents of the selected cells in regular expression grid.
<body of received message>	**XML, JSON**	The body of the response loaded through a file.
<regular expression>	**Text**	A grid with regular expression containing the response body values to validate.
Clear	**XML, JSON**	Clears the contents of the body.
Import Schema	**XML**	Imports an `.xsd` file containing the schema of the response.
Load JSON	**JSON**	Loads a `.json` file with the response values.
Load XML	**XML**	Loads an `.xml` file with the response values.
Received Message Body	**XML, Text, JSON**	The way in which to represent the response: **XML, Text**, or **JSON.**

Input/Checkpoints/Output Properties Tab (Properties Pane)

This tab enables you to define input properties, input parameters, and checkpoint properties for your test steps.

To access	1. Open the Properties pane. 2. Select a test step in the canvas. 3. In the Properties Pane, select the tab. This tab's name and content depend on the test step you selected. ▪ If you selected a **Start** or **End** steps, this tab is **Test Input/Output Properties**. ▪ If you selected the test flow, this tab is **Input**. ▪ If you selected any other step, this tab is **Input/Checkpoints**.

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

For details about activity-specific properties, see "Standard Activities" on page 396.

UI Elements (A-Z)	Description
<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. • **Expected 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 (A-Z)	Description
<context menu>-input properties	Provides shortcuts for including properties and setting their values. • **Collapse/Expand All.** Controls the display of array elements. • **Set Auto-value.** Inserts a sample value for the argument, based on its data type. • **Include/Include All.** Includes the current or all Choice input properties in the test run. • **Exclude/Exclude All.** Excludes the current or all Choice input properties from the test run. • **Copy XPath.** Copies a simplified XPath expression to the clipboard. For details, see "XPath Checkpoints" on page 586. • **Copy Fully Qualified XPath.** Copies the complete XPath expression of the value to the clipboard. • **Link to Data Source.** Opens the "Select Link Source Dialog Box" on page 630 • **Clear Cell Contents.** Clears the contents of the cell. **Note:** Some options are only available for specific property types.

UI Elements (A-Z)	Description
<context menu> - checkpoints	Provides shortcuts for including properties and setting their values. • **Remove Array Element.** Removes the selected array element. • **Duplicate Array Element.** Duplicates the selected array element with its values. • **Select All/Expand All.** Selects/clears the Validate check box for all child array elements. • **Collapse/Expand All.** Controls the display of array elements. • **Set Auto-value.** Inserts a sample value for the argument, based on its data type. • **Copy XPath.** Copies a simplified XPath expression to the clipboard. For details, see "XPath Checkpoints" on page 586. • **Copy Fully Qualified XPath.** Copies the complete XPath expression of the value to the clipboard. • **Link to Data Source.** Opens the "Select Link Source Dialog Box" (described on 630). • **Display Outgoing Links.** Lists the property's outgoing links. For details, see "Outgoing Links" on page 612. • **Insert Keyword.** Insert a keyword in the **Expected Value** column. The available keywords are `#NIL#`, `#EXISTS#`, `#NOT_FOUND#`, and `#SKIP#`. For details, see "Data Keywords" on page 617.
<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 "Loop - Input Properties" on page 418.

UI Elements (A-Z)	Description
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 enables you to select values when appropriate. For a list of the most common icons, see the "Array Control Buttons" on page 216. **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.

Web Service and SOAP Request Checkpoint Options

The following options only apply to **Web Service** and **SOAP Request** steps.

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 enabled by default. To send messages using JMS transport, clear this check box. Link a subsequent JMS step to the output of this step. For Web services, you can set the security settings and include attachments, and send this over JMS.

UI Elements	Description
Validate Structure	Adds a checkpoint that verifies 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. This option is available only when the **Send request to server** option is enabled. It is enabled by default. **Note:** 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	Adds a checkpoint to verify that the SOAP response in compliance with WS-I standards. The Run Results Viewer indicates whether the response was in compliance. It also provides a **View Report** link that opens the WS-I Validation report in a separate window.

XPath Checkpoint Options

The following options are only relevant for steps with XML output properties, such as **String to XML**.

UI Elements	Description
XML tab	A grid representation of the response's schema. In this section you can enter the expected response values. • **Import Schema.** Imports a schema for the XML response. • **Load XML.** Loads an XML file as a basis for the schema of the response. • **Clear.** Removes the displayed schema.
XPath tab	A list of XPath expressions used to evaluate the XML response. • Adds a line for a new XPath expression. • Removes the selected XPath expression. • **Ignore namespaces:** Ignores namespaces in the XPath validation, allowing you to use simple XPath expressions. For details, see "How to Set XPath Checkpoints" on page 592.

Database Property Options

The following options are only relevant for properties of the **Database > Select Data** step.

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 455.
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.

Multipart Tab (Properties Pane)

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

To access	1. Open the Properties pane. 2. Select an HTTP Request in the canvas. 3. In the Properties Pane, select the **Multipart** tab  .
Relevant tasks	"How to Send a Multipart HTTP Request" on page 401

User interface elements are described below:

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

For details about the input and output properties, see "Network Activities" on page 426.

Result Tab (Properties Pane)

This tab enables you check the output of **String to XML** and **String to JSON** activity steps.

To access	1. Open the Properties pane. 2. Select a **String to XML** or **String to JSON** step in the canvas. 3. In the Properties Pane, select the **Results** tab  .
See also	• "XML Activities" on page 451 • "JSON Activities" on page 430

User interface details are shown below:

UI Elements	Description
Result	A boolean variable indicating whether the string had the correct structure to be converted into XML or JSON.
Validate	Instructs Service Test to check if the expected output is correct.
Expected Value	The expected result of the step: **True**, **False**, **0**, or **1**.

Security Tab (Properties Pane)

This tab enables you to configure the security settings for your Web Service activities.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **Security Settings** tab .
Relevant tasks	• "How to Set Security for a Web Service on the Port Level" on page 547 • "How to Set Security for a Specific Step" on page 547
See also	• "Setting Security Overview" on page 540 • "Security Settings for Port <Port_Name> Dialog Box" on page 559

User interface elements are described below:

UI Element	Description
Use the port's security settings	Enables you to use the security settings from the currently selected step's port. **Note:** If you clear this check box, you must configure the security settings in this tab.
Save	Saves the security settings.
Import	Next, unlabeled elements in angle brackets. Use sentence caps, in alphabetical order. Use thumbnails/popups as necessary.
<security settings area>	Enables you to set security information for the Web Service activity. For full user interface details, see "Security Settings for Port <Port_Name> Dialog Box" on page 559.
Edit port's settings	Enabled only when the Use the port's security settings option is selected. Opens the "Security Settings for Port <Port_Name> Dialog Box" (described on 559), enabling you to configure the security information used for the Web Service's port.

SOAP Fault Tab (Properties Pane)

This tab enables you to set the expected result of an Web Service or SOAP activity.

To access	1. Open the Properties pane. 2. Select a Web Service or SOAP step in the canvas. 3. In the Properties Pane, select the **SOAP Fault** tab .
See also	• "Network Activities" on page 426 • "Negative Testing" on page 586

User interface elements are described below.

UI Elements (A-Z)	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.

Test Settings Tab (Properties Pane)

This tab enables you to set global preferences and properties for your test.

To access	1. Open the Properties pane. 2. Select a step in the canvas. 3. In the Properties Pane, select the **Test Settings** tab .
Important information	Test settings apply to all steps in the test.

User interface elements are described below.

Test Settings Tab - Action Buttons

UI Elements (A-Z)	Description
Export	Exports the current Test settings to an XML file.
Import	Imports an XML file with previously saved Test settings.

Load Settings

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

JVM (Java Virtual Machine) Settings

These settings are relevant for all Java related activities such as JMS and **Call Java Class** activities.

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

JMS Settings

UI Elements (A-Z)	Description
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.
JMS connection factory	The JNDI name of the JMS connection factory. This setting is unique for each test.

UI Elements (A-Z)	Description
JMS security credentials	The principal's credentials for the authentication scheme.
JMS security principal	Identity of the principal (for example the user) for the authentication scheme.
JNDI initial context factory	The fully qualified class name of the factory class that will create an initial context. It provides a list of context factories and enables manual entries.
JNDI provider URL	The URL of the service provider. For example: Websphere - `iiop://myserver:myport`
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. • **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.
User defined timeout	The amount of seconds to wait for the message before timing out. The default is 20 seconds.

.NET Settings

UI Elements	Description
Assembly paths	The .NET assembly paths for the test. This entry should include all relevant folder paths and environment variables. Separate multiple values with semicolons.

General Settings

UI Elements	Description
Stop test on step failure	Exits the test if a step fails during the test run. **Default:** `True`

Reporting Settings

UI Elements	Description
Use the reporting mechanism	Tells Service Test to automatically open the Run Results Viewer after a test run.

Test Variables Tab (Properties Pane)

This tab enables you to set global test variables for use in your test.

To access	1. Open the Properties pane. 2. Select the **Start** or **End** step in the canvas. 3. In the Properties Pane, select the **Test Variables** tab  .
Relevant tasks	"How to Define Test Properties or User/System Variables" on page 59
See also	• "New Test Profile Dialog Box" on page 74 • "Manage Profiles Dialog Box" on page 75

User interface elements are described below.

UI Elements (A-Z)	Description
System variables	A read-only list of common system variables and their values: • `ScenarioID` • `SystemTempDir` • `GroupName` • `TestDir` • `TestName` • `LocalHostName` • `OS` • `OSVersion` • `ProductDir` • `ProductName` • `ProductVer` • `UserName`.
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.

Test Variables Tab - Action Buttons

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.

XML Body Tab (Properties Pane)

This tab enables you to configure the response XML data for a SOAP Request step.

To access	1. Open the Properties pane. 2. Select a Web Service step in the canvas. 3. In the Properties Pane, select the **XML Body** tab  .
See also	"Network Activities" on page 426

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

UI Elements	Description
Grid	Shows a grid view of the schema.
Text	Opens an editable text view for the schema.
Revert	Text view only—reverts back to the last version of the schema since you last opened the Text view.
	**Syntax error.** Data at the root level is invalid. Move the mouse over the icon to see the error's line number (Text view only).
	**Syntax warning.** A warning indicating a problem in the entered text, for example, if the specified element is not in the namespace. For details, move the mouse over the icon (Text view only).
Import Schema	Imports an `.xsd` file representing the schema, for both Input properties and checkpoints.

UI Elements	Description
Load XML	Loads an `.xml` file containing values for the schema, for both Input properties and checkpoints.
Clear	Deletes the contents that you entered for the schema. This affects both the Grid and Text views.
Input pane	A grid or text representation of the schema's request header and body.
Checkpoints pane - XML view	A grid representation of the response's schema. In this section you can enter the expected response values. • **Import Schema.** Imports a schema for the XML response. • **Load XML.** Loads an XML file with the expected response values. • **Clear.** Deletes the contents of the schema.
XPath Checkpoints pane	A list of XPath expressions used to evaluate the XML response. • Adds a line for a new XPath expression. • Removes the selected XPath expression. **Ignore namespaces:** Ignores namespaces in the XPath validation, allowing you to use simple XPath expressions. For details, see "How to Set XPath Checkpoints" on page 592. **Note:** To retrieve an XPath expression, use the right-click menu options in the Grid view.

Action Buttons

The following buttons represent actions, primarily performed on step properties. The availability of the buttons depends on the step type.

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 623.
	Edit Property. Opens the Edit Property dialog box for changing the name, type, or description of a property. **Note:** Only available for Custom Code, REST method, and Start/End steps.

UI Elements	Description
	**Load from Replay.** Loads data from a recent replay in order to populate the service's properties with values (for Web Services only). For details, see "How to Run an API Test" on page 288.
	**Remove Property.** Deletes the selected property from the list. **Note:** Only available for Custom Code, REST method, and Start/End steps.
	**Add Input/Output Property/Parameter.** Opens the "Add Input/Output Property/Parameter Dialog Box" (described on page 497), allowing you to define a new property and its data type. **Note:** Only available for Custom Code, REST method, and Start/End steps, or when selecting the entire canvas. For details, see "Coding Service Test Events" on page 641.
Load XML	Loads XML data in order to populate a service's schema with values (for Web Services only). For details, see "How to Run an API Test" on page 288.
Load options	Loads an XML file containing the **Compare XML** ignore settings that you saved with another test.
Save options	Saves the **Compare XML** ignore settings so that you can load them into a future test.
Select GUI/API Test	For **Call GUI Test** and **Call API Test** steps: Invokes the "Select Action or Test Dialog Box" (described on page 516), enabling you to selecting a test to call from this step.
Select Java File	Opens the "Java Class Dialog Box" for setting additional classpaths for the call (for Call Java Class steps only). For details, see the "Java Class Dialog Box" on page 459.
Refresh GUI/API Test	For **Call GUI Test** and **Call API Test** steps: Updates the test from its original location.
Load JSON	Loads a JSON file to convert to or from a string. **Note:** Only available for **JSON to String** and **String to JSON** steps.

Value Column Icons

The following buttons represent information, drop-down lists, or actions within the Properties pane's **Value** column:

UI Elements	Description
	**Include argument in the request.** Toggle to clear the triangle and exclude the argument.
	**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.`
	• 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.** Enables you to locate a file or folder for example, when using a **File System** type step. For an **Open Connection** step, this opens the "Connection Builder Dialog Box" (described on 453).
	**Link to a data source.** Opens the "Select Link Source Dialog Box" (described on 630), allowing you to select values for the property from a data source.
	**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 612.

Array Control Buttons

The following buttons allow you to handle array type properties. These buttons are adjacent to the property name in the Properties pane's left pane.

After you add array elements, you can set the number of iterations to use the different array values. For details, see "Flow Control Activities" on page 417.

UI Elements	Description
	**Add array element.** Adds one array element to the selected node in the in the **Properties** tree.
	**Remove array element.** Removes the selected array element from the **Properties** tree.
	**Duplicate array element.** Adds a copy of the selected array element.

Troubleshooting and Limitations -Properties Pane

The tooltips displayed in the Properties pane are not localized.

Chapter 11: Run Step Results Pane

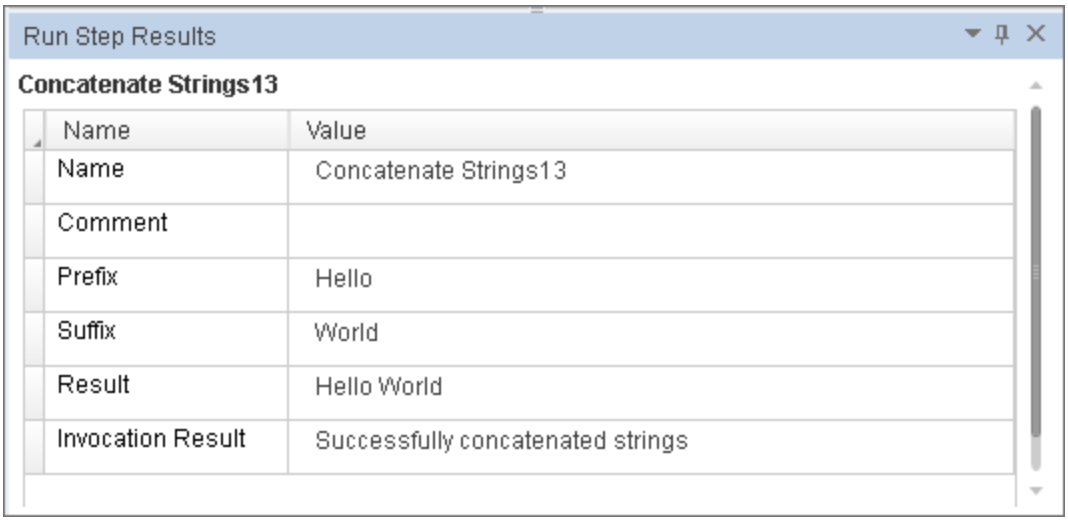
This chapter includes:

Concepts

Run Step Results Pane Overview

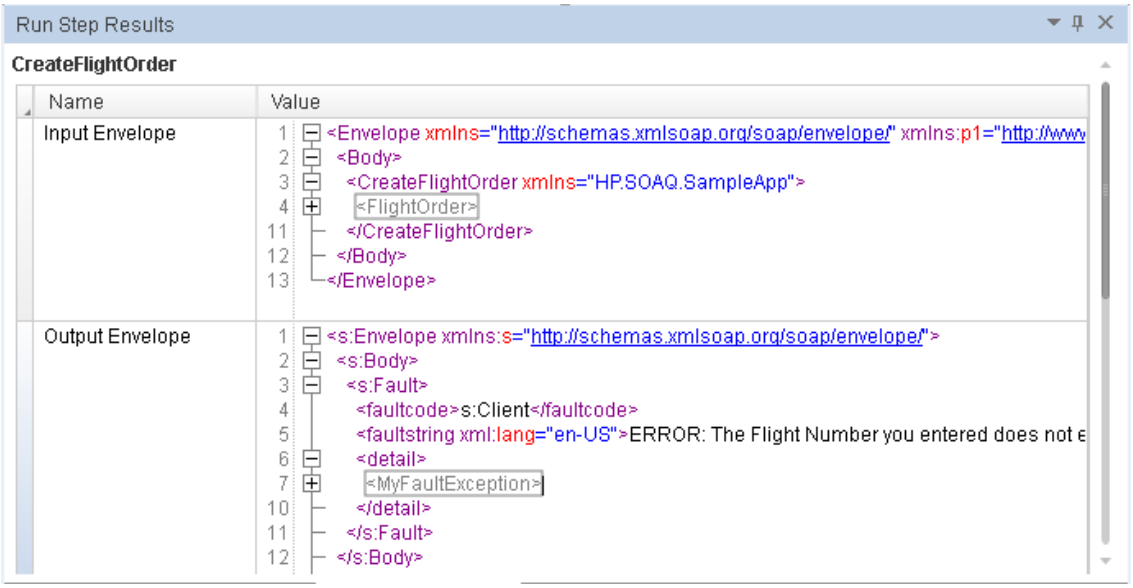
Before you run your test, you can test individual steps to make sure they run properly. This feature, Run Step, is available directly from the canvas and is available for most step types. To run an individual step, select it and choose **Run Step** from its right-click menu.

After you run the step, the Run Step Results pane opens and shows the properties and results.



For services using SOAP, this pane shows the Input and Output envelopes. You can expand and collapse the nodes of the envelope to make it more readable.

This can be especially useful in checking the validity of your input property values. In the following example, the specified flight number was not valid.



The Run Step feature also verifies checkpoints that are selected in the Checkpoints pane, including those of Fixed array type.

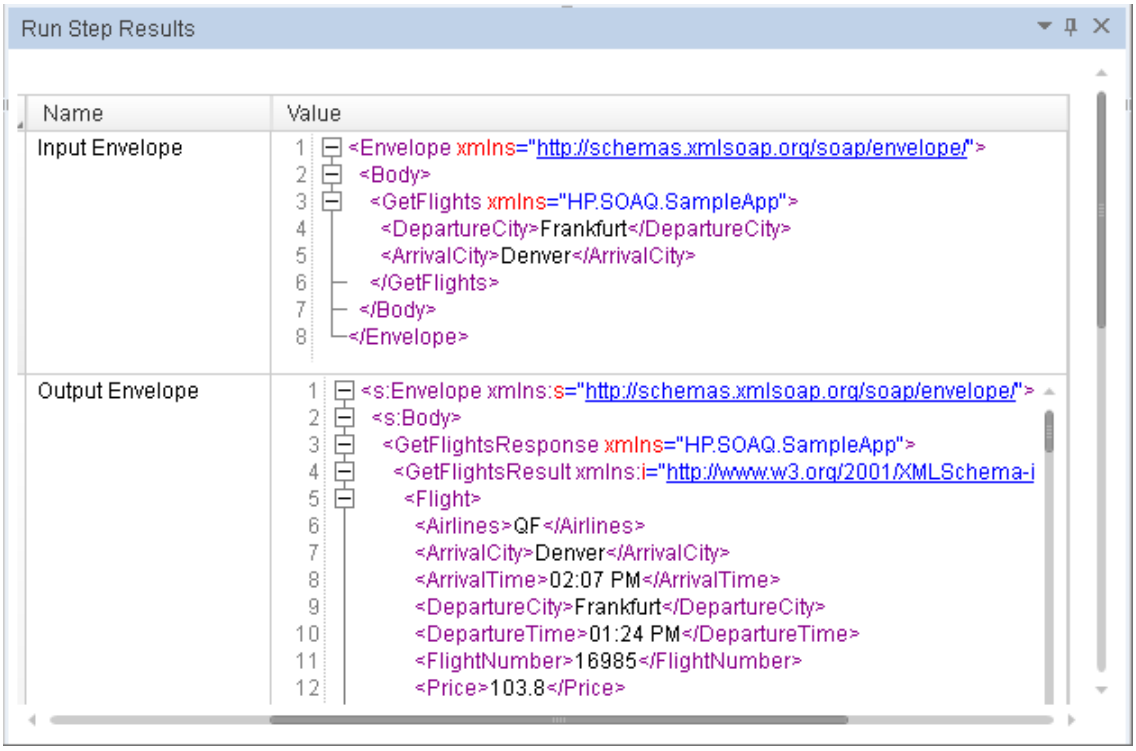
For a user interface description, see "Run Step Results Pane User Interface" on next page.

For REST methods, use the Run Method feature in the REST user interface. For details, see "Add/Edit REST Service Dialog Box" on page 495.

Reference

Run Step Results Pane User Interface

The Run Step Results pane shows the results of the run. For a simple built-in activity, it shows the status of the run. For SOAP messages, it shows the Input and Output envelopes.



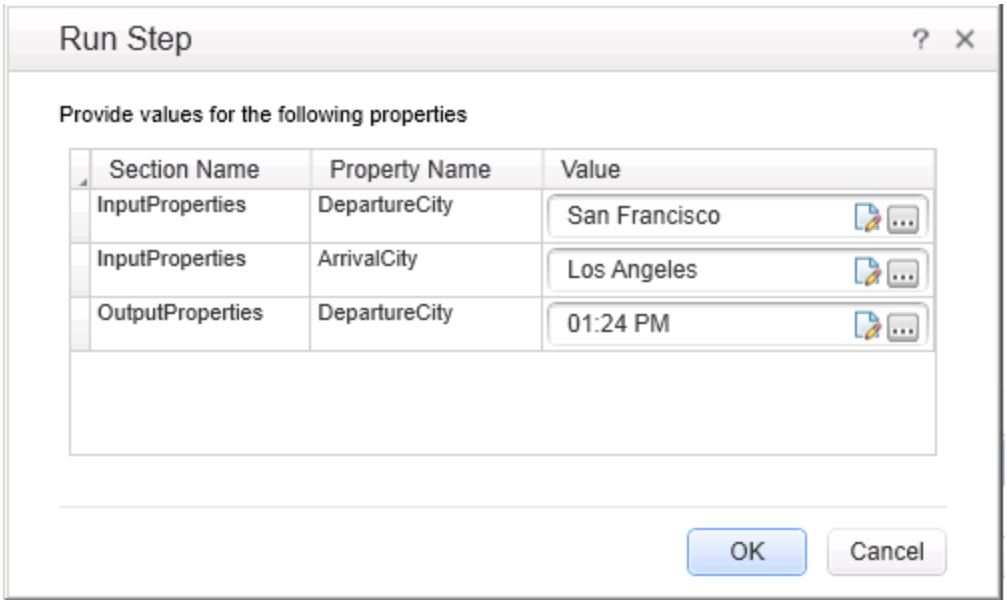
To access	Select **View > Run Step Results**
Relevant tasks	"How to Create an API Test" on page 383
See also	• "The Canvas" on page 373 • "Run Step Dialog Box" on page 224

The user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Step Name> table	A table showing the names and values of properties and an invocation summary. The following is a list of the common fields: • **Name.** The name of the step as it appears in the General properties. • **Comment.** The text in the General properties tab's **Comment** field. • **<Properties>.** The input and output property names and values (non SOAP steps). • **Input Envelope.** A tree hierarchy of the SOAP envelope for the input properties. • **Output Envelope.** A tree hierarchy of the SOAP envelope for the output properties. • **Invocation Results.** Text describing the results of the run—both upon success and failure.
Checkpoints (<success ratio>) table	A table showing the checkpoints, with the following information: • **Property name** • **Expected** and **Actual** values • **Operator (=, <, >, and so forth)** • **XPath of the property**

Run Step Dialog Box

This dialog box enables you to set constant values for the **Run Step** operation.



To access	Do the following: 1. Add a step to the canvas that supports **Run Step.** 2. Link one or more properties to an output property from a previous step. 3. Click the step in the canvas and select **Run Step** from the right-click menu.
Important information	A property is listed when one of the following applies: • The property's input value is linked to the output property of a previous step. • The property requires a value, such as a URL for an HTTP Request step, and that value was not specified in the Properties pane. **Note:** When you assign constant values to the properties, the value is only used for this local **Run Step** step—the link to the output property in the Properties pane is not affected.
Relevant tasks	"How to Create an API Test" on page 383.
See also	"Run Step Results Pane" on page 219

User interface elements are described below:

UI Element (A-Z)	Description
	**Open Text Editor.** Opens an editor for editing text strings. **Note:** For properties with enumeration, this button is not shown. Instead, a drop down list is available for selecting a value.
	**Import from File.** Allows you to import a property value from a file. This is useful for complex properties whose values are stored in XML files. **Note:** For properties with enumeration, this button is not shown. Instead, a drop down list is available for selecting a value.
<property list>	A list of the properties to which a data source was assigned, that need to be given a constant value. It shows the following information: • **Section Name.** The origin or type of property, usually `Input` or `Output`. • **Property Name.** The property name as it appears in the Properties pane. • **Value.** The value to use in the run. By default, it takes the value from the first row of the data source.

Troubleshooting and Limitations - Run Step Results

This section describes troubleshooting and limitations for the Run Step Results pane.

- When working in non-English operating systems, certain entries in the Run Step Results pane are hard-coded in English and not translated.

Chapter 12: Search Results Pane

This chapter includes:

Concepts

Search Results Pane Overview

The Search Results pane displays all occurrences of the search criteria you define using the Find dialog box or other Search menu items. Search results are displayed in one continuous list or grouped according to source file.

You can browse through the search results, locate specific results in their source files, and perform recent searches again to retrieve updated results.

For details on searching for specific strings, see "How to Navigate Through the Search Results Pane" on next page.

For user interface details, see "Search Results Pane User Interface" on page 231.

Note: Searches are not supported in the canvas.

Tasks

How to Navigate Through the Search Results Pane

This task describes how to navigate through results displayed in the Search Results pane, as well as how to access the recent search history.

This task includes the following steps:

- "Prerequisite" below
- "Modify the search results display" below
- "Browse and locate search results" on next page
- "Display updated results for recent searches" on next page
- "Clear the recent search history" on next page

For user interface details, see "Search Results Pane User Interface" on page 231.

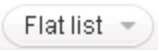
Prerequisite

Do one of the following:

- Perform a search using the **Find All** button in the "Find Dialog Box" (described on page 152). For details, see "How to Find or Replace Strings in Files" on page 141.
- Search for references using the Search menu options. For details, see "How to Search for References or Classes in Tests in the Editor" on page 140.

Modify the search results display

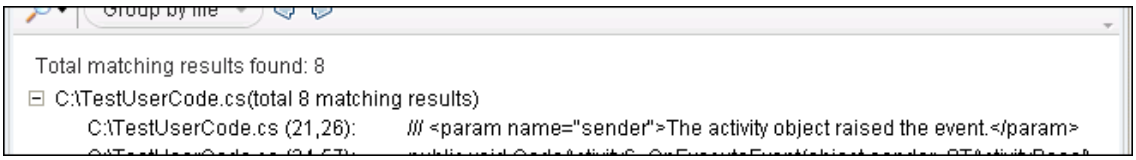
To toggle between display modes:

Click the **Select search list mode** 

button to toggle between the following display modes:

- **Flat list.** Displays all search results in a single list.
- **Grouped per file.** Displays all search results in a hierarchical list according to their source files.

When the results are grouped per file, a file information line is displayed above each group of results, indicating the source file and number of occurrences found in the file. For example:



To collapse or expand the results per file:

Click the **Collapse per file** or **Expand per file** ⊞ ⊟ icon to the left of the file information line to view the results found in a specific file.

Browse and locate search results

- Use the scroll bar to move up or down in the search results list.
- Do one of the following to jump to the location of a search result in the document pane and highlight the search string:
  - Double-click a search result in the list.
  - Click **Previous** to select the previous item in the search results list.
  - Click **Next** to select the next item in the search results list.

Display updated results for recent searches

Click the **Show last searches** button and select a search string from the drop-down list.

The **Show last searches** drop-down list displays the last ten search strings used.

Clear the recent search history

Click the **Show last searches** button and select **Clear history.**

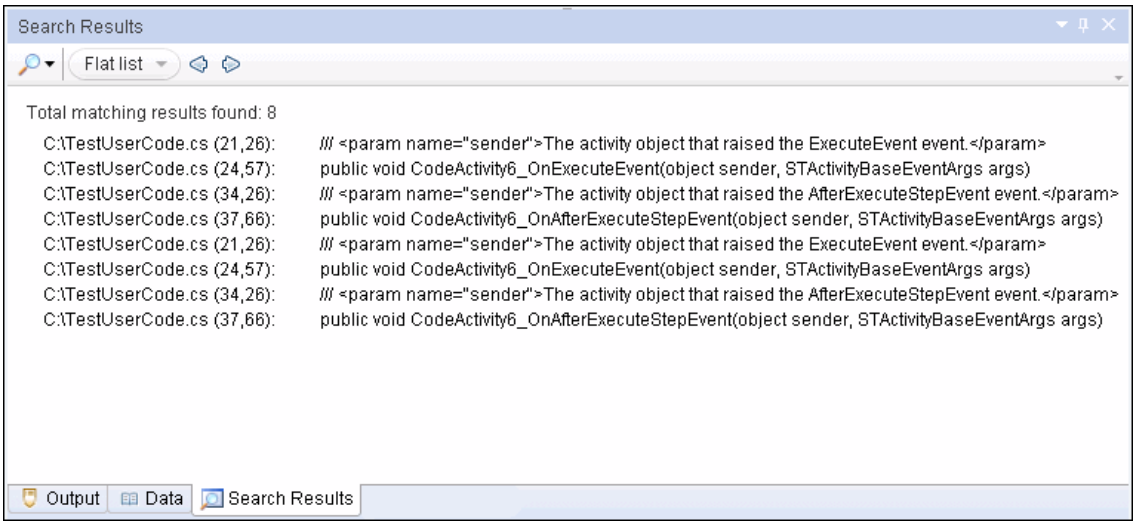
The **Show last searches** drop-down list is cleared of all recent search strings.

Reference

Search Results Pane User Interface

This pane displays all occurrences of the search criteria you defined and enables you to:

- Locate specific search results in their source files
- Retrieve updated search results for recent searches
- Clear the search history



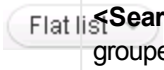
To access	Do one of the following**:** • Select **View > Search Results** to view the results of your last search. • In the **Find** dialog box, define the search criteria and click **Find All**. • Perform a search for references using the Search menu options.
Important information	Click on a row or use the **Previous** and **Next** buttons to jump to the location of the search result in the document pane and highlight the search string. **Note:** Searches are not supported in the canvas.
Relevant tasks	• "How to Navigate Through the Search Results Pane" on page 229 • "How to Find or Replace Strings in Files" on page 141 • "How to Search for References or Classes in Tests in the Editor" on page 140
See also	"File and Item Types Included in String Searches" on page 136

The following sections describe:

- "Main User Interface Elements" below
- "Context Menu User Interface Elements" on next page

Main User Interface Elements

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

UI Element	Description
	**<Show recent searches>.** Enables you to: • View a drop-down list displaying the last ten **Find All** search strings used. • Clear the search history entirely. Select a search string to retrieve the most recent results, or select **Clear search history** to clear the recent search history.
	**<Search list display mode>.** Enables you to display all search results in a single list or grouped according to source file.
	**Previous result.** Selects the previous item in the search results list, jumps to the location of the search result in the document pane, and highlights the search string.
	**Next result.** Selects the next item in the search results list, jumps to the location of the search result in the document pane, and highlights the search string.
	<Collapse or expand per file>. Enables you to collapse or expand the list of search results located in a specific file, when viewing an expanded list of results grouped by file. **Note:** This icon is displayed when the results are grouped by file.

UI Element	Description

<Search results list>	Search results are displayed with the following syntax, depending on the type of source file:
	Test flow
	• **Grouped per file:**
	`<Activity StepID>/<Tab Name>/<Property Name> : <Property Value>`
	• **Viewed in flat list:**
	▪ `<Test Name>/<Activity StepID>/<Tab Name>/<Property Name> : <Property Value>`
	or
	▪ `<Test Name><(Action)>/<Activity StepID>/<Tab Name>/<Property Name> : <Property Value>`
	User source code
	• **Grouped per file:**
	`<line>,<column> : <text>`
	• **Viewed in flat list:**
	`<File Path> (<line>,<column>): < text >`

Context Menu User Interface Elements

The user interface elements described below are available when you right-click a search result in the Search Results pane:

UI Elements	Description
Copy	Copies the content of the selected search result to the clipboard.
Locate	Jumps to the location of the selected search result in the document pane and highlights the search string. **Note:** You can also double-click the search result.

Chapter 13: Solution Explorer Pane

This chapter includes:

Concepts

Solutions in Service Test Overview

A **solution** is a collection of tests and other resources, similar to a binder or notebook. You can use solutions to organize your tests to help you perform comprehensive application testing.

For example, suppose you want to test a Web application for a flight booking service. You can create a solution containing several tests that verify various aspects of your service, such as logging in, booking a reservation, verifying the connection between your application and database, and verifying the transfer of booking information from your application to an airline server.

In Service Test, all tests must be part of a solution. Therefore, when you create a test, you assign it immediately to a solution. You can provide descriptive names for your solutions, or you can accept the default names provided by Service Test.

Service Test solutions can include tests, business components, business process tests and flows, or user-created code documents. When you add a test to a solution, it retains its unique associations to other tests and resource files and does not create links to the other tests in that solution. The relationship between solutions and tests goes in one direction. The solution contains references to its tests, but the tests do not contain a reference back to any solutions that contain it.

You can create solutions for own use, or you can share solutions with other users by saving them in an accessible location, such as a network drive. Although your tests can be stored in the file system or ALM, solutions are always stored somewhere in the file system.

You view and manage solutions in the Solution Explorer pane.

Solution Explorer Pane Overview

You use the **Solution Explorer** pane to view and manage the tests in a solution. The Solution Explorer is displayed as a tree in the Solution Explorer pane. This tree displays a separate node for each test, with sub-nodes for all referenced items.

From the Solution Explorer, you can:

- Add new or existing tests to a solution.
- Open any test directly from the Solution Explorer. You can open several tests side-by-side in the same session.
- Display the properties and settings for the solution or any of its tests by highlighting the relevant node. This refreshes most of the panes in Service Test as the Solution Explorer pane serves as the master pane in Service Test.
- Add user-created functions to a test by adding a C# user code file.
- Add an external reference file toa test for a run session or add a Web reference for use in a test.
- Rename the solution and many of its sub-nodes.
- Run a test or component from the solution.

Note: When you open a test from outside of the Solution Explorer (from the File menu, for example), the test always opens in a new, untitled solution, even if it is part of the currently open solution. Therefore, if you want to open a test that is part of the current solution, and you do not want to close any other open tests, open the test from the Solution Explorer.

For an overview of solutions, see "Solutions in Service Test Overview" on previous page.

For task details, see "How to Manage Items in the Solution Explorer Pane" on next page.

For user interface details, see "Solution Explorer Pane User Interface" on page 240.

Tasks

How to Manage Items in the Solution Explorer Pane

This task describes how to perform various operations in the Solution Explorer pane.

This task includes the following steps:

Prerequisite: Show the Solution Explorer pane

Do one of the following:

- Select **View > Solution Explorer**.
- Click the **Solution Explorer** button  in the toolbar.
- Click the **Solution Explorer** tab in the main Service Test window.

Create a solution

Do one of the following:

- Select **File > New > Solution**.
- Click the **New** button down arrow  in the toolbar and select **New Solution**.
- Select **File > New > Test**.

In the "New <Document> Dialog Box" (described on page 67), enter a solution name in the **Solution Name** field and select the **Create directory for solution** option.

A new solution, also containing the test selected, opens in the Solution Explorer pane.

Note: If you do not create a named solution, a generic solution called `Solution Untitled`

is created.

Open a solution

Select **File > Open > Solution**. In the dialog box, select the name of your solution file.

If a test included in a solution is unavailable, due to issues with ALM connectivity or resource file problems, it appears in gray and an error describing the problem is displayed in the Errors pane. Right-click **Reload** after resolving the error to add the test to the solution for editing.

Note:

- If you open a solution when another solution is open, the first solution is closed and you are prompted to save any unsaved changes to tests contained in the closed solution.
- You can save and open solutions only on the file system.

Create a test

Right-click the **<solution name>** node and select **Add > Add New Test/Business Component**.

The "Add Test/Component to Solution Dialog Box"(described on page 64) opens, enabling you to create a new test and add it into your solution file.

Add a test to the current solution

Right-click the **<solution name>** node and select **Add > Add Existing Test/Business Component**.

The "Add <Existing Document> to Solution Dialog Box" (described on 63) opens, enabling you to add a test to your solution file.

Note: You can add a maximum of 10 tests to a solution.

Open an existing test from the current solution

Double-click a node for your test:

- **For tests and components:** the **Flow** node located under the test or component node.
- **Forevent handlers:** the **Events** node located under the test or component name.
- **For user code files:** the <**User Code file name>** node located under the test or component name.

A tab opens in the document pane with the selected document.

Run a test or component from the current solution

Right-click on a test or component node and select **Run**. The Run dialog opens, enabling you to choose preferences for your run session. For details, see "Run Dialog Box" on page 296.

View or hide the sub-nodes in the test

Do one of the following:

- Right-click the **Test** node in the tree and select **Expand All** or **Collapse All**.
- Click the expand ⊞ or collapse ⊟ icons next to the node to expand or collapse the node.

Remove a test from the current solution

Do any of the following:

- Right-click a test node and select **Remove from Solution**.
- Select a test and press `DELETE`.

> **Note:** Removing a test from a solution does not delete it from your file system or ALM project.

Save an individual test within the solution

Right-click a test name and select **Save As**.

The "Save <Resource>/Save <Document> As Dialog Box"(described on 72) opens, enabling you to save the test with a different name or in a different location.

Save the solution

Select **File > Save All** or press the **Save All** button  on the toolbar.

> **Note:** Selecting this command also saves any modified test within the solution.

Close the solution

Select **File > Close Solution**.

> **Note:** Closing a solution closes all tests that are part of the solution file. These files are not deleted from your file system or ALM project when a solution is closed.

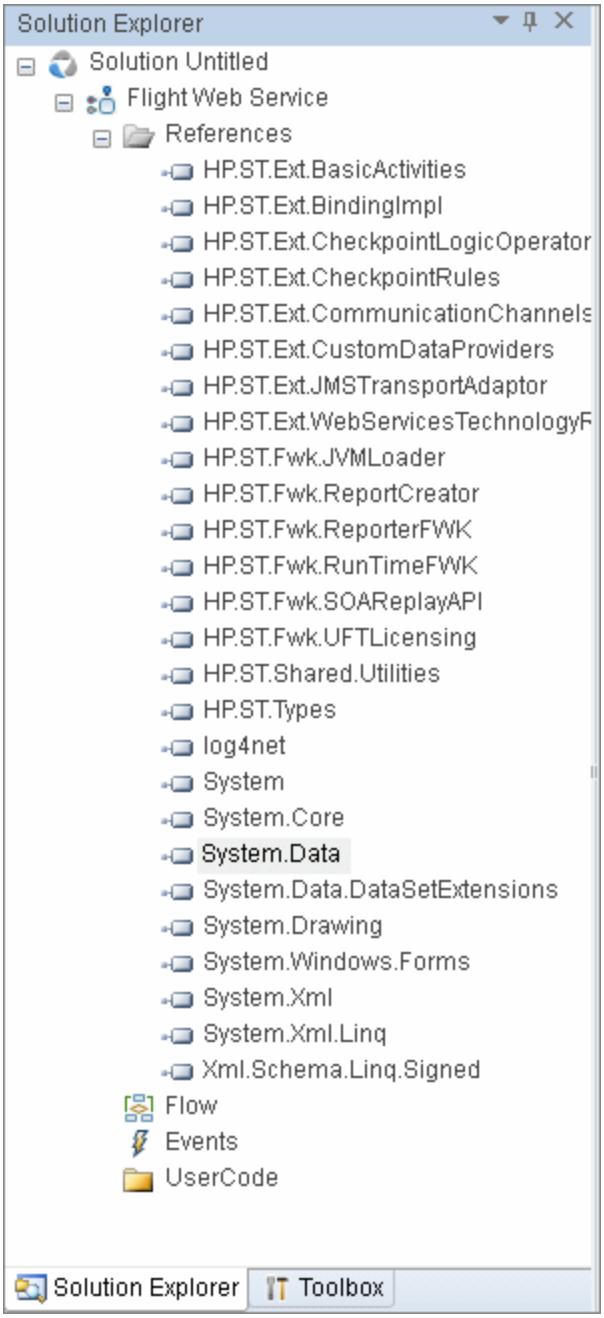
Reference

Solution Explorer Pane User Interface

This pane enables you to:

- View the currently open solution and its tests
- Add or remove items from a solutions
- Open tests contained in a solution
- Run a test or component

The following image shows the Solution Explorer pane displaying the available nodes when tests are part of a solution.



To access	Do one of the following: • Select **View > Solution Explorer** • Click the **Solution Explorer** toolbar button .

Important information	• The Solution Explorer pane is displayed by default when you start Service Test. • The Solution Explorer pane is the control pane within Service Test. Other panes, menus, and toolbars update their content to match the test selected in the Solution Explorer pane. • Test nodes are displayed in alphabetical order. • The active test is shown as bold in the Solution Explorer pane.
Relevant tasks	"How to Manage Items in the Solution Explorer Pane" on page 237
See also	"Solution Explorer Pane Overview" on page 235

The Solution Explorer pane can display the following key nodes:

Note: All user interface descriptions of each node also contain the descriptions of the context menus available for the node.

Solution Node (Solution Explorer Pane)

The **solution node** displays the name of the solution and contains all the tests and resource nodes necessary to edit and run a test. For details about a specific type of test node, see the relevant section on each node below.

The following context menu options are available when you right-click the solution:

- **Add New/Existing <Document>.** Opens the "Add Test/Component to Solution Dialog Box" (described on page 64) or "Add <Existing Document> to Solution Dialog Box" (described on page 63), which adds a new or existing test to a solution.

  Note: You can add a maximum of 10 tests to a solution.

- **Expand All.** Expands all test, component, or application area sub-nodes contained in the solution.
- **Collapse All.** Collapses all sub-nodes contained in the solution.
- **Compile solution (API testing only).** Compiles the solution files that have changed.
- **Recompile solution (API testing only).** Compiles all solution files, regardless of whether they changed. You should run the **Clean solution** command before this command.
- **Clean solution (API testing only).** Removes all files generated in the previous compilation of the solution.

Note: The **Compile**, **Recompile**, and **Clean** commands are applicable if your solution contains only APIService Test tests. This command is not available if a solution contains GUI tests or components.

Service Test Test/Component Node (Solution Explorer Pane)

This **Service Test Test node** is available only if your solution contains aService Test test or component. Some of the nodes contained within this one, are available only if a relevant resource is created in a test or component.

This node can contain the following sub-nodes:

- "References Node" below
- "Flow Node" below
- "Events Node" below
- "Action Node" on next page
- "External Tests/Actions Node" on next page
- "UserCode Node" on next page
- "Context Menu Options" on next page

References Node

This node contains all of the `.dll` files used by Service Test to run tests.

Note: Do not remove any of the reference files contained by default in this node. They are required to perform aService Test run session. You can add additional user-created reference files using the context menu options. For details, see "Add Reference Dialog Box" on page 245.

Flow Node

This node provides a link to the Service Test canvas, which displays the test's flow. Double-clicking this node displays the canvas or opens the canvas's tab (if closed). Each individual action also contains a flow node for the action's steps.

For details on using the Service Test canvas, see "The Canvas" on page 373.

Events Node

This node provides a link to the test or action's `TestUserCode.cs` file used for coding event handlers. Double-click on this node to display the file. Each individual action within a test also has an events node for the action.

Note: If you double-click on this link independently of the events tab in the Properties pane, the global test or component user code is displayed, including all event handlers currently written into the test.

For details on coding Service Test events, see "Writing Code for Events Overview" on page 642.

Action Node

This node contains all of the actions created in aService Test test.

Each action node contains a **Flow** and **Events** node.

Note: This node is displayed only when you create an action in aService Test test. For details, see "Actions Overview" on page 508.

External Tests/Actions Node

This node contains all of the tests or actions (both UFT and Service Test) called from the Service Test test. The different types of called actions are noted by their icons.

Note: This node is only displayed when you create a call to an external test or action. For details on using HP Functional Testing activities in aService Test test, see "HP Automated Testing Tools Activities" on page 449.

UserCode Node

This node contains any user code files associated with the test. The following user interface elements are available:

UI Element	Description
<file name>	The name of the user code file name. **Note:** Only `.cs` files can be associated with aService Test test.
<folder name>	The name of a folder containing user-generated code to add to a test.

Context Menu Options

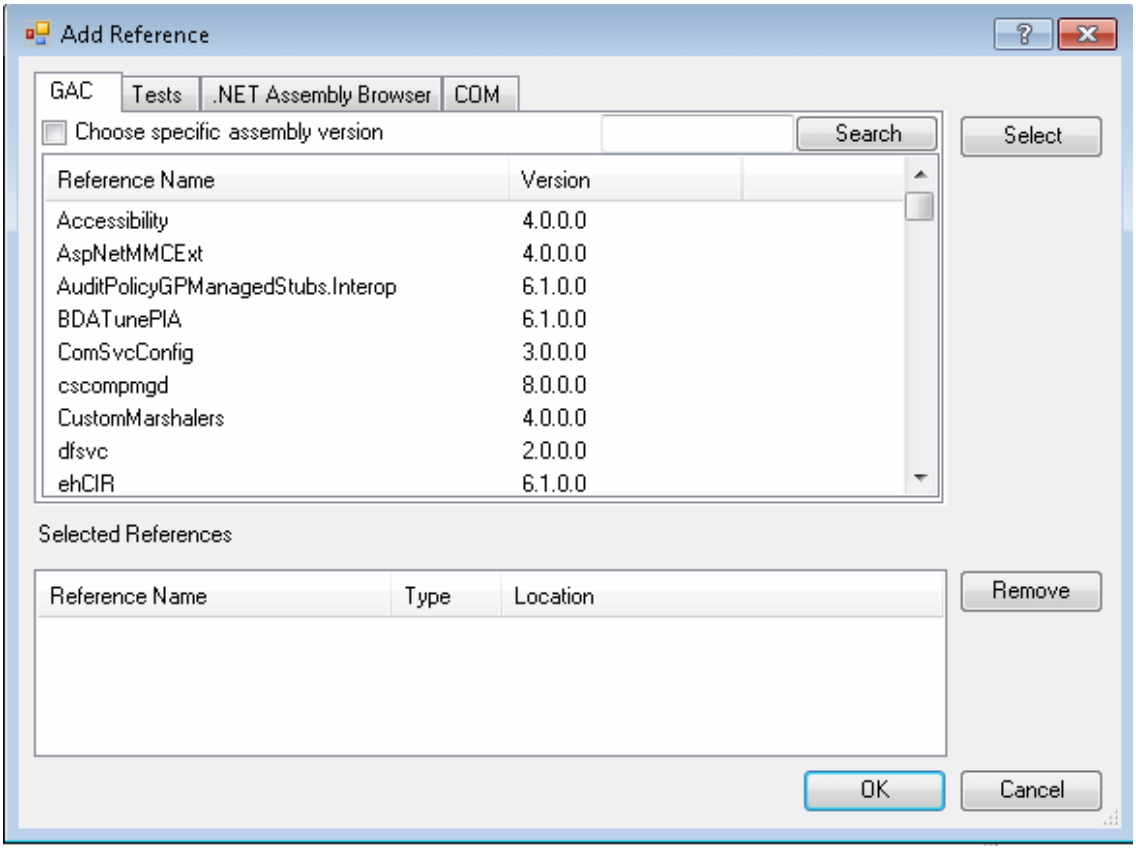
The following context menu options are available for the **Service Test** node:

Node	Context Menu Option
<test name>	• **Run.** Opens the "Run Dialog Box" (described on page 296) to begin a run session for the current test. • **Expand All.** Expands all sub-nodes found under the test. • **Collapse All.** Collapses all sub-nodes found under the test. • **Remove from Solution.** Removes the current test from the solution file. • **Save As.** Opens the "Save <Resource>/Save <Document> As Dialog Box" (described on page 72), enabling you to save the selected test under a different name or in a different location.
References folder	• **Add Reference.** Opens the "Add Reference Dialog Box" (described on page 245), which adds a external reference file to your test.

Node	Context Menu Option
<reference name>	• **Refresh.** Updates the selected reference file with any changes made to the file. • **Remove.** Removes the current reference from the list. • **Properties**. Displays the properties of the reference file in the Properties pane. **Caution:** Do **not** remove the reference files included in this node, as doing so may cause a run session to fail.
<action name>	• **Delete Action.** Removes the selected action from the test. • **Rename Action.** Opens the "Rename Action Dialog Box" on page 517, enabling you to rename the action. **Note:** Before renaming an action, you must close the action's tab and user code file and save your test.
User Code	• **Add New File.** Opens a dialog box to add a new custom code file to your test. For user interface details, see "Open/New <Document>/<Resource> Dialog Box" on page 69. • **Add Existing File.** Opens the "Add <Existing Document> to Solution Dialog Box" (described on page 63), which adds an existing custom code file to the test.
<user code file/folder name>	• **Open.** Opens the file in the document pane or brings it into focus. • **Open Containing Folder in Explorer.** Opens the folder containing the user code file in Windows Explorer. • **Remove.** Removes the user code file from the user code node and deletes it. • **Exclude From Test.** Removes the association of this user code file from your test. • **Rename.** Renames the selected file. • **Properties.** Opens the file's properties in the Properties pane.

Add Reference Dialog Box

This dialog box enables you to add a reference file to your Service Test test.



To access	Do one of the following: • Select the **References** node of a test, right-click and select **Add Reference**. • Select **Design > Add Reference.**

User interface elements are described below:

UI Element	Description
GAC tab	Displays a list of the GAC (Global Assembly Cache) assemblies and version numbers. **Note:** • If you select the **Choose specific assembly version** option, the pane displays a list of all versions of all assemblies. You can then select a specific version of an assembly to insert in the test. • If you enter a text string into the **Search** field, Service Test searches through the list of assemblies for the assembly that matches your search string.
Tests tab	A list of all tests contained in the current solution.

UI Element	Description
.NET Assembly Browser tab	Enables you to searches the file system for .NET assemblies to import into a test. The browser displays the following information: • **<directory location>.** The location on the file system for a folder or assembly. • **Back/Up** buttons. Enable you to return to the previously selected directory or the previous level on the directory hierarchy to search for an assembly. • **<files list>.** The list of all available files in a directory location.
COM tab	A list of all software components and their location in the file system. **Note:** If a path to a component is not listed in the pane, then the component is located in the `<Service Testinstallation>/bin` folder.
Select	Adds the current assembly, project, or software component to the **Selected References** list.
Remove	Removes the current assembly, project, or software component from the **Selected References** list.
Selected References	A list of all assemblies, projects, or software components to add to your test. The grid displays the following information: • **Reference Name.** The reference name as displayed in the References node in the Solution Explorer pane. • **Type.** The type of reference for the selected reference, project, or assembly. • **Location.** The location of the selected reference in the file system. If a location is not displayed for the reference, it is located in the< **Service Test installation>/bin** folder.

Business Process Test/Flow Node (Solution Explorer Pane)

The Solution Explorer pane does not display sub-nodes for business process tests or flows. You can view the components in the document pane.

The following context menu options are available for a business process test or flow:

- **Run.** Runs the current business process test or flow.
- **Remove from Solution.** Removes the current business process test or flow from the solution.

For details on business process testing, see the *HP Business Process Testing User Guide*.

Troubleshooting and Limitations - Solution Explorer Pane

This section describes troubleshooting and limitations for the Solution Explorer pane.

Solutions stored on a network location are not locked when opened by Service Test.

Chapter 14: Tasks Pane

This chapter includes:

Concepts

Tasks Pane Overview

The Tasks pane enables you to create and manage TODO comments for issues that need to be handled in your user code files.

TODO comments are reminders that are inserted as comments adjacent to the relevant steps in your testing document. For example, you can provide instructions to someone else during a handover, or you can remind yourself to do something.

When you create event handlers, the editor automatically inserts TODO text, indicating where you need to enter your code.

You can access TODO comments from the Tasks pane or directly from the testing document. For details, see "Tasks Pane User Interface" on next page.

If needed, you can export your TODO comments to an XML file or Microsoft Excel.

Tasks

How to Create and Manage TODO Comments

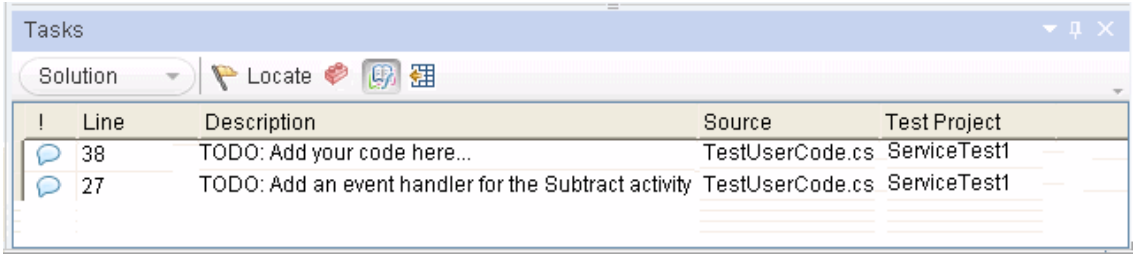
This task describes the different operations you can perform to manage TODO comments in user code files.

- To **add** a new TODO comment, insert a comment adjacent to the relevant step in your document. A comment can begin with any of the following permutations of the words **to do**: `To Do`, `todo`, `to-do`, or `TODO`.
- To **delete** a TODO comment, you must delete the source line in the document. This removes the TODO comment from the pane. (To jump to the source line, you can either double-click a comment in the Tasks pane, or you can highlight the comment in the Tasks pane and then click the **Locate** button.)
- To **sort** by a specific column, click on the column header.
- To **rearrange columns**, drag a column header to a different location.
- To **filter** TODO comments, click the **Show Tasks from** button and select one of the following:
  - **Solution.** Displays all of the TODO comments stored in all of the tests included in the solution (including closed tests).
  - **A specific test or component.** Displays only the TODO comments that are stored in that test or component.
- To **export** TODO comments:
  a. Click the **Export Task List**  button.
  b. In the "Save <Resource>/Save <Document> As Dialog Box" on page 72 (described on page 72), browse to the required location in the file system.
  c. Enter a file name and specify the file type. You can export the file as XML, CSV, or `.xls`/`.xlsx` (if Excel is installed on the computer).
  d. Click **Save**. The TODO comments are saved to a file in the specified location and in the specified format.

Reference

Tasks Pane User Interface

This pane enables you to view and access TODO comments in user code files.



To access	In the main Service Test window, select **View > Tasks**.
Important information	• This view can display any comment step that begins with any of the following permutations of the words **to do**: `To Do`, `todo`, `to-do`, or `TODO` (not case-sensitive). **Example:** `To Do need to ask Sarah to add design steps` • The text displayed in the Comments view is limited to 260 characters. If the text exceeds this limit, and you want to view the entire comment, you can jump to the comment in the testing document by double-clicking the comment line in the Comments view.
Relevant tasks	"How to Create and Manage TODO Comments" on previous page
See also	"Tasks Pane Overview" on page 250

User interface elements are described below:

Toolbar Buttons

Toolbar Option	Description
<Filter>	Enables you to show all of the TODO comments for the solution, or to filter the display to show TODO comments only for a specific test or component.
Locate	Jumps to the comment line in the user code file for the currently selected TODO comment. **Tip:** You can also double-click a TODO comment to jump to its location in the source document. **Note:** This option is available only when a line in the Tasks pane is selected.

Toolbar Option	Description
Export Task List	Saves the TODO comments to an external file, such as a text file. You can save the list of TODO comments in any of the following formats: • XML (Extensible Markup Language) • XLS/XLSX (Microsoft Excel file) • CSV (Comma-Separated Values file)

Columns

Column	Description
Line	The line number of the TODO comment in the source document.
Description	The text of the TODO comment.
Source	The file name of the user code file containing the TODO comment.
Test	The name of the test containing the TODO comment, or associated with it.

Context Menu Options

Context Menu Option	Description
Copy	Copies the TODO comment to the Clipboard.
Locate	Jumps to the comment line in the user code file for the currently selected TODO comment. **Tip:** You can also double-click a TODO comment to jump to its location in the source document.

Chapter 15: Toolbox Pane

This chapter includes:

Concepts

Toolbox Pane Overview

The Service Test panes allow you to create tests using a drag and drop mechanism.

The **Toolbox** pane provides a collection of standard service activities for functional testing in areas such as XML, SOAP, Java, JMS, IBM MQWebsphere and HTTP. You can add more activities to the Toolbox pane by importing WSDLs or using services from the repository.

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

You can also create new custom activities, using the built-in Activity Wizard. For details, see "Extensibility in Service Test" on page 661.

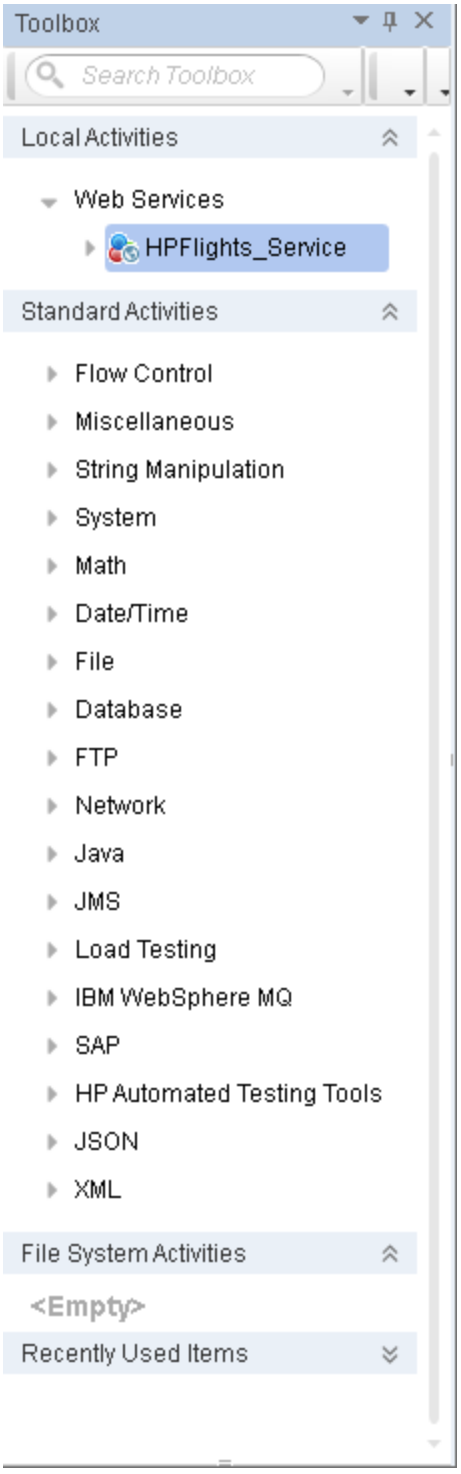
References

Toolbox Pane User Interface

The Toolbox pane 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 pane is Service Test's main window for creating and populating tests.

View Image


Toolbox
Search Toolbox
Local Activities
Web Services
HPFlights_Service
Standard Activities
Flow Control
Miscellaneous
String Manipulation
System
Math
Date/Time
File
Database
FTP
Network
Java
JMS
Load Testing
IBM WebSphere MQ
SAP
HP Automated Testing Tools
JSON
XML
File System Activities
<Empty>
Recently Used Items

To access	1. Create or open a test or component. 2. Do one of the following: ▪ Click the **Toolbox** tab in the bottom of the left pane. ▪ Select **View > Toolbox** or click the Toolbox tab in the main Service Test window.
Important information	• You double-click or drag activities from this pane onto the canvas. • The activity becomes as test step when it appears in the canvas. • Click on the down arrows at the top of the pane to access the toolbar controls.
Relevant tasks	"How to Create an API Test" on page 383

Toolbar Controls

The following toolbar controls elements are included in the Toolbox pane. To see all of the controls, expand the pane or use the drop down menus.

Tree Elements	Description
	**Expand All.** Expands all nodes in the Toolbox pane.
	**Collapse All.** Collapses all nodes in the Toolbox pane.
	**Security Settings.** Opens the "Security Settings for Port <Port_Name> Dialog Box" (described on 559). These settings apply to all operations in the port. **Note:** Only available when selecting the Port node of a Web service.
	**Choose Activities.** Opens the Choose Toolbox Activities dialog box, for selecting activities stored in the File System or ALM repository to add to the Toolbox pane. This is useful for retrieving activities that you removed from the Toolbox using the **Remove Activities** command.
	**Remove Activities.** Opens the Remove Toolbox Activities dialog box for selecting the activities to remove from the Toolbox pane. **Note:** • Removing an activity stored in the repository, only removes it from the Toolbox pane. It remains, however, in the repository, for future use. • This command does not apply to **Standard Activities**.

Tree Elements	Description
	**Update WSDL.** Reimports the WSDL from its original location. If the service's operations changed, this will be reflected in its node in the Toolbox pane. **Note:** Only available when selecting the parent node of a Web service.
Update WSDL from	Opens the Update WSDL From dialog box, allowing you to update the WSDL from any location. **Note:** Only available when selecting the parent node of a Web service.
	**Refresh.** Refreshes the activities in their stored location. This is useful when the WSDL is stored on a shared location and may have been modified by another user. You can refresh the activities instead of re-importing the service. **Note:** Only available when selecting the parent node of a Web service.
	**Delete.** Deletes the selected entry from the Toolbox pane. This is not available for built-in activities.
	**Validate WSI-Compliance.** Runs the WSI validator. The Output pane shows a summary of the validation and provides a path to the report. **Note:** Only available when selecting the parent node of a Web service.
	**Edit Service.** Opens the "Add/Edit REST Service Dialog Box" (described on 495). **Note:** Only available when selecting the parent node of a REST service.
Move to	**Move to.** An expandable menu that enables you to move an item to the file system or ALM repository. For details, see "Activity Sharing" on page 466. **Note:** Only available when selecting the parent node of an item in the **Local Activities** section.
Filter box	Filters the Toolbox pane display by the entered text.

Toolbox Activities

The Toolbox pane contains the following sections:

Activity Category	Description
Local Activities	A tree hierarchy of all imported or items created by the user: **Web Services**, **REST Services**, and **.NET Assemblies**. For details, see "Local Activities" on page 463.
Standard Activities	The built-in activities, by category. For details, see "Standard Activities" on page 396.
File System Activities	Local activities that were moved to the file system repository using the **Move to** option from the right-click menu. For details, see "Activity Sharing" on page 466.
ALM Activities	Local activities that were moved to the ALM repository using the **Move to** option from the right-click menu. For details, see "Activity Sharing" on page 466.
Recently Used Items	The activities or operations that were used most recently.

Part 3: Service Test Configuration

Chapter 16: Service Test Global Options

This chapter includes:

Concepts

Global Options - Overview

The Options dialog box enables you to modify the general appearance and behavior of Service Test. For example, you can define the user interface language, set startup options, or modify the font and colors of code elements in the Editor. The values you set remain in effect for all documents and for subsequent testing sessions.

> **Note:** The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.

For user interface details, see:

- "General Tab (Options Dialog Box)" on next page
- "API Testing Tab (Options Dialog Box)" on page 271
- "Coding Tab (Options Dialog Box)" on page 277
- "Text Editor Tab (Options Dialog Box)" on page 279

Reference

General Tab (Options Dialog Box)

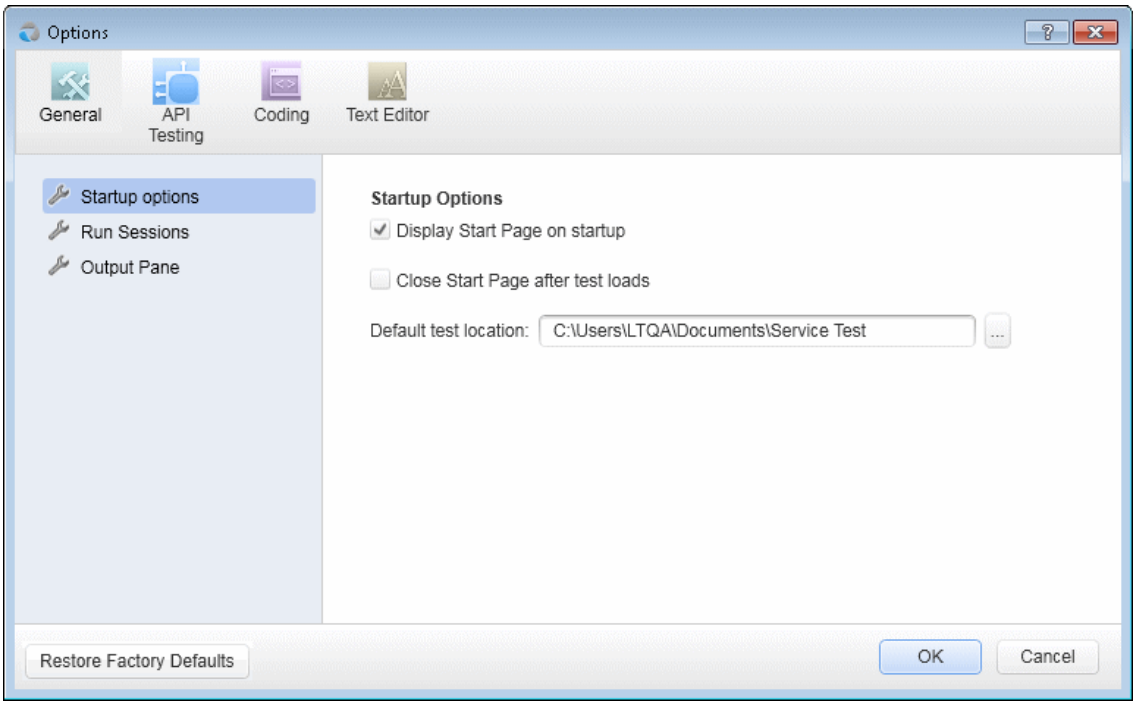
This tab enables you to define general options for Service Test.

To access	Select **Tools > Options > General** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "API Testing Tab (Options Dialog Box)" (described on page 271 • "Coding Tab (Options Dialog Box)" (described on 277) • "Text Editor Tab (Options Dialog Box)" (described on 279)

This tab includes the following panes:

Startup Options (Options Dialog Box > General Tab)

This pane enables you to select what to show when Service Test opens.



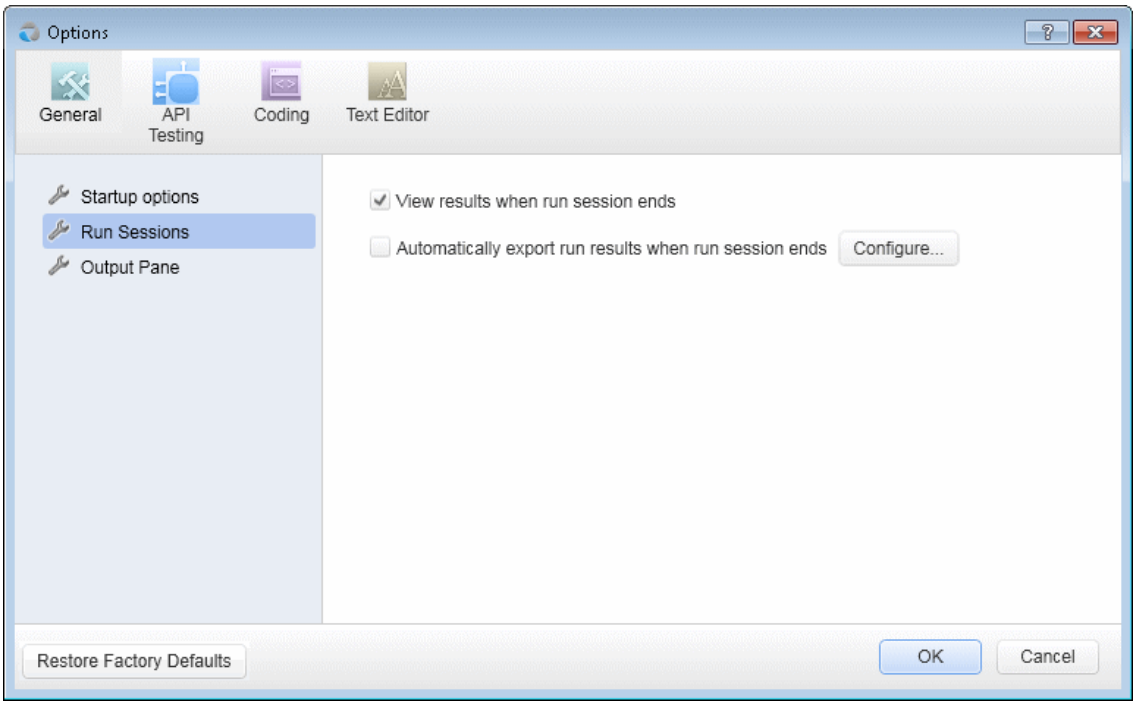
To access	Select **Tools > Options > General** tab > **Startup Options** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
Relevant tasks	"How to Start Service Test" on page 48
See also	• "Start Page" on page 51

User interface elements are described below:

UI Element	Description
Display Start Page on startup	Displays the Start Page when Service Test opens.
Close Start Page after test loads	Instructs Service Test to close the Start Page after a test or business component is created or opened.
Default test location	The default place to save a new test. **Default:** `C:\My Documents\Service Test` **Note:** Business components are always saved in the Business Components module in ALM.

Run Sessions Pane (Options Dialog Box > General Tab)

This pane enables you to determine global Service Test run settings.



To access	Select **Tools > Options > General** tab > **Run Sessions** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
Relevant tasks	See the following tasks in the *HP Run Results Viewer User Guide*: • How to Open Run Results • How to Export Run Results)
See also	The following topics in the *HP Run Results Viewer User Guide*: • The Run Results Viewer overview • The section on the Run Results Viewer User Interface

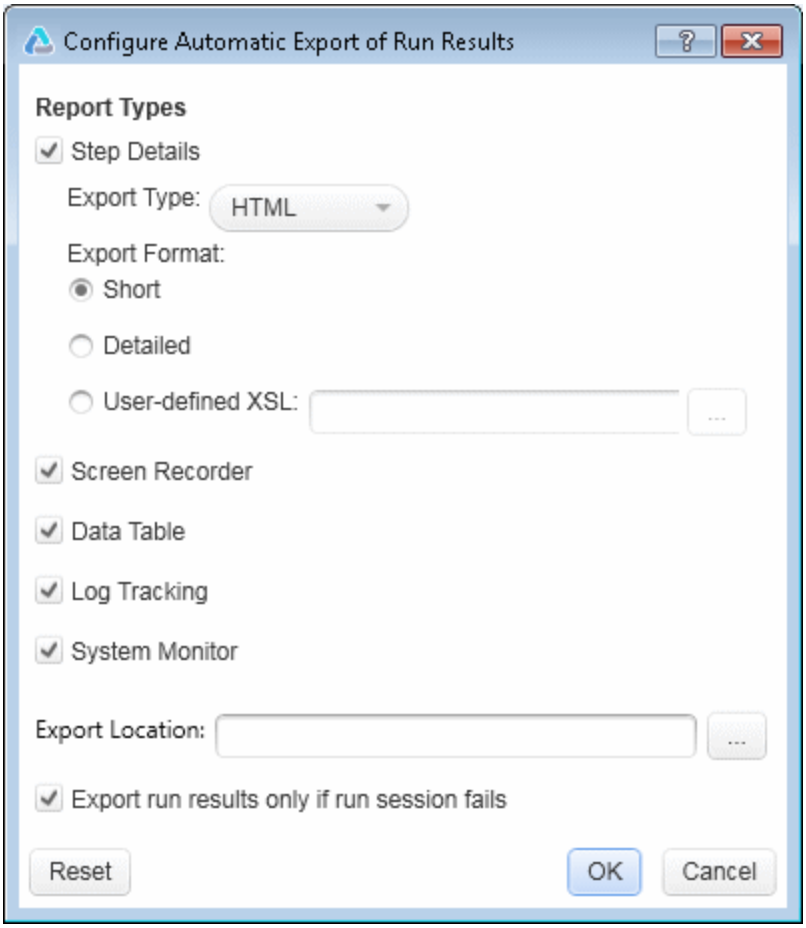
User interface elements are described below:

UI Element	Description
View results when run session ends	Instructs Service Test to open the Run Results Viewer and display the results automatically following the run session.
Automatically export run results when run session ends	Instructs Service Test to export the run results to the location you specify in the "Configure Automatic Export of Run Results Dialog Box" (described on 267). **Note:** If this option is selected, and the Run Results Viewer is not installed on the computer running the test or component, then the results are not exported and an error message is displayed at the end of the run session.
Configure	Opens the "Configure Automatic Export of Run Results Dialog Box" (described on 267), in which you can define settings that instruct Service Test to automatically export run results to a specific folder in the file system after a run session. These settings specify the type of report, the export location in the file system, and whether to export all run results or only results for failed run sessions.

UI Element	Description
Stop command shortcut key	Enables you to define a shortcut key or key combination that stops the current record (for GUI tests only) or run operation, even if Service Test is not in focus or is in hidden mode. Click in the field and then press the required key or key combination on the keyboard. The default key combination is **CTRL+ALT+F5**. **Note:** It is important to define a shortcut that is not already defined for some other operation by application being tested. If this is the case and: • you open the application manually before you click **Record** or **Run**, the shortcut defined in the application is applied for its original purpose. • you start a record or run session and Service Test opens the application for you, the shortcut you define stops the session.

Configure Automatic Export of Run Results Dialog Box

This dialog box enables you to define settings that instruct Service Test to automatically export run results to a specific folder in the file system after a run session.



To access	1. Select **Tools > Options > General** tab > **Run Sessions** node to open the Run Sessions pane. 2. Select **Automatically export run results when run session ends**. 3. Click the **Configure** button.
Important information	• **Manually exporting run results.** You can also export run results manually after a particular run session. For details, see the section on how to export run results (described in the *HP Run Results Viewer User Guide*). • **Note:** Not all options are relevant for Service Test tests. • The settings from this dialog box are saved in a configuration file, `ReportExportOption.xml`.

User interface elements are described below:

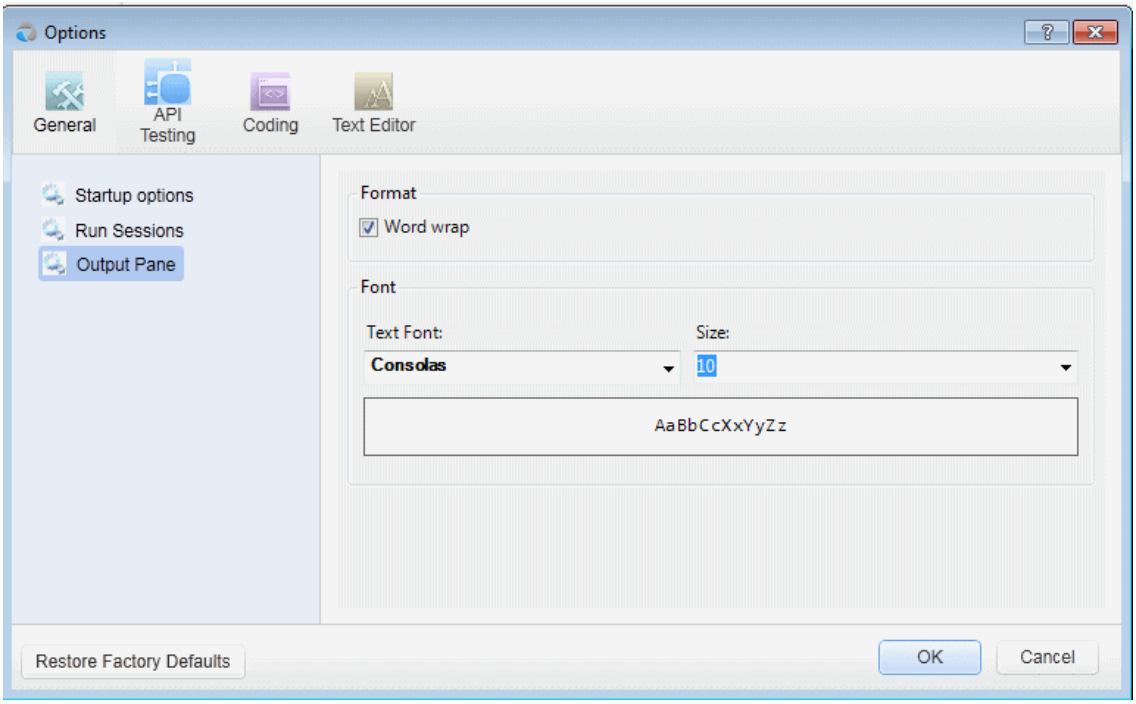
UI Element	Description
Step Details	The main report type. This includes the entire run results tree. (Default) • **Export type.** The file type for the exported results. Possible options: ▪ **HTML** (Default) ▪ **PDF** • **Export format.** The format and amount of detail to include in the exported results. Possible options: ▪ **Short.** Exports a summary line (when available) for each item in the run results tree. The short report does not include still images associated with the steps in your run results. (Default) ▪ **Detailed.** Exports all available information for each item in the run results tree. The detailed report includes still images associated with the steps in your run results. (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 details, see the section on the Run Results XML File (described in the *HP Run Results Viewer User Guide*).
Screen Recorder	Not relevant for Service Test.
Data Table	The runtime version of the data table associated with your ALM configuration. It displays the values used to run a test or configuration that contains Data Table parameters, as well as any output values retrieved from a test or an ALM configuration during a run session. For details on the Data pane, see "Data Pane" on page 96.
Log Tracking	Not relevant for Service Test.
System Monitor	Not relevant for Service Test.
Export location	The folder in which to store the exported files. Clicking the browse button opens the Browse For Folder dialog box, enabling you to select a folder on the file system. If you do not specify a folder, Service Test automatically exports the files to the default folder: `<test folder>\<result folder>\Report`. (For example, `%ProgramFiles%\HP\Service Test\Tests\MyTest\Res9\Report`.)

UI Element	Description
Export run results only if the run session fails	• Select this check box to export run results only for failed run sessions. • Clear this check box to export all run session results.

Service Test stores the export options in the `ReportExportOption.xml` file in the `%AppData%\Roaming\Hewlett-Packard\STUFT\ConfigurationUI` folder. This file is useful when using the Test Batch Runner from the command line. For details, see "How to Run a Test Batch Using the Windows Command Line" on page 294.

Output Pane (Options Dialog Box > General Tab)

This pane enables you to define display options for the Output pane.



To access	Select **Tools > Options > General** tab > **Output Pane** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "Output Pane Overview" on page 180 • "Output Pane User Interface" on page 181

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

UI Element	Description
Word wrap	When selected, wraps the text of each message onto the next line.
Text Font	The font in which to display the content of the Output pane.
Size	The size in which to display the text in the Output pane.
<preview area>	Shows an example of the formatting selected in the **Text Font** and **Size** drop-down lists.

API Testing Tab (Options Dialog Box)

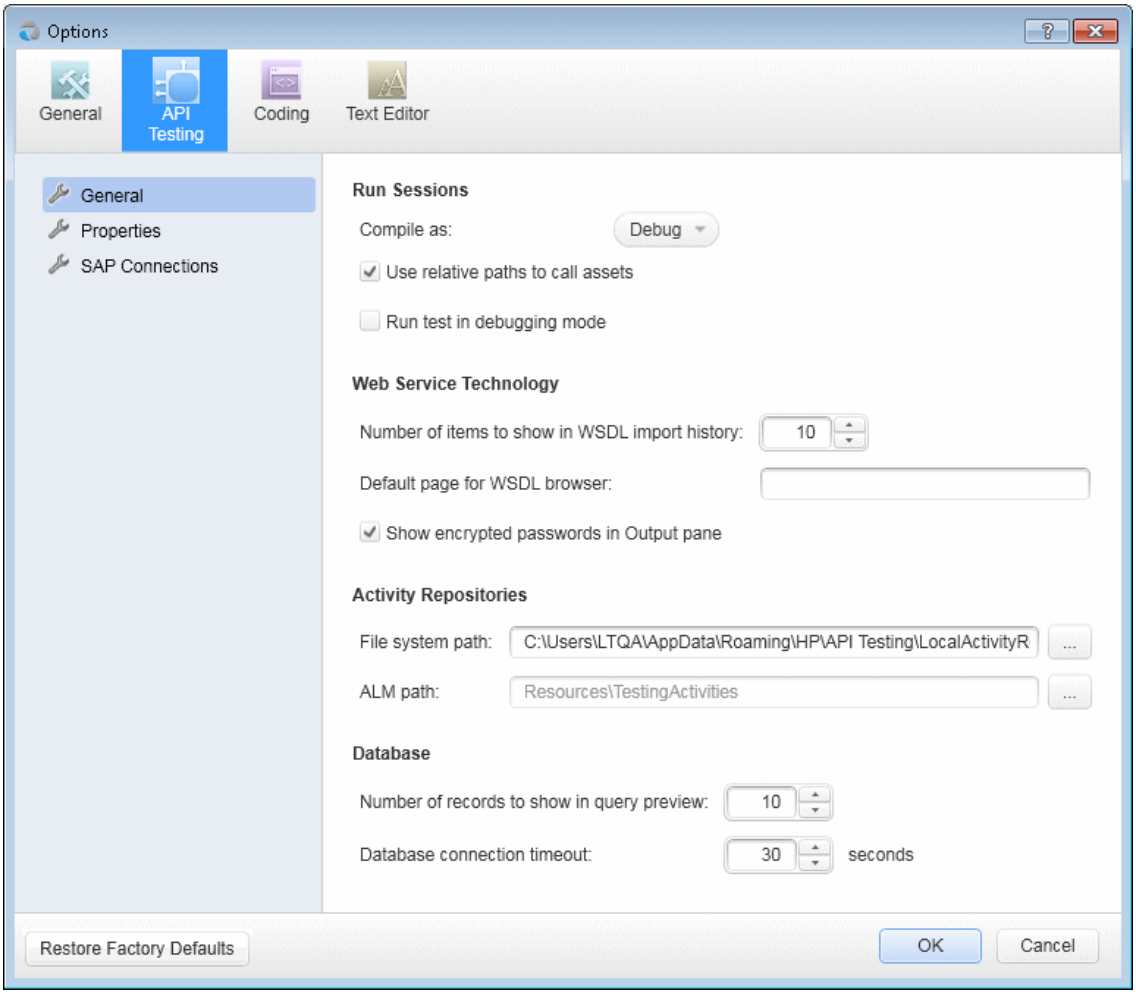
The options in the API Testing tab of the Options dialog box enable you to customize your work environment when working with Service Test tests or components. For example, you can define relative or absolute paths for saved files, repository locations, and auto values.

To access	Select **Tools > Options > API Testing** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "General Tab (Options Dialog Box)" on page 264 • "Coding Tab (Options Dialog Box)" on page 277 • "Text Editor Tab (Options Dialog Box)" on page 279

This tab includes the following panes:

General Pane (Options Dialog Box > API Testing Tab)

This pane enables you to set the test execution mode, the default location for importing WSDL files, and the repository locations for activity sharing.



To access	Select **Tools > Options > API Testing** tab > **General** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	"Auto Values Pane (Options Dialog Box > API Testing Tab)" on page 274

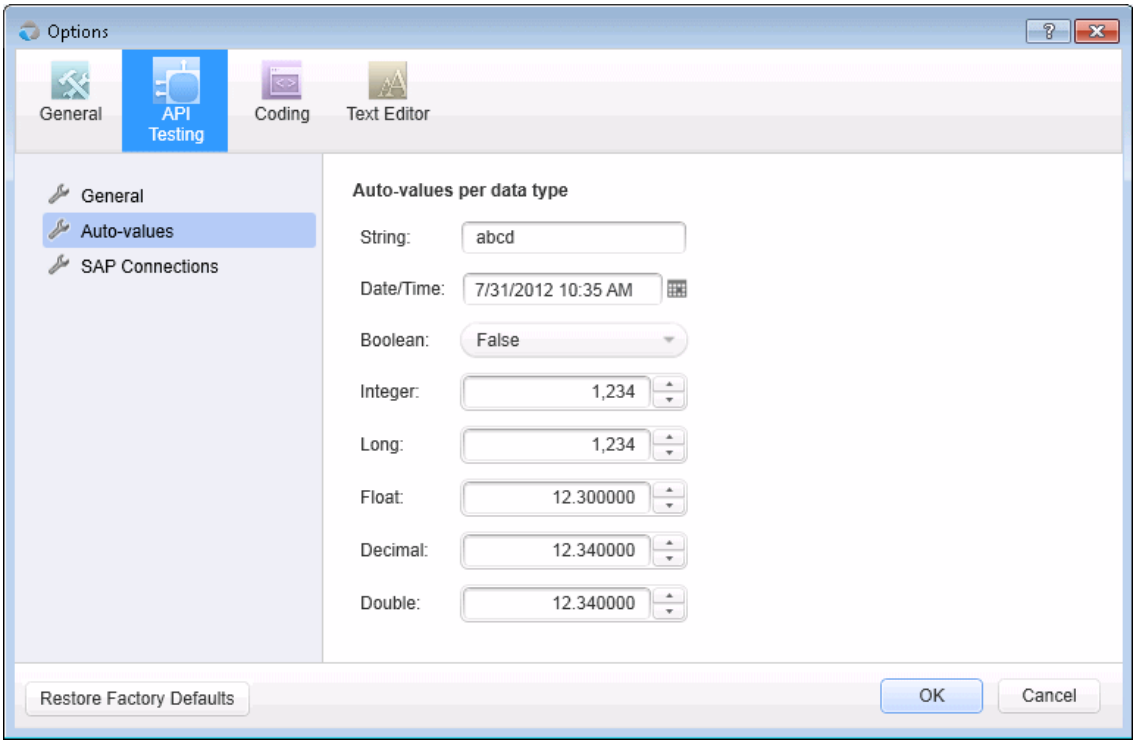
User interface elements are described below:

UI Element	Description
Compile as	The mode for a run session. Options include **Debug** or **Release**. Select the **Release** option for faster run sessions. Run sessions performed with the **Debug** option run more slowly and generate additional information in the Output pane.
Use relative paths to call assets	Instructs Service Test to use relative paths for all steps that require a path or file name.

UI Element	Description
Run test in debugging mode	Runs a test with debugging capabilities active. By default, tests run without any of the debugging capabilities, enabling them to run faster. If you check this option, it activates the debugging capabilities during a test run, slowing the speed of a test run.
Number of items to show in WSDL import history	Specifies the maximum number of WSDL addresses to display in the address drop down list of the "Import/Update WSDL from URL or UDDI Dialog Box" (described on 489). **Default:** 10
Default page for WSDL browser	The default Web page that opens when you click the **Browse** button in the "Import/Update WSDL from URL or UDDI Dialog Box" (described on 489).
Shows encrypted passwords in the Output pane	Displays passwords used in a test or component in the information displayed in the Output pane in an unencrypted form.
Activity Repositories	Displays the locations for activity repositories: • **File system path.** The folder in the file system in which to store activities. • **ALM path.** The path in the ALM repository in which to store activities. **Tip:** • Click the **Browse** button to select a location. • To move an activity to a repository so that it can be used in subsequent tests, right-click the parent node and select **Move to**. **Note:** You must be connected to ALM to set the **ALM activities path** location (select **ALM > ALM Connection**).
Number of records to show in query preview	Sets the maximum number of strings for: • The Query Preview window (accessible by clicking the **Check SQL Statement** button in the "Set SQL Statement Page " (described on page 103). • A database source in the Data Pane **Default:** `10`
Database connection timeout	The number of seconds to wait without a successful connection to a database before timing out the connection. **Default:** `30`

Auto Values Pane (Options Dialog Box > API Testing Tab)

This pane enables you to provide default values for populating activity properties.



To access	Select **Tools > Options > API Testing tab** > **Auto-values** node.
Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • To assign an auto value to a property, select **Set Auto-value** from the right-click menu in the **Value** column of the Properties pane. • To set values for an array of properties, select the parent node.
See also	"General Tab (Options Dialog Box)" on page 264

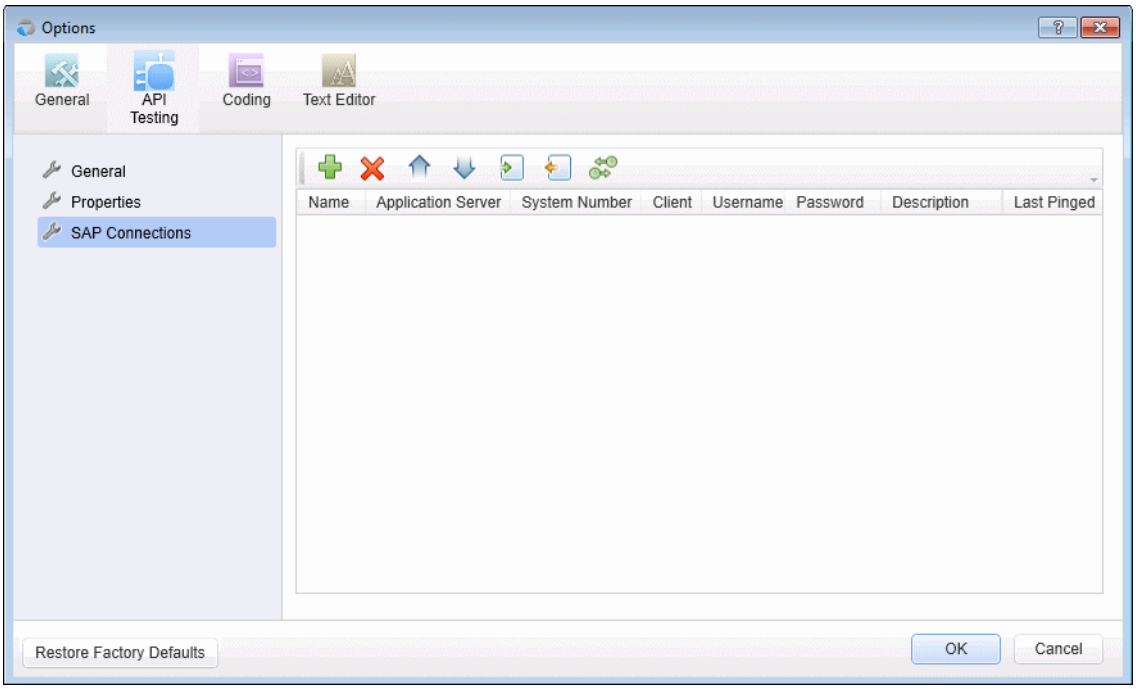
User interface elements are described below:

UI Element	Description
String	The default value to use for a property defined as a **string**. **Default:** `abcd`.
Date/time	The default value to use for a property requiring a time and date. **Default:** long notation of the current **time and date**.

UI Element	Description
Boolean	The default value to use for a property defined as **boolean**: `True` or `False`. **Default:** `False`.
Integer	The default value to use for a property defined as **integer**. **Default:** `1,234`.
Long	The default value to use for a property defined as **Long**. **Default:** `1,234`.
Float	The default value to use for a property defined as **Float**. **Default:** `12.300000`.
Decimal	The default value to use for a property defined as **decimal**. **Default:** `12.340000`.
Double	The default value to use for a property defined as **double**. **Default:** `12.340000`.

SAP Connections Pane (Options Dialog Box > API Testing Tab)

This pane enables you to manage SAP connections. You can add and remove connections, add authentication information, and check the connection.



To access	Select **Tools > Options > API Testing tab > SAP Connections** node.
Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • To define a connection, you must have the 32-bit version of SAP .NET Connector installed on your machine. The installation is available on the SAP Help portal, at http://help.sap.com/saphelp_NW04/helpdata/en/e9/23c80d66d08c4c8c044a3ea11ca90f/content.htm.
Relevant tasks	"How to Create an SAP Test Step" on page 485
See also	"Local Activity Overview" on page 464

User interface elements are described below:

UI Element	Description
	**Add Connection.** Adds a new editable entry to the list of connections.
	**Delete Connection.** Deletes the selected connection.
	**Move up.** Moves the selection up on the connection list.
	**Move down.** Moves the selection down in the connection list.
	**Import.** Allows you to import a saplogon and connection file (`.ini` or `.xml`) with the connection settings information.
	**Export.** Allows you to export the connection settings to an XML file.
	**Ping Connection.** Tests the selected connection and inserts the date and time in the **Last Pinged** column.
<SAP connection list>	Displays information about the current connection: • **Name.** The connection's name. • **Application Server.** The URL or IP of the application server. • **System Number.** The value of the SAP system number. • **Client.** The type of client. • **Username/Password.** The authentication information. • **Description.** A description of the connection. • **Last Pinged.** The date and time that a ping request was issued to the server.

Coding Tab (Options Dialog Box)

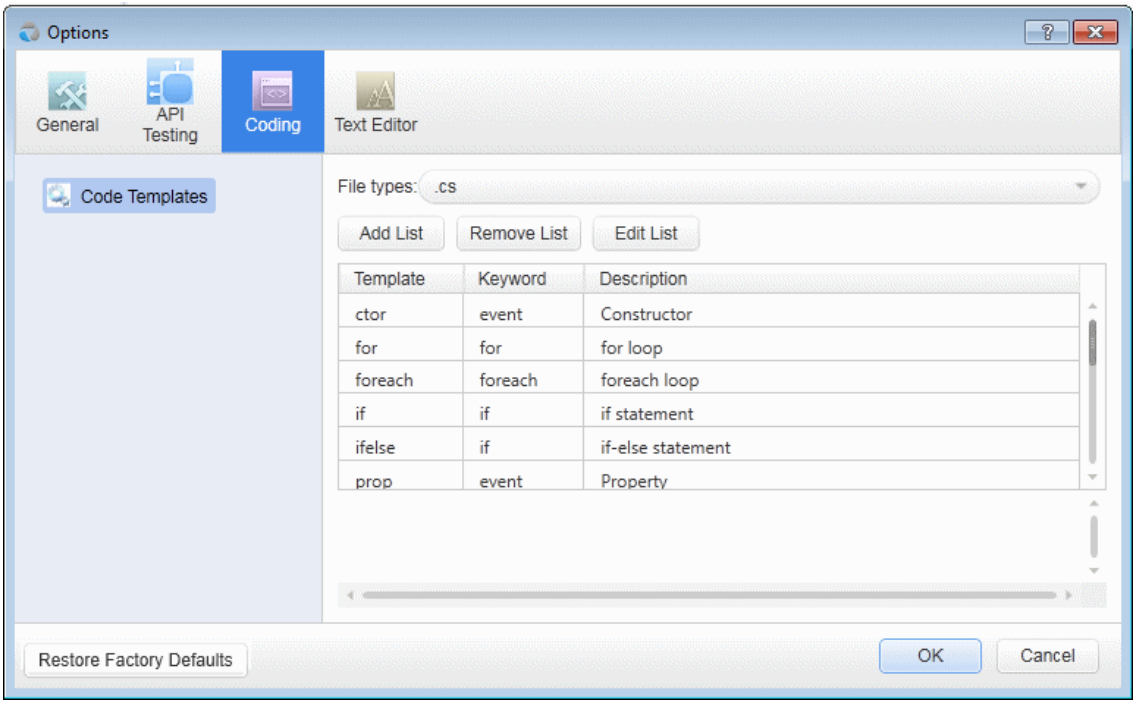
This tab enables you to set options that facilitate designing code.

To access	Select **Tools > Options > Coding** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "General Tab (Options Dialog Box)" on page 264 • "API Testing Tab (Options Dialog Box)" on page 271 • "Text Editor Tab (Options Dialog Box)" on page 279

This tab includes the following pane:

Code Templates Pane (Options Dialog Box > Coding Tab)

This pane enables you to create and edit templates of pre-designed code snippets or blocks of text used for automatic code completion. You can create different lists of templates for different file types.



To access	Select **Tools > Options > Coding** tab > **Code Templates** node.

Important information	• The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. • You can edit the names, keywords, descriptions, and code snippets of any template defined in this pane. • Service Test provides default file types and snippets. You can add additional lists of file types and snippets as needed.
Relevant tasks	"How to Use Code Snippets and Templates" on page 139
See also	"Automatic Code Completion" on page 135

User interface elements are described below:

UI Element	Description
File types	The file types for which the displayed list of templates is used. Select an item from the drop-down list to display the templates associated with the selected file extensions. Default options include: • `.vb` files . • `.cs` files
Add List	Adds a new list of templates and the file types for which to use them. A new item is added to the **File types** drop-down list.
Remove List	Removes the currently displayed list of templates and the associated **File types** item. **Caution:** Deleting a list is irreversible.
Edit List	Enables you to edit the file extensions associated with the displayed list of templates.
Template	The name of the template.
Keyword	The keywords that trigger the use of the template when invoking code completion in files of the selected types.
Description	A string that describes the code snippet.
<code snippet>	The code that is inserted into your code file. The snippet can include text that is inserted as is, as well as variables (prefaced by a dollar sign $) that are replaced with actual values when they are inserted into the code file. **Tip:** To edit the code snippet, modify the text in this area and click another template row or **OK**.

Text Editor Tab (Options Dialog Box)

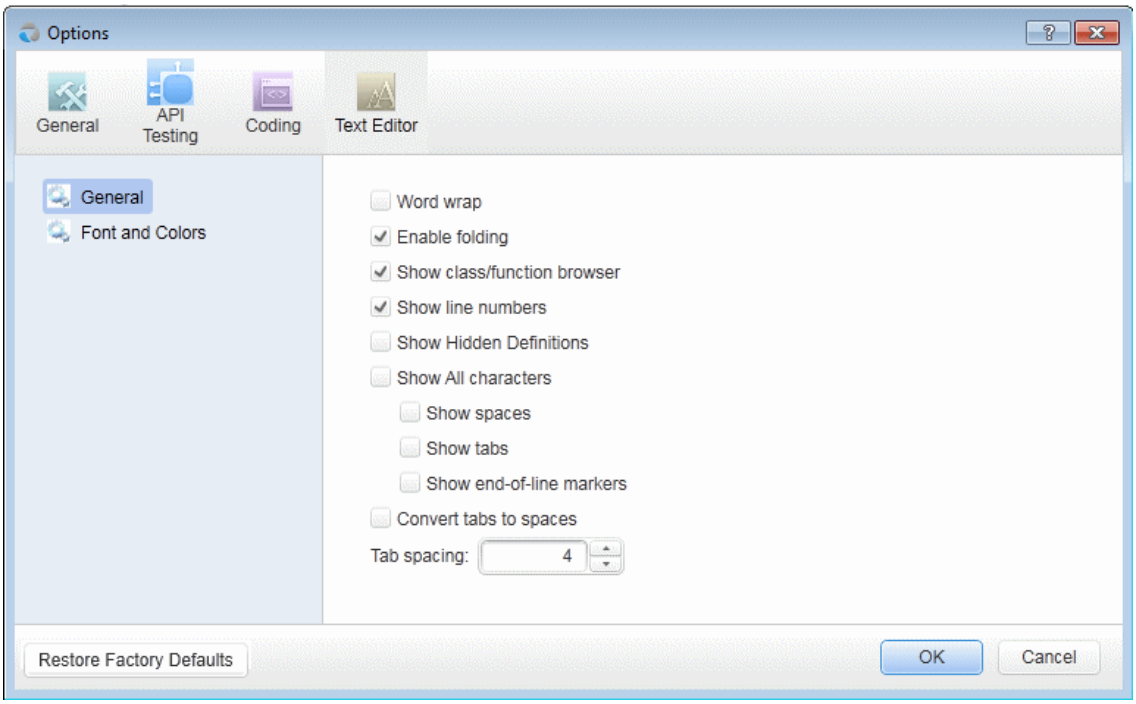
This tab enables you to set options for editing text and code files.

To access	Select **Tools > Options > Text Editor** tab.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "General Tab (Options Dialog Box)" on page 264 • "API Testing Tab (Options Dialog Box)" on page 271 • "Coding Tab (Options Dialog Box)" on page 277

This tab includes the following panes:

General Pane (Options Dialog Box > Text Editor Tab)

This pane enables you to set general preferences for editing code and text files.



To access	Select **Tools > Options > Text Editor** tab > **General** node.

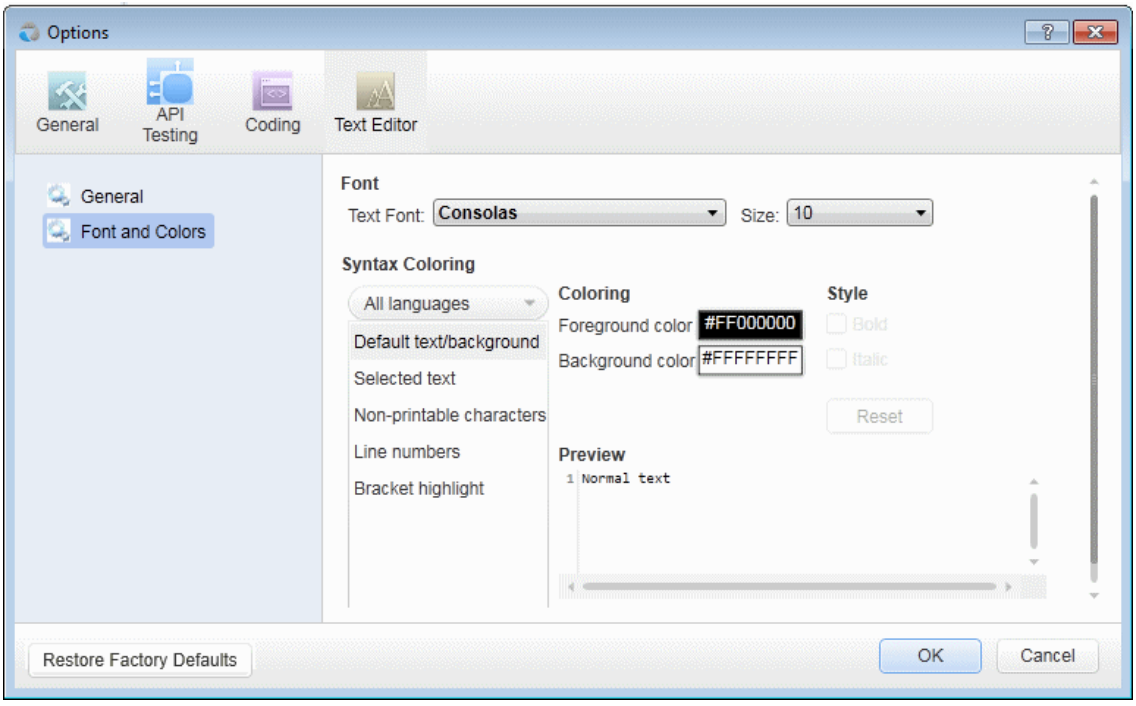
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults.
See also	• "Editing Text and Code Documents" on page 133 • "Editor User Interface" on page 147

User interface elements are described below:

UI Element	Description
Word wrap	Wraps lines of code to continue onto subsequent lines without the insertion of a line break character. If disabled, the line of code continues indefinitely on a single line until you manually stop it by pressing the **ENTER** key or entering a line break character.
Enable folding	Enables sections of code that are grouped together to be expanded and collapsed.
Show class/function browser	Shows or hides the drop-down lists of classes and functions. These lists are located at the top of the Editor. Using the browsers, you can navigate to different classes and functions within your event handler or user code file.
Show line numbers	Shows or hides the line number to the left of each line of code.
Show hidden definitions	Shows or hides text and function definitions defined as hidden.
Show All characters	Displays all **SPACE**, **TAB**, and **End of Line** characters. You can also select to display only some of these characters by selecting or clearing the relevant check boxes.
Convert tabs to spaces	Changes **TAB** keystrokes in your code to the number of **SPACE** characters defined in the **Tab spacing** option.
Tab spacing	The number of characters inserted into a line of code for a **TAB** character. The number of characters can range from 1 to 16. **Default:** 4

Fonts and Colors Pane (Options Dialog Box > Text Editor Tab)

This pane enables you to specify color preferences for the text you are editing. You can set unique coloring rules for each code element.



To access	Select **Tools > Options > Text Editor** tab > **Fonts and Colors** node.
Important information	The **Restore Factory Defaults** button resets all Options dialog box options to their defaults. The options in this pane apply only to documents in the Editor.
See also	• "Editing Text and Code Documents" on page 133 • "Editor User Interface" on page 147

User interface elements are described below:

UI Element	Description
Font	The font name and size to use for all elements. The Default is Consolas, 10 points. **Note:** When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your user code file may not be correctly displayed in the document window. However, the test or component still runs in the same way, regardless of the font you choose. If you are working in an environment that is not Unicode-compatible, you may prefer to choose a fixed-width font, such as `Courier`, to ensure better character alignment.

UI Element	Description
Syntax Coloring	The language or languages whose appearance you want to customize. Available options: • **All languages** • **C#** • **XML** The box below your selection displays all of the elements for which you can configure color and style settings. You can modify the **Foreground** and **Background** colors for a selected element. You can also modify the **Style** (**Bold** or **Italic**), if enabled.
Preview	An example of the text element according to your selections above.
Reset	Resets all colors and styles to default settings.

Part 4: Service Test Running and Debugging Operations

Chapter 17: Running Tests and Components

This chapter includes:

Concepts

Running API Tests - Overview

You can run tests directly from the Service Test interface or from the command line. The test run enables 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 "Run Dialog Box" on page 296.

For details about the command line, see "Command Line Syntax" on page 597.

The Run Results Viewer provides a detailed report about each step and its checkpoints.

When you run custom code from the user interface, you can also use breakpoints and other tools to debug your code.

Using the Options dialog box, you can set the run configuration—**Debug** or **Release**. The **Release** mode is more efficient, and is recommended for Load Testing. For details, see "General Pane (Options Dialog Box > API Testing Tab)" on page 271.

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/Output Properties Tab (Properties Pane)" on page 201.

For Web Service steps, you can validate the SOAP response against a schema or WS-I standards. For details, see "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.

Server-Side Execution

When working with tests saved on an ALM server with Lab Management enabled, you can run tests in server-side execution.

In server-side execution, you run tests on remote host machines at predefined times or on an ad-hoc basis, not requiring anyone to be logged in to the host to initiate and control the test runs.

In client-side execution, although the test is stored on ALM, it runs on the machine running Service Test, and requires someone to be logged in to that machine.

You set up server-side execution on ALM:

First create functional test sets in ALM's **Test Lab** module. A functional test set is a group of automated tests or test configurations in an ALM project, designed to achieve a specific goal.

Next, using ALM's **Lab Resources > Testing Hosts** module, you indicate the hosts upon which the tests and can run remotely.

Finally, in the ALM's **Timeslots** module, you schedule the test runs. If you are running the tests ad-hoc, you do not need to set a timeslot.

You can also link your test parameters to ALM AUT Environment parameters, which are then used when running the tests using Server-Side execution. For details, see "AUT Environment Parameters" on next page.

Note: Server-side execution is available only for ALM Edition and Performance Center Edition, version 11.50 or later.

For task details, see "How to Perform Server-Side Execution" on page 290.

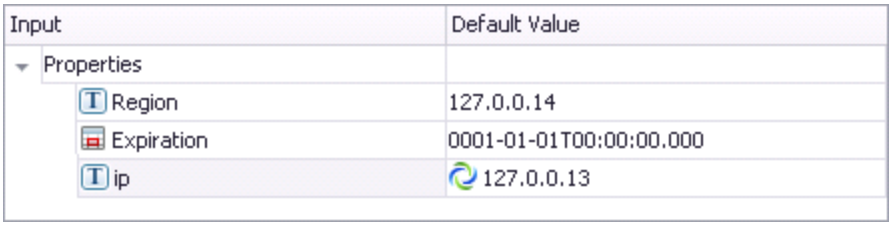
For additional information, see the *HP Application Lifecycle Management User Guide*.

AUT Environment Parameters

When performing server-side execution, you can use parameters from an ALM AUT Environment.

When defining test parameters, you can link them to parameters defined in a specific AUT environment in ALM. All of the AUT parameters that you use must be from the same ALM AUT environment.

In the Properties pane, an ALM icon next to a parameter's default value indicates that it is an AUT parameter.



When the test runs in server-side execution, it uses the values provided by ALM for the corresponding AUT environment parameter.

When the test runs from Service Test, the test uses the parameters' default values.

If you save the test to the file system, the links to the AUT parameters are removed, and the test parameters become ordinary parameters. The default values remain defined.

For task details on linking test parameters to ALM AUT environment parameters, see "How to Perform Server-Side Execution" on page 290.

What happens if the ALM AUT parameters linked to the Service Test test parameters are deleted from ALM?

- If you open a test with test parameters that are linked to deleted ALM AUT parameters, the links remain unchanged.
- If you run a test with test parameters linked to deleted ALM AUT parameters, the test will fail to run.

Test Batch Runner Overview

Test Batch Runner enables you to run tests in a collective, successive test run. Tests are run individually but sequentially in a single session.

You can use Test Batch Runner only on tests stored on the file system. You cannot use Test Batch Runner for tests stored in ALM.

Using Test Batch Runner, you create a list of tests and save the list as a batch `.mtb` file, so that you can run the same batch of tests again at another time. You can choose to include or exclude a test in your batch list from running during a particular batch run without affecting the other tests in the batch.

You can add tests individually to Test Batch Runner by navigating to the folder in which the test is located. Test batches can also be added to another test batch by adding an `.mtb` test batch file to a new test batch. When Test Batch Runner opens, it checks to make sure that all tests within a test batch exist.

When running the test batch, the Output pane allows you view the results of the test run in run time. During the test batch run, the Output pane displays the test's path in the file system, the progress of the test, as well as any errors that occur during the run.

Following the test batch run, the results are saved to a run results file including whether the test passed or failed and errors in running the test. The **Report** column of the **Tests** pane displays a link to the run results file.

If you do not want to run the test batch through the Test Batch Runner interface, you can run the Test Batch Runner from the Windows Command Line. You provide the location of a `.mtb` file, a test folder, or a directory of tests as a command argument and Test Batch Runner runs the test batch and displays the run results.

Tip: Using the command line options of the Test Batch Runner, you can include Service Test tests in as part of build runs in a continuous integration system.

Tasks

How to Run an API Test

This task describes how to run a test through the user interface and view its results.

For details on running a test through the command line, see "Command Line Syntax" on page 597.

This task includes the following steps:

- "Add new tests to a solution - optional" below
- "Set values for the test variables - optional" below
- "Configure checkpoints" below
- "Save the test in ALM - optional" on next page
- "Add external references - optional" on next page
- "Set the number of iterations" on next page
- "Select a location for the run results - optional" on next page
- "Set runtime property values - optional" on page 290
- "Validate the structure of the SOAP response " on page 290
- "Run the test" on page 290
- "Debug the test - optional " on page 290
- "Analyze the Results" on page 290

1. ### Add new tests to a solution - optional

   To add more tests to the solution, select **File > Add > New Test** or **File > Add > Existing Test.** To add tests to an ALM repository, first connect to the ALM server. For details, see "ALM Connection Dialog Box" on page 335.

   Service Test adds the test to the Solution Explorer pane.

   For user interface details, see the "Add Test/Component to Solution Dialog Box" on page 64 or "Add <Existing Document> to Solution Dialog Box" on page 63

2. ### 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 59.

3. ### Configure checkpoints

   Checkpoints let you validate the results. For details, see "Checkpoint Validation" on page 585.

   To add checkpoints, do the following:

   a. Click the **Input/Checkpoints** tab  . By default, the pane displays the **XML** tab. Provide an expected value for each row in one of the following ways:

- Manually
- Select Link 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. Set checkpoints for array elements. For details, see "How to Set Array Checkpoints" on page 589:

e. To validate against an XPath expression, click the **XPath** tab and provide the expression. For details, see "How to Set XPath Checkpoints" on page 592:

f. 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 651.

For user interface details, see the "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.

For details on how to work with array checkpoints, see "How to Set Array Checkpoints" on page 589.

4. ## Save the test in ALM - optional

To save results in ALM, you need to first save the actual test in ALM. Connect to ALM and then select **File > Save as.**

5. ## Add external references - optional

You can add one or more references to your test.

a. Open the Solution Explorer pane.

b. Expand the test's tree and select the **References** node.

c. Select **Add Reference** from the context menu.

d. Specify an external `.dll` file as a reference.

For details, see the "Add Reference Dialog Box" on page 245.

6. ## Set the number of iterations

Click in the **Test Flow** or **Loop** box and select the **Input/Checkpoints** tab . Set the number of iterations in the Properties pane. For details, see "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.

7. ## Select a location for the run results - optional

> **Tip:** To run a test bypassing the Run dialog box, select **Run > Run Now**.

a. Click the **Run** button on the toolbar and select the **Results Location** tab.

b. For tests 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 "Run Dialog Box" on page 296.

c. For tests saved in ALM, specify a run name. If relevant, select a **Test Set** and **Instance**.

8. ### Set runtime property values - optional

   In the Run dialog box, click the **Input Parameters** tab and set the values that you want the run session to use for the listed properties.

9. ### 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. Set the desired validations in the bottom of the Checkpoints area. For details, see the "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.

10. ### Run the test

    Click the **OK** button in the Run dialog box.

11. ### Debug the test - optional

    By default, Service Test runs tests in release mode—not debugging mode. This allows the test to run much quicker. To run the test in debugging mode, which also allows you to pause the test during its run, you need to enable it in the Options dialog box. For details, see "General Pane (Options Dialog Box > API Testing Tab)" on page 271.

    In debugging mode, select **Run > Pause** to stop the test run at any point. Use the debug panes to solve any runtime issues. For user interface details, see the or "Debug Panes" on page 110.

12. ### Analyze the Results

    View the results in the Run Results Viewer. The Run Results 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, select **View > Last Run Results**.

    To prevent the Results viewer from opening, select **Tools > Options > General** tab **> General** node**.** Clear the **View results when run session ends** option.

How to Perform Server-Side Execution

This task describes how to run a test through server-side execution.

For an overview, see "Server-Side Execution " on page 285.

This task includes the following steps:

1. ## Create tests and save them in ALM

   Create your tests and save them to ALM. Make sure to connect to a project with Lab Management support. For details on connecting to ALM, see "ALM Connection Dialog Box" on page 335.

2. ## Create test sets

   a. Open ALM's **Testing > Test Lab** module.

   b. Select **Test Sets > New Test Set**. Create a functional type test set and specify the required information. Click **OK**.

   c. Select the **Execution Grid** tab and click **Select Tests**.

   d. In the Test Plan tree, select the tests you want to add to the test set.

3. ## Select AUT Parameters - optional

   To use property values from value sets stored in ALM:

   a. In ALM's Lab Management, open the **Lab Resources > AUT Environments** module. Define an AUT environment with parameters. Set the desired values. For details, see the *HP Application Lifecycle Management User Guide*.

   b. In Service Test, open the Properties pane to the test parameters tab:

   Do one of the following:

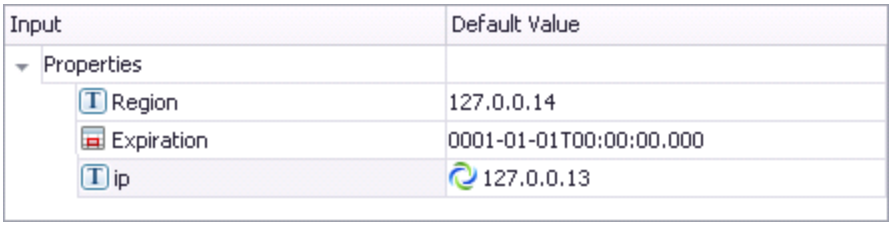
   - Open an API test, select the **Start** or **End** steps in the canvas, and open the Properties pane. Then select the Test Input/Output Properties tab  .
   - Select a GUI test in the solution explorer. Open the Properties pane, and select the Parameters tab  .

   c. Click **Add > Add Input Parameter**.

   d. In the Add Input Parameter dialog box that opens, click the **Select ALM application parameters** button  .

   e. In the Select AUT Parameter dialog box that opens, expand the **AUT Environment** node and select a parameter. Click **OK**. For details, see "Select AUT Parameter Dialog Box" on page 299.

   f. In the Add Input Parameter dialog box, provide a name for the parameter and set the default value, if necessary. Click **OK**. In the Properties pane, an ALM icon  next to a parameter's default value indicates that it is an AUT parameter.



g. To edit the parameters, click the **Edit Parameter** button in the Properties pane.

h. To change the parameter to an ordinary parameter, without a linkage to an AUT environment, click the **Remove link to ALM application parameters** button in the Edit Parameter dialog box .

You can repeat these steps to add additional test parameters and link them to ALM AUT parameters, but all of the AUT parameters that you use must be from the same ALM AUT environment.

For more details, see "AUT Environment Parameters" on page 286.

4. ### Set up the hosts

In the **Lab Resources > Testing Hosts** module, select host machines. You can define a purpose for the host, such as Service Test. For details, see the *HP Application Lifecycle Management User Guide*.

5. ### Schedule the tests - optional

In the **Testing > Timeslots** module, schedule the run times. If you are running the tests ad-hoc, you can skip this step.

6. ### Run the tests

Run the tests from the **Tests Lab** module. For details, see the *HP Application Lifecycle Management User Guide*.

How to Create and Run a Test Batch

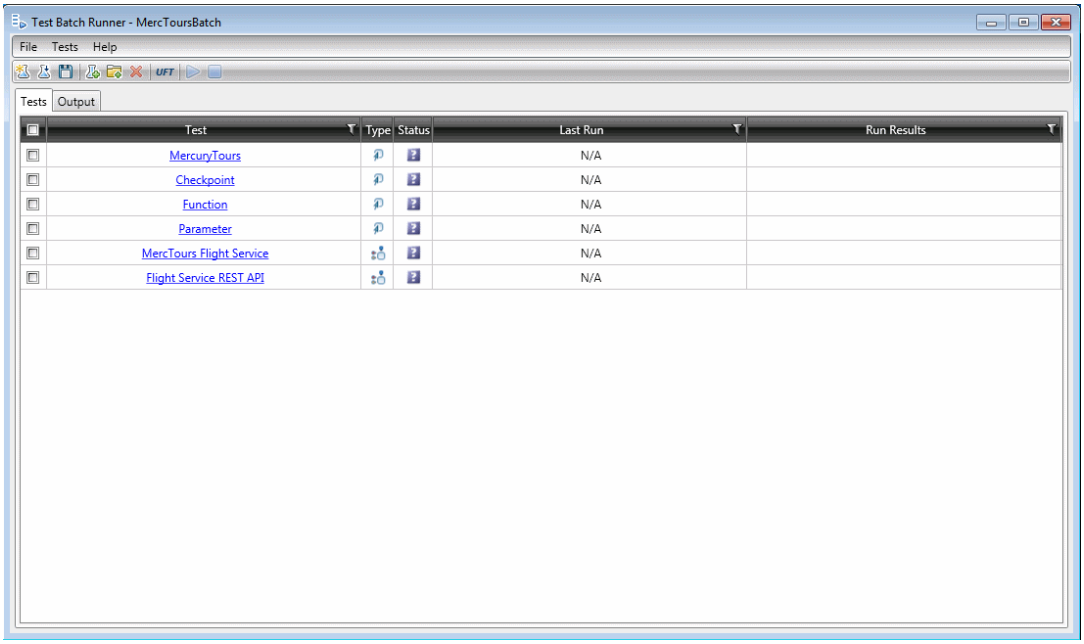
This task explains how to create and run a test batch using Test Batch Runner.

This task includes the following steps:

1. ### Open Test Batch Runner

Select **Start > All Programs > HP Software > HP** Service Test **> Tools > Test Batch**

Runner. This opens a separate window for the Test Batch Runner program.



Note:

- You do not need to have Service Test open to use Test Batch Runner.
- For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

2. Access the test batch file

- To create a new test batch file, select **File > New** or click the **New** batch file button . Give your batch file a meaningful name and assign it to a place in your directory.
- To open an existing batch file, select **File > Open** or click the **Open** batch file button . In the Open dialog box, navigate to the folder in which the batch file is found and click **Open.** The tests from the opened batch file are added to the Test Batch Runner main window.

3. Add batches or tests

- To add a test batch file (**.mtb**), select **File > Add** or click the **Add** button . Navigate to the folder in which the batch file is saved.
- To add individual tests, select **Tests > Add** or click the **Add** button . In the **Browse For Folder** dialog box, select the folder in which your tests are located. All the tests from the selected folder are added to the **Tests** pane in the main Test Batch Runner window.

Note: When adding tests through the **Tests > Add** menu command, you must select all the tests from the target folder. If you do not want to run all the tests in the target folder, select the check boxes next to the tests you want to run before you run the test batch.

4. **Select the tests to be part of the test batch run**

   Select the checkboxes for the tests that you want to include in the test batch run.

5. **Run the test batch**

   Click the **Run** button to run the test batch. The Output pane provides run log details of the batch run while the batch is running.

6. **View the test batch run results**

   In the Tests pane, click the results link for a specific test in the **Run Results** column. This opens the results for that test in the Run Results Viewer. For details on the Run Results Viewer, click F1 within the viewer window.

How to Run a Test Batch Using the Windows Command Line

This task describes how to run a test using the Windows Command Line.

This task includes the following steps:

- "Open the Windows Command Line window" below
- "Provide the source folder for the batch file or tests" below
- "Run the test batch" below
- "View the test batch run results" below

1. **Open the Windows Command Line window**

   Run `cmd.exe` to open the Command Line window. (For example, from the Windows Run dialog box.)

2. **Provide the source folder for the batch file or tests**

   In the Command Line window, enter `UFTBatchRunnerCMD.exe` and the **source** switch followed by the test batch file (`.mtb`) or folder containing the test.

   For example, your command line might contain text like this:

   ```
   UFTBatchRunnerCMD.exe -source "C:\users\MySample.mtb"
   UFTBatchRunnerCMD.exe -source "C:\users\APITest1"
   ```

3. **Run the test batch**

   After entering the Test Batch Runner command and the location of the folder containing your tests, press ENTER. Test Batch Runner runs the test batch.The test log is displayed in the command window.

4. **View the test batch run results**

   When the test batch run is complete:

   a. Open the Run Results Viewer from the Start menu. The Open Run Results dialog box opens.

> **Note:** For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

b. In the Open Run Results dialog box, select the **Results XML file** option.

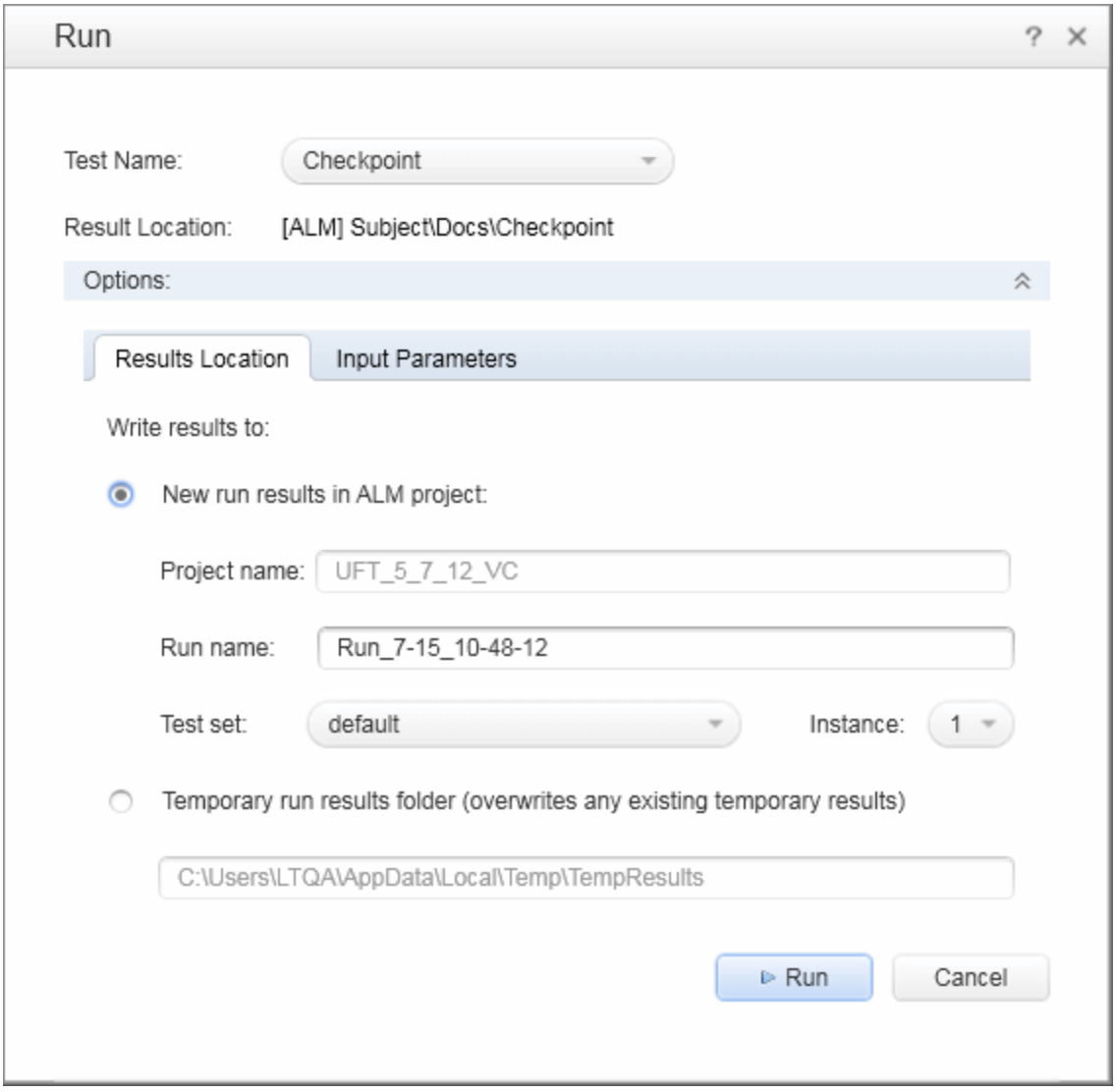
c. Navigate to the `Results.xml` file under the test's **Report** folder and click **Open**.

   Each test has its own results. For details, see the *HP Run Results Viewer User Guide*.

Reference

Run Dialog Box

This dialog box lets you prepare for the test run by selecting the test to run and providing a location for the results.



To access	1. Do one of the following: ▪ Ensure that a test or component is in focus in the document pane. ▪ In the Solution Explorer, select a test or component node. 2. Select the following: ▪ Click the **Run** button or select **Run > Run**.

Important information (for tests)	• When the test stops running, the Run Results Viewer opens unless you have cleared the **View results when test run ends** check box in the **Run Sessions** pane of the Options dialog box (**Tools > Options > General** tab **> Run Sessions** node). For details, see "Run Sessions Pane (Options Dialog Box > General Tab)" on page 265. • You can report defects to an ALM project either automatically as they occur, or manually directly from the Run Results Viewer. For details, see the online help in the Run Results Viewer.
Relevant tasks	"How to Run an API Test" on page 288

User interface elements are described below:

UI Elements	Description
Test Name	The test or component to run. You can select any open item from the drop-down list.
Results Location	The target location for the results, as specified in the **Results Location** tab.
Options	Expands or collapses the dialog box to show the **Results Location** and **Input Parameters** tabs.
Run	Begins the run session.

Run Dialog Box: Results Location Tab (For Tests Stored on File System)

This tab enables you to specify where to save run results when running tests stored on the file system.

User interface elements are described below:

UI Elements	Description
New run results folder	An area allowing you to indicate the results location. The area displays the default path and folder name in which the results are saved. A new folder is created for each run. By default, results are stored in the following locations: • **For tests:** The test folder • **For components:** An ALM cache folder on your computer

UI Elements	Description
Temporary run results folder	Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder. **Note:** • The path in the text box of the **Temporary run results folder** option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts. • Service Test stores temporary results in `%TMP%\TempResults` (which is usually `<System Drive>\Documents and Settings\<user name>\Local Settings\Temp\TempResults`).

Run Dialog Box: Results Location Tab (For Tests Stored in ALM)

This tab enables you to specify where to save run results when running tests stored in ALM.

User interface elements are described below:

UI Elements	Description
New run results in ALM project	• **Project name.** Displays the ALM project to which you are currently connected. • **Run name.** The name of the run. You can accept the automatically generated name or enter a different one. • **Test set.** A group of tests selected to achieve specific testing goals. For example, you can create a test set that tests the user interface of the application or the application's performance under stress. (You define test sets when working in the ALM test run mode. For details, see your ALM documentation.) • **Instance.** The instance of the test in the test set. If there is more than one instance, select the instance of the test for which you want to save the results.
Temporary run results folder	Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder. **Note:** • The path in the text box of the **Temporary run results folder** option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts. • Service Test stores temporary results in `%TMP%\TempResults` (which is usually `<System Drive>\Documents and Settings\<user name>\Local Settings\Temp\TempResults`)

Run Dialog Box: Input Parameters Tab

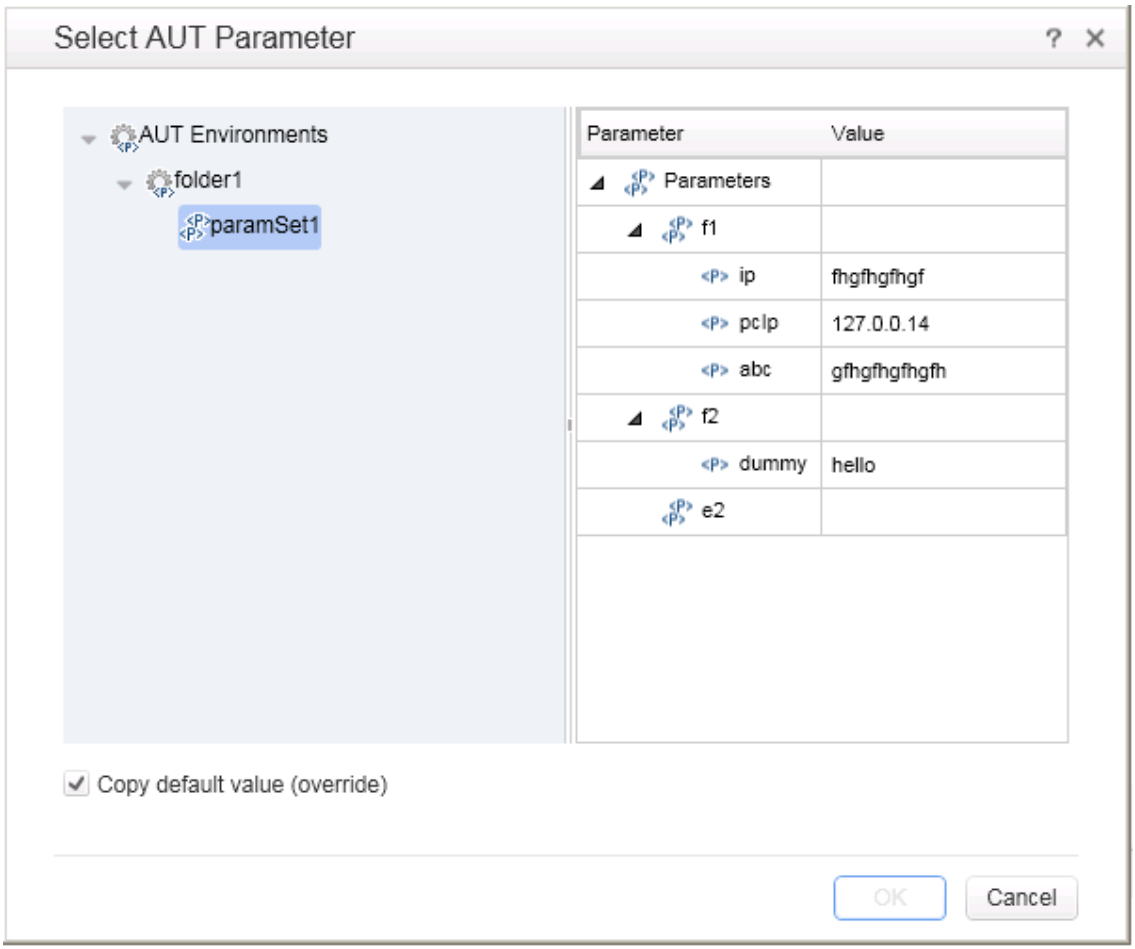
This tab enables you to specify to specify the run-time values of input parameters to be used during the run session.

User interface elements are described below:

UI Elements	Description
Input parameters	The input parameters that were defined for the test or component. **To set the value of a parameter to be used during the run session:** Click in the **Value** field for the specific parameter and enter the value, or select a value from the list. If you do not enter a value, it uses the default value from the Test Settings or Business Component Settings dialog box during the run session.

Select AUT Parameter Dialog Box

This dialog box lets you select an AUT parameter stored in ALM's **Lab Resources > AUT Environment** module in ALM Lab Management and link it to a test parameter.



To access	1. Open a test stored on an ALM server with Lab Management. Do one of the following: ▪ Open an API test, select the **Start** or **End** step in the canvas, and open the Properties pane. Then select the Test Input/Output Properties tab. ▪ Select a GUI test in the solution explorer. Open the Properties pane, and select the Parameters tab. 2. Click **Add > Add Input Parameter** or select a parameter and click the **Edit Parameter** button. 3. In the Add/Edit Input Parameter dialog box, click . **Note:** Available only if Service Test is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
Important information	The AUT parameters are used for Server Side Execution. To set up Server Side execution, create test sets and schedule the tests using the Lab Management module. For details, see "Server-Side Execution " on page 285. **Note:** All of the AUT parameters that you use must be from the same ALM AUT environment. For details, see the *HP Application Lifecycle Management User Guide*.
Relevant tasks	"How to Perform Server-Side Execution" on page 290
See also	• "Add Input/Output Parameter for Test Settings" on page 499

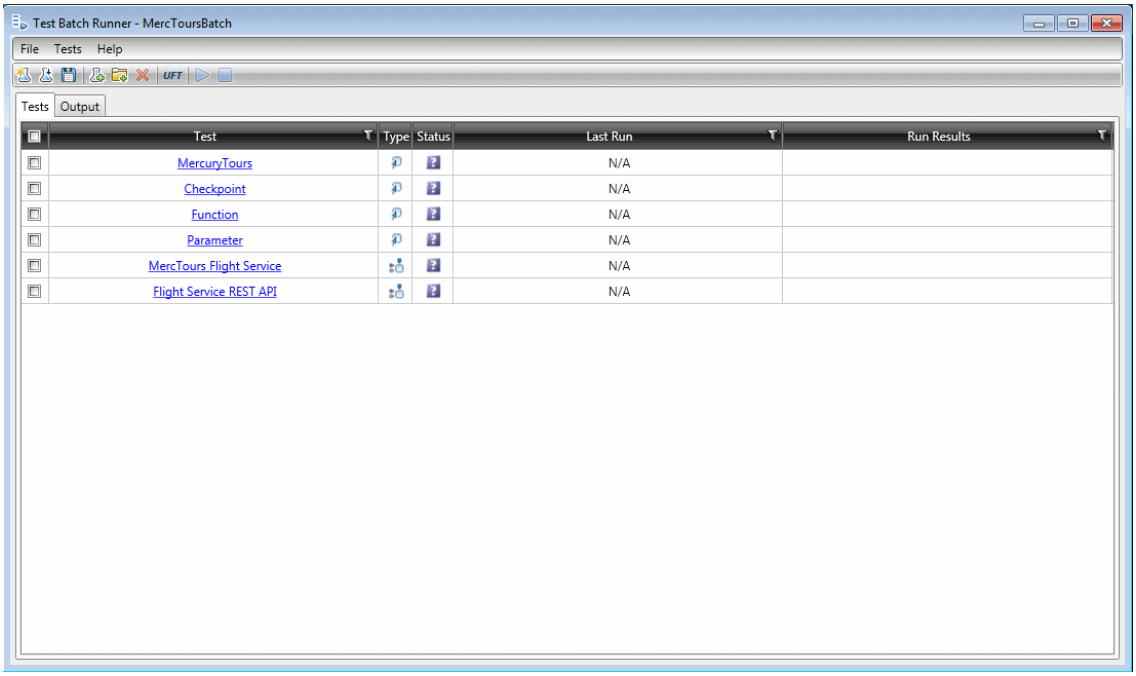
User interface elements are described below:

UI Elements	Description
AUT Environments	The list of AUT Parameters environment defined in the **Lab Resources > AUT Environment** module. Expand the **AUT Environments** node to view the available AUT environments, and then select an environment to view the available parameters. If any test parameter is already linked to an AUT parameter, only the environment containing that AUT parameter is displayed.
Parameter column	The AUT parameter names, grouped in AUT environment configurations.
Value column	The value defined for the AUT parameter in ALM.

UI Elements	Description
Copy default value (override)	Specifies whether to copy the parameter value displayed in this dialog box into the **Default Value** box in the Edit/Add Input Parameter dialog box. Copying overwrites any **Default Value** previously defined in the Edit/Add Input Parameter dialog box. However, after it is copied, you can edit the **Default Value** manually.

Test Batch Runner Window

The Test Batch Runner enables you to create and maintain test batch files and add tests to a test batch file.



To access	Select **Start > All Programs > HP Software > HP Service Test > Tools > Test Batch Runner.** **Note:** For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.
Important information	• You can use Test Batch Runner without having Service Test open. • Test Batch Runner cannot include tests created in QuickTest 9.2 or earlier or Service Test 10.00 or earlier. • The panes within this window can be moved or pinned. Click on a pane and drag it to move it to your desired location, or right-click on the title of the pane to change the options for its display.

Relevant tasks	"How to Create and Run a Test Batch" on page 292
See also	"Test Batch Runner Menu Commands" on next page

Tests Pane

The user interface elements below describe the columns in the Tests pane.

UI Element	Description
	Checkboxes that enable you to select the tests to include in the current test batch run.
Test	The name of the tests or test batches included in the test batch.
	Opens the filter window that enables you to group tests and results included in the batch file. You can filter the tests by a number of criteria using the drop-down menus at the bottom of the filter window. You can use multiple filters by choosing the combination argument from the middle drop-down menu. **Note:** This filter can be used for both the **Test**, **Last Run**, and **Run Results** columns.
Type	Displays the type of test: Service Test (ST) or QuickTest (QTP).
Status	Displays the run status of the test: • **Unknown.** The test has not been run or its status is unknown. • **Passed.** • **Failed.** • **Running.** • **Error.** The test encountered an error when running the test, such as Test Batch Runner not being able to find the test or a test missing an testing object. **Note:** If the status is displayed as **Error**, hover over the error icon to see a short description of the error.
Run Results	The link to the folder in which the results for the test is saved.

Output Pane

The Output pane displays the run long of the test batch. This includes information on the test run:

- the test currently running
- the step that is currently running within a test

- errors encountered during the test run
- the location of the run results.

> **Note:** This area will remain blank unless there is a test batch running or a batch that has completed running.

Test Batch Runner Menu Commands

You manage your test batch and individual tests the **File** and **Tests** menu commands.

File Menu Commands

File menu commands are used to create, update, and maintain test batch (`.mtb`) files.

Command	Shortcut Key	Function
New	**CTRL+N**	Creates a new batch file (`.mtb`).
Open	**CTRL+O**	Opens an existing batch file.
Add	**INSERT**	Adds a batch file to the current batch.
Save	**CTRL+S**	Saves a new batch file.
Save As		Enables you to save the batch file under another name.
Exit	**ALT+F4**	Closes Test Batch Runner.

Tests Menu Commands

Test menu commands are used to add and maintain individual tests within the batch file.

Command	Shortcut key	Function
Add		Adds individual tests (not batch files) to the current batch. **Note:** When using this command, you must add all tests from the selected folder to the batch file.
Remove	**DEL**	Removes a test from the batch file.
Run	**F5**	Runs the selected tests in the batch file.
Stop		Stops the test batch run.

Test Batch Runner Toolbar

Many of the menu commands are also available by default from the toolbar, shown below:

In addition, the Service Test button is not present in the **File** or **Tests** menus. It is described below:

UI Element	Description
UFT	**Use HP Unified Functional Testing license.** Instructs Test Batch Runner to use a Unified Functional Testing license during a batch run. Use this button only if your batch contains both Service Test and QuickTest tests. **How does it work?** When you click **Run** (or select **Tests > Run**), Test Batch Runner instructs Service Test to use an HP Unified Functional Testing license. If it cannot use the license type, an error message opens informing you that if your run a test batch with both Service Test and QuickTest tests, they will fail. **Note:** If you click this button and Service Test begins using an HP Unified Functional Testing license, Service Test will continue to use this license for the duration of the batch run session. To clear this license, you must close the Test Batch Runner.

Test Batch Runner Toolbar

Many of the menu commands are also available by default from the toolbar, shown below:

In addition, the Service Test button is not present in the **File** or **Tests** menus. It is described below:

UI Element	Description
UFT	**Use HP Unified Functional Testing license.** Instructs Test Batch Runner to use a Unified Functional Testing license during a batch run. Use this button only if your batch contains both Service Test and QuickTest tests. **How does it work?** When you click **Run** (or select **Tests > Run**), Test Batch Runner instructs Service Test to use an HP Unified Functional Testing license. If it cannot use the license type, an error message opens informing you that if your run a test batch with both Service Test and QuickTest tests, they will fail. **Note:** If you click this button and Service Test begins using an HP Unified Functional Testing license, Service Test will continue to use this license for the duration of the batch run session. To clear this license, you must close the Test Batch Runner.

Troubleshooting and Limitations - Run Sessions

This section describes troubleshooting and limitations related to running tests.

Service Test run sessions

- If you want to run Service Test in a minimized RDP (remote desktop protocol) session, and you are using an RDP 6.0 or later client, you can enable it by setting a registry value on the computer that is running the RDP client:

  a. If it does not exist, create the `RemoteDesktop_SuppressWhenMinimized` registry value (DWORD type) in one of the following registry paths on the computer that is running the RDP client:

  - **For 32-bit operating systems:** `<HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE>\Software\Microsoft\Terminal Server Client`
  - **For 64-bit operating systems:** `<HKEY_CURRENT_USER>\Software\Wow6432Node\Microsoft\Terminal Server Client`

  b. Set the data for this value to 2.

  > **Note:** You must restart your remote session in order for this setting to take effect.

- When running Service Test on a remote machine using a Remote Desktop Connection session (RDC) or using Citrix, if the computer on which the application is being tested is logged off or locked, the following problems may occur:
  - The test or component run session may fail.
  - Steps that contain keyboard or focus operations may fail.
  - The Run Results still image capture and/or the Screen Recorder may display a black screen.
  - Steps for which the device level replay is configured to use the mouse (instead of browser events) to run mouse operations may fail. (You set the device level replay using a **Setting.WebPackage("ReplayType")** statement or by setting the **Replay type** option in the Advanced Web Options dialog box.)

  Workaround: If you are using Citrix or a Remote Desktop Connection session to run a test or component, make sure that the computer on which the application is being tested is not logged off or locked.

- When running Service Test tests or components on a local machine, if the computer on which the application is being tested is locked, your test run may fail.

  Workaround: Install Service Test on a virtual machine (without a screensaver or lock password), and start or schedule your run session on the virtual machine. Then you can lock your local computer without locking the virtual machine.

- It is not recommended to use Test Batch Runner with the UAC (User Account Control) feature set to `ON`.

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.

- Running tests on remote machines using shared folders, may require adjusting the .NET 2.0 security settings.

  Suggestion: Open the **Control Panel** and locate the **Administrative Tools**, either by browsing or through a search. In the list of **Administrative Tools**, look for the following entry: `Microsoft .NET Framework 2.0 Configuration`. If it is not present, you must install the .NET Framework 2.0 SDK.

- The Validate Structure checkpoint fails if the expected value is a SOAP Fault and the Web Service call returns an **UnsupportedMediaType** status.

Chapter 18: Debugging Tests and Components

This chapter includes:

Concepts

Debugging Overview

After you create an event handler or user code file, you should check that it runs smoothly, without errors in syntax or logic.

By controlling and debugging your run sessions, you can identify and handle problems in your event handlers or user code files.

Service Test provides different options that you can use to detect and isolate defects in anuser code file. For example:

- You can control the run session using breakpoints and various step commands that enable you to step into, over, and out of a specific step.
- When a run session is suspended, you can use the Debug panes to check and modify the values of code objects and variables and to manually run script or code commands.

Considerations for Debugging User Code Files

- In order to use the debugging features, you must enable the debugger in the API Testing **General** pane of the Options dialog box (**Tools > Options > API Testing** tab **> General** node). Select the **Run test in debugging mode** check box. For details, see "General Pane (Options Dialog Box > API Testing Tab)" on page 271.
- While tests are running in debug mode, they are read-only. You can modify user code files after you stop the debug session (not when you pause it).
- When you open a test saved on an ALM project, Service Test creates a local copy of the external references that are saved to your ALM project. Therefore, any changes you apply to any external reference that is saved in your ALM project, such as a WSDL file, will not be recognized in the test or component until the test or component is closed and reopened. (An external reference is any resource that can be saved separately from the test or component, such as a WSDL file, XML file, or data table.)

Single Step Commands

You can run a single step or step-by-step in a user code file by using the **Step Into**, **Step Out**, and **Step Over** commands.

Step Into

Step Into runs only the current step in the active user code file.

Step Out

After using **Step Into** to enter a step in a user code file, you can use the **Step Out** command. **Step Out** continues the run to the end of the user code file, returns to the calling test and then pauses the run session at the next line (if one exists).

Step Over

Step Over runs only the current step in the active user code file.

If the current step calls a user-defined function, the called function is executed in its entirety, but the called function script is not displayed in the document pane. The run session then returns to the calling user code file and pauses at the next step (if one exists).

For task details, see "Step into, out of, or over a specific step during a debug session" on page 311.

Watching the Values of Variables and Properties of Objects During a Run Session

You can use the Watch pane and Local Variables pane to view the current value of different code expressions, variables, and object properties in a suspended run session of your test or component. A run session is suspended, for example, if you use the **Run > Pause** command, or when it stops at a breakpoint.

The Local Variables pane displays the current values and types of all variables in the main script of the current action, or in a selected function in your user code files.

The Watch pane displays the current values and the types of code expressions and objects that you add to the pane.

As you continue stepping through the subsequent steps in your user code file, Service Test automatically updates the Watch pane and Local Variables pane with the current value for any variable or expression whose value changes. In addition, Service Test reevaluates the information displayed in the Watch pane and Local Variables pane as you make changes in the context of your debug session, as selected in the Call Stack pane,

You can add any of the following types of expressions to the Watch pane:

- The name of a variable
- The name of a property
- Any other type of code expression

> **Caution:** Service Test runs the expressions in the Watch pane to evaluate them. Therefore, do not add a method or any expression whose evaluation could affect the state of the test, as this can lead to unexpected behavior of your user code file.

Expressions added to the Watch pane are saved with the test and updated accordingly as you make changes to your test.

For task details, see "Check and modify the values of variables and code expressions during a debug session" on page 312.

Breakpoints

You can use breakpoints to instruct Service Test to pause a run session at a predetermined place in anuser code file. Service Test pauses the run when it reaches the breakpoint, before executing the

step. You can then examine the effects of the run up to the breakpoint, make any necessary changes, and continue running the user code file from the breakpoint.

Breakpoints are saved with your user code file and are maintained even after you close and reopen Service Test.

You can use breakpoints to:

- Suspend a run session and inspect the state of your application
- Mark a point from which to begin stepping through anuser code file using the "Single Step Commands" on page 308

For task details, see "Use breakpoints in your user code file" on next page.

Enabling and Disabling Breakpoints

You can instruct Service Test to ignore an existing breakpoint during a debug session by temporarily disabling the breakpoint. Then, when you run your user code file, Service Test runs the step containing the breakpoint, instead of stopping at it. When you enable the breakpoint again, Service Test pauses there during the next run. This is particularly useful if your user code file contains many steps or events, and you want to debug a specific part of it.

You can enable or disable breakpoints individually or all at once. For example, suppose you add breakpoints to various steps throughout your user code file, but for now, you want to debug only a specific part of it or a specific event. You could disable all breakpoints in your user code file and then enable breakpoints only for specific steps or in specific event handlers. After you finish debugging that section, you could disable the enabled breakpoints, and then enable the next set of breakpoints (in the section or event you want to debug). Because the breakpoints are disabled and not removed, you can find and enable any breakpoint, as needed.

Enabled breakpoint. An enabled breakpoint is indicated by a filled red circle icon () in the left margin adjacent to the selected step.

Disabled breakpoint. A disabled breakpoint is indicated by an empty circle icon () in the left margin adjacent to the selected step.

Tasks

How to Debug Your User Code File

This task describes different ways you can control and debug your run sessions so you can identify and handle problems in your user code files.

To practice this task, see "How to Debug a User Code File - Exercise" on page 314).

This task contains the following sections:

Prerequisite

To debug tests, you must enable the debugger. Select **Tools > Options > API Testing** tab **> General** node and select **Run test in debugging mode**.

Note: For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

Step into, out of, or over a specific step during a debug session

- To use the **Step Into** command select **Run > Step Into**, click the **Step Into** button  , or press **F11.**
- To use the **Step Out** command select **Run > Step Out**, click the **Step Out** button  , or press **SHIFT+F11**.
- To use the **Step Over** command select **Run > Step Over**, click the **Step Over** button  , or press **F10**.

For details, see "Single Step Commands" on page 308.

Use breakpoints in your user code file

For details, see "How to Use Breakpoints " on next page.

Check and modify the values of variables and code expressions during a debug session

- To add an expression to the Watch Pane, click the **Add New Watch Expression** button  in the Watch Pane and enter the name of the expression in the Add New Watch dialog box.
- To remove an expression from the "Watch Pane", select the row in the Watch pane that you want to remove and press the **DELETE** key on your keyboard or press the **Delete** button  .
- To view the current values for all variables up to the current step in the user code file, use the "Local Variables Pane".

You can also change the value of a variable or property manually by running code commands in the Console pane.

For more details, see:

- "Watch Pane" on page 126
- "Local Variables Pane" on page 122
- "Console Pane" on page 124

Manually run code commands during a debug session

For details, see "Console Pane" on page 124.

View the current call stacks

To view the currently running call stacks in your run session, select **View > Debug > Call Stack**. You can double-click on a stack name in the pane to navigate directly to the line of code that begins the call stack.

Note: If the location of the call stack is not in a currently open file, Service Test opens the relevant file.

View currently running threads

To view the code threads currently running in the run session, select **View > Debug > Threads**. You can double-click on a thread's name in the "Threads Pane " (described on page 120) to navigate directly to the beginning of the thread.

Note: If the file containing the beginning of the selected thread is not open, then Service Test opens the relevant file.

View the loaded modules associated with the run session

To view the associated loaded modules for your run session, select **View > Debug > Loaded Modules**.

How to Use Breakpoints

The following steps describe how to set breakpoints, and temporarily enable or disable them. After you finish using them, you can remove them from your user code file.

This task includes the following sections:

Set a breakpoint

To set a breakpoint, do one of the following:

- Click in the left margin of a step in the user code file where you want the run to stop.
- Select the line where you want the run to stop and select **Run > Insert/Remove Breakpoint**.

The breakpoint symbol is displayed in the left margin adjacent to the selected step.

Enable or disable a breakpoint

To enable/disable a specific breakpoint, do one of the following:

- Right-click the step containing the breakpoint and select **Enable/Disable Breakpoint**.
- In the " Breakpoints Pane", select the breakpoint you want to enable or disable and press the **Disable/enable breakpoint** button .

Enable or disable all breakpoints

To enable/disable all breakpoints, select **Run > Enable/Disable All Breakpoints.** If at least one breakpoint is enabled, Service Test disables all breakpoints in the document. Alternatively, if all breakpoints are disabled, Service Test enables them.

Remove a single breakpoint or all breakpoints

To remove a single breakpoint, click the breakpoint icon in the left margin of the step. The breakpoint symbol is removed from the left margin of the user code file.

To remove all breakpoints, do one of the following:

- Select **Run > Clear All Breakpoints.**
- In the " Breakpoints Pane", click the **Remove all** button .
- In the " Breakpoints Pane", right-click and select **Remove all**.

All breakpoint symbols are removed from the left margin of the user code file.

Navigate to a specific breakpoint

1. In the " Breakpoints Pane", select the specific breakpoint to which you want to navigate.
2. Do one of the following:
   - Double-click the line containing the breakpoint name.
   - Click the **Go to source** button .
   - Right-click the line containing the breakpoint and select **Go to Source**.

   The cursor flashes in the main document window in the line containing the breakpoint.

How to Debug a User Code File - Exercise

In this exercise, you create and debug a Service Test user code file to practice using some of Service Test's debugging capabilities for Service Test tests.

Note: For a task related to this scenario, see "How to Debug Your User Code File" on page 311.

This scenario includes the following steps:

1. ## Create test steps

   a. Create a test.

   b. From the "Toolbox Pane", drag the **Add** activity, **Multiply** activity, and the **Custom Code** activity to the canvas.

2. ## Set properties for the math steps

   a. In the **Input/Checkpoints** tab , in the Input pane, enter the values for the **Add4** operation.

      - In the **Value** column for **A**, enter `10`.
      - In the **Value** column for **B**, enter `6`.

   b. In the **Input/Checkpoints** tab, enter the values for the **Multiply** operation.

      - In the **Value** column for **A**, enter `2`.
      - In the **Value** column for **B**, enter `4`.

3. ## Create parameters for the Custom Code activity

   a. In the "Add Input/Output Property/Parameter Dialog Box" (described on page 497), enter the details for the input parameter.

      - In the **Name** field, enter `AddResult`.
      - In the **Type** field, select `String` from the drop-down list (if it is not already selected).

A new input property called **AddResult** appears in the **Input** pane inside the **Input/Checkpoints** tab.

b. Create another input parameter called **MulResult**. This time, in the "Add Input/Output Property/Parameter Dialog Box":

- In the **Name** field, enter `MulResult`.
- In the **Type** field, select `String` from the drop-down list (if it is not already selected).

A new input property called **MulResult** appears in the **Input** pane inside the **Input/Checkpoints** tab.

c. Click the **Add** button again and select **Add Output Property**.

d. In the "Add Input/Output Property/Parameter Dialog Box" (described on page 497), enter the details for the output parameter.

- In the **Name** field, enter `Result`.
- In the **Type** field select `Decimal` from the drop-down list.

e. In the **Checkpoints** pane, enter the value of `128`.

4. Link the Custom Code activity to existing steps

a. In the "Select Link Source Dialog Box" (described on page 630), select the **AddActivity** step. In the right pane, select **Result** and click **OK.**

The link source `Step.OutputProperties.AddActivity<number>.Result` appears in the **AddResult** row.

b. In the dialog, select the **Multiply** step. In the right pane, select **Result** and click **OK**.

The link source `Step.OutputProperties.MultiplyActivity<number>.Result` appears in the **MulResult**.

5. Create events for the Custom Code activity

a. In the Properties pane, select the **CustomCode** activity from the drop-down list or by clicking the **CustomCode** activity in the canvas.

b. In the Properties pane, select the **Events** tab.

c. In the "Events Tab (Properties Pane)" (described on page 193), create a default handler for **ExecuteEvent** and **AfterExecuteStepEvent**. Two events, `CodeActivity<number>_OnExecuteEvent` and `CodeActivity<number>_OnAfterExecuteEvent` are added to the `TestUserCode.cs` file.

For details on Service Test events, see "Writing Code for Events Overview" on page 642.

d. In the `TestUserCode.cs` file, enter the following text in the `CodeActivity_OnExecuteEvent:`

```
decimal AddResult = AddActivity.Result;
decimal MulResult = MultiplyActivity.Result;
CodeActivity<number>.Output.Result = AddResult*MulResult;
```

e. Enter a breakpoint on the last line of this method.

f. In the `TestUserCode.cs` file, enter the following text in the `CodeActivity<number>_OnAfterExecuteStepEvent`:

```
decimal result = CodeActivity<number>.Output.Result;
```

g. Enter a breakpoint on the last line of this method.

h. Save the test.

6. **Run the test**

Select **Run > Run** or press F5.

7. **Check the value of the variables at the first breakpoint**

a. When the test run stops at the first breakpoint, select **View > Debug > Local Variables** to open the "Local Variables Pane".

b. In the Local Variables pane, see the current values of the **Add** and **Multiply** activity. The current values should be `16` for the **AddActivity** and `8` for the **MultiplyActivity**.

You can expand the notes on various rows to see the variable values of the different items used in your test run.

8. **Add a variable to the Watch Pane**

a. In the **TestUsercode.cs** tab, highlight the text `AddResult`.

b. Open the Watch pane by selecting **View > Debug > Watch**.

c. In the "Watch Pane" (described on page 126), click the **Add New Watch Expression** button  and enter `Add Result`.

A line with the expression `AddResult`, with a value of `16`, and type `Decimal` appears.

d. Click the **Add New Watch Expression** button again to add the variable **MulResult** to the Watch pane. The pane should display the expression `MulResult`, with a value of `8`, and type `Decimal`.

9. **Check the value of the variables at the next breakpoint**

a. Continue the run session by selecting **Run > Continue** or pressing F5.

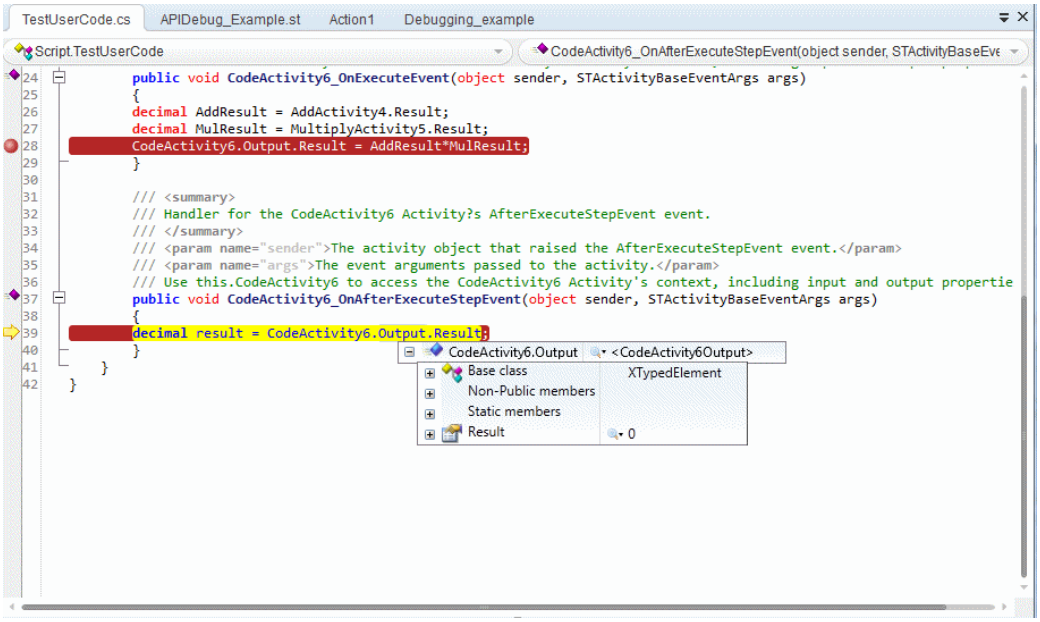
b. When the run session pauses at the next breakpoint, highlight the text `CodeActivity<number>.Output.Result` and add it to the Watch pane.

The pane displays that the value of this variable is `128`.

Note that the **AddResult** and **MulResult** values, which you added in the previous step to the Watch pane, are undefined with a type `Incorrect Expression`. This is because these values are present and relevant to the current event.

c. Click the Local Variables tab. Note that the line `Result` displays a value of `128`, since the custom code entered earlier noted that `Result` is equal to `CodeActivity6.Output.Result`, which was equal to `AddResult*MulResult`.

In addition, if you hover over the variable names in the Editor in the paused run session, an expandable tooltip displaying the current value of the variable and its properties can be viewed.



d. Select **Run > Continue** or click F5 to complete the run session and view the run results.

Reference

Run/Debug Session Toolbar

This toolbar is displayed when you begin a run or debug session.

To access	Start a run or debug session.
Important information	The toolbar buttons described are unique to the Run/Debug Session toolbar. Most of the toolbar buttons displayed during a run or debug session are common to the toolbar displayed when designing or editing a test or component.
Relevant tasks	"How to Debug Your User Code File" on page 311
See also	"How to Debug a User Code File - Exercise" on page 314

User interface elements are described below:

Toolbar button	Name	Description
	**Run**	Starts a run session or continues a paused run or debug session.
	**Stop**	Stops the current run session.
	**Pause**	Pauses the current run session.
	**Step Into**	Enables you to run only the current step in your user code file. For details, see "Step Into" on page 308.
	**Step Over**	Enables you to run only the current step in the user code file. For details, see "Step Over" on page 309.
	**Step Out**	Continues the current run session after the current step until the end of the current function. For details, see "Step Out" on page 308.
	**Solution Explorer Pane**	Displays the Solution Explorer pane as the active pane.
	**Toolbox Pane**	Displays the Toolbox pane as the active pane.
	**Data Pane**	Displays the Data pane as the active pane.
	**Properties Pane**	Displays the Properties pane as the active pane.

Toolbar button	Name	Description
	**Debug Pane**	Opens a drop-down list of the debug panes you can bring into focus. For details, see "Debug Pane Interface" on page 112.
	**Output Pane**	Displays the Output Pane as the active pane.
	**Errors Pane**	Displays the Errors pane as the active pane.
	**Tasks Pane**	Displays the Tasks pane as the active pane.
	**Active Screen**	Displays the Active Screen pane as the active pane.

Part 5: Service Test Integration With HP ALM

Chapter 19: ALM Integration

Note: Unless otherwise specified, references to **Application Lifecycle Management** or **ALM** in this guide apply to all currently supported versions of ALM and Quality Center. Note that some features and options may not be supported in the specific edition of ALM or Quality Center that you are using.

For a list of the supported versions of ALM or Quality Center, see the *HP Service Test Product Availability Matrix*, available from the Service Test Help or the root folder of the Service Test DVD. The most up-to-date product availability matrix is available from the HP Software Product Manuals site, at http://h20230.www2.hp.com/selfsolve/manuals (requires an HP Passport).

For details on ALM or Quality Center editions, see the *HP Application Lifecycle Management User Guide* or the *HP Quality Center User Guide*.

This chapter includes:

Concepts

ALM Integration Overview

Service Test integrates with ALM, the HP centralized quality solution. ALM helps you maintain a project of all kinds of tests (such as tests, components, business process tests, manual tests, tests created using other HP products, and so on) that cover all aspects of your application's functionality. Each test or component 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 in unique groups.

ALM provides an intuitive and efficient method for scheduling and running tests or components, collecting results, analyzing the results, and managing test and component versions. It also features a system for tracking defects, enabling you to monitor defects closely from initial detection until resolution.

At its most basic level, integrating Service Test with ALM enables you to store and access tests, components, and resource files in an ALM project, when Service Test is connected to ALM.

Service Test integrates with Service Test Management, ALM's extension for storing and managing application components, such as Web services. Service Test Management is an extension of ALM and it provides an efficient method for storing and retrieving application components.

This section also includes: "What is an ALM Project?" below

What is an ALM Project?

An ALM project is a database for collecting and storing data relevant to a testing process. For Service Test to access an ALM project, you must connect to the local or remote Web server where ALM is installed.

When Service Test is connected to ALM, you can:

- **Create tests, components, and resources and save them in your ALM project.**
- **View the contents of your tests.** This can help you decide if you want to run a test as part of a test set. Note that the Test Flow in ALM and the canvas in Service Test display only the actions that are run when the currently selected test runs. This means that if a nested action is commented out, for example, that action is not displayed in ALM or in the Service Test canvas. You can uncomment it in the Service Test Editor when needed.
- **Run your tests and components and view the results in ALM**.
- **Associate a test or a component with external files stored in the Test Resources module of an ALMproject.**
- **Associate external files for all tests or for a single test.**You can set the default activity repository location for your custom test activities in the Test Resources module in ALM.

  For details on specifying the default location for activity repositories, see "General Pane (Options Dialog Box > API Testing Tab)" on page 271.

- **Take advantage of all the features provided with the Resources and Dependencies model.** For details, see "Resources and Dependencies Model" on page 343.

For details on working with ALM, see the *HP Application Lifecycle Management User Guide*.

Required Access Permissions

You must have the following access permissions to use Service Test with ALM:

- Full read and write permissions to the ALM cache folder (located on the ALM client side)
- Full read and write permissions to the Service Test Add-in for ALM installation folder

Data Awareness in ALM

In ALM, you can upload and store your Microsoft Excel (`.xls`) test data resource files in the Test Resources module, enabling you to use the same data for multiple tests or components, or different data for each test run.

In ALM, you run **configuration** instead of standard tests. When working with data-driven tests in ALM, each **configuration** is a test that is set to run with a selected data resource file and optional data filter settings. The filter settings enable you to filter data by specified row numbers, by parameter text values, or both.

Note: You define and manage configurations only in ALM.

Example

Suppose you define three configurations in a single test to test an application that provides different solutions for Gold Card, Silver card, and Bronze Card users. You could use the same data resource for each configuration by filtering the rows or text of the data resource to match the relevant user type. Similarly, if your data is stored in various `.xls`or `.xlsx` files, you might want to run the same test using a different data source each time. You would do this by associating a different data source with each configuration.

By managing your data as a resource in ALM, you can see at a glance which data resource files are associated with a particular test or configuration, and which tests or configurations are using a specific data resource. For more details, see "Advantages of Data Awareness in ALM" on next page.

How Does Data Awareness in ALM Work?

When you create a test in ALM (or save a test to an ALM project from Service Test for the first time), a default **configuration** with the same name as the test is created simultaneously. You can view this configuration in the Configuration tab of the Test Plan module.

In the ALM Test Resources module, you create one or more data resources and upload one Microsoft Excel (`.xls`or `.xlsx`) file for each.

After you upload a data resource file to your ALM project, you can define the test-level configuration settings for a particular test. You do this in the Parameters tab of the Test Plan module by selecting

the data resource file and mapping its column names to the data table parameters in your test. You can associate one data resource with the test-level configuration. By mapping the column names to data table parameters, you enable Service Test to identify and use the correct parameter value during a run session.

> **Note:**
>
> When running a test, Service Test's Data Navigation looks at the data resulting from the specific configuration. For example, if your ALM configuration filters the data to use only rows 3 through 5, and the Service Test navigation says to iterate rows 2 through 3, the actual data used will be rows 4 and 5 from the test resource.

In the configuration tab of the Test Plan module, you can create as many additional configurations as needed. By default, every configuration uses the test-level configuration settings unless you override the data resource.

If you associate a configuration with the data resource defined in the Parameters tab, you do not need to map the data table parameters to column names. You can, however, modify the filter settings by applying text filters to any parameter, and/or selecting the rows on which the configuration can run.

If you associate a different data resource with a configuration, or you select to override the test data resource and then select the same data resource file that you selected in the Parameters tab (which the same as selecting a different data resource), you need to map the data table parameters to the column names in the `.xls` file. You can also apply filter settings, as described previously.

Next, in the Test Lab module, you can run any or all of the configurations defined for a test (or associated with a requirement) by adding them to a test set. When you run a configuration containing steps that use data table parameters, the parameter values are retrieved from the data resource files according to the settings defined for that configuration.

For a task describing how to work with configurations, see "How to Data Drive a Test Using Data from ALM" on page 328.

For a chart listing modules and tabs in which you perform tasks in ALM, see "ALM Data Awareness - Task Breakdown" on page 334.

This section also includes:

Advantages of Data Awareness in ALM

Test reuse. You can use the same test to test various scenarios, each time with a different set of data. You can do this associated a different data resource with each configuration, by associating a single data resource with different configurations and then filtering each configuration, or by using a combination of both.

Data Reuse. You can associate the same data resource with multiple tests or configurations.

Ease of Management. All of your resources are stored in one central location.

Visibility. You can see which tests are using a particular data table, and you can see which data table each test is using.

Requirements Coverage. You can cover all of your testing requirements by linking them to specific configurations. This enables you, for example, to use a single test to cover multiple requirements by associating different configurations in the same test with each requirement.

Resources and Dependencies Model. By storing your data in the Test Resources module, you can take advantage of additional ALM integration features, such as:

- Version control and baselines
- Asset comparison Tool and Asset Viewer
- Sharing data resources across ALM projects

Considerations for Data Awareness in ALM

Consider the following when managing your data resources in ALM:

- Only Microsoft Excel (`.xls` or `.xlsx`) files are supported as data resources.
- If you modify a data table parameter or a column name in the associated data resource after they are mapped to each other, you must update the mappings accordingly. Additionally, if you update an Excel data source and link properties to the new data source, the new values will not be reflected in ALM until you remap the properties using the Map Parameters dialog box.

  For details, see "How to Data Drive a Test Using Data from ALM" on page 328.

Tasks

How to Work with Tests and Components in ALM

The following steps describe the workflow of how to work with tests and components saved in an ALM project.

This task includes the following steps:

- "Prerequisites for Windows Vista, Windows 7,and Windows Server 2008, and Windows Server 2008 R2 Users" below
- "Connect to an ALM Project" below
- "Create or open a test or component" below
- "Run your test or component in the ALM project" below
- "Manage versions of your project using version control - optional" below
- "Disconnect from the ALM project" on next page

Prerequisites for Windows Vista, Windows 7,and Windows Server 2008, and Windows Server 2008 R2 Users

The security settings in the following operating systems may prevent you from connecting to an ALM 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 ALM project.

To connect to ALM for the first time, you must disable the UAC option and restart your computer. After you successfully connect to ALM, you can turn the UAC option on again. Thereafter, you should be able to connect to ALM, as needed.

Connect to an ALM Project

For details, see "ALM Connection Dialog Box" on page 335.

Create or open a test or component

For details, see "New <Document> Dialog Box" on page 67 or "Open/New <Document>/<Resource> Dialog Box" on page 69.

Run your test or component in the ALM project

For details, see"How to Run an API Test" on page 288 and "Run Dialog Box" on page 296.

Manage versions of your project using version control - optional

For details, see "Managing Versions of Assets in ALM Overview" on page 357.

Disconnect from the ALM project

For details, see "ALM Connection Dialog Box" on page 335.

Note: When you disconnect from a project, all open tests from the project automatically close.

How to Data Drive a Test Using Data from ALM

This task provides a general overview of the steps involved in data driving a test with data stored in ALM. After you are familiar with the steps, you can perform many of them in the order you choose. Some steps may not be necessary in all cases.

This task includes the following steps:

1. ### Prerequisites

   a. Connect to ALM, as described in "ALM Connection Dialog Box" on page 335.

   b. Make sure that your test is saved in your ALM project. For details on creating tests and saving them to ALM, see "New <Document> Dialog Box" on page 67

2. ### Import data into a test

   a. In the Data pane, click the **New Data Source** button  and select **Excel**.

   b. In the "New/Change Excel Data Source Dialog Box" (described on page 98), select the `.xls` or `.xlsx` file containing the data and Select the **Allow other tools to override the data** option. Click **OK** to import the data source into your test.

3. ### Data drive the test steps

   For details, see "How to Assign Data to Test Steps" on page 621.

4. ### Create a data resource file in your ALM project

   In the Test Resources module:

   a. Expand the Resources tree and select the required node.

   b. Select **Resources > New Resource** to add a resource under that node.

   c. In the New Resource dialog box:

- In the **Type** list, select **Data table**.
- In the **Name** box, enter a name for the data resource, for example, the name of the Microsoft Excel (`.xls`or `.xlsx`) file you plan to use.
- Fill in the remaining fields (optional) and click **OK** to close the dialog box.

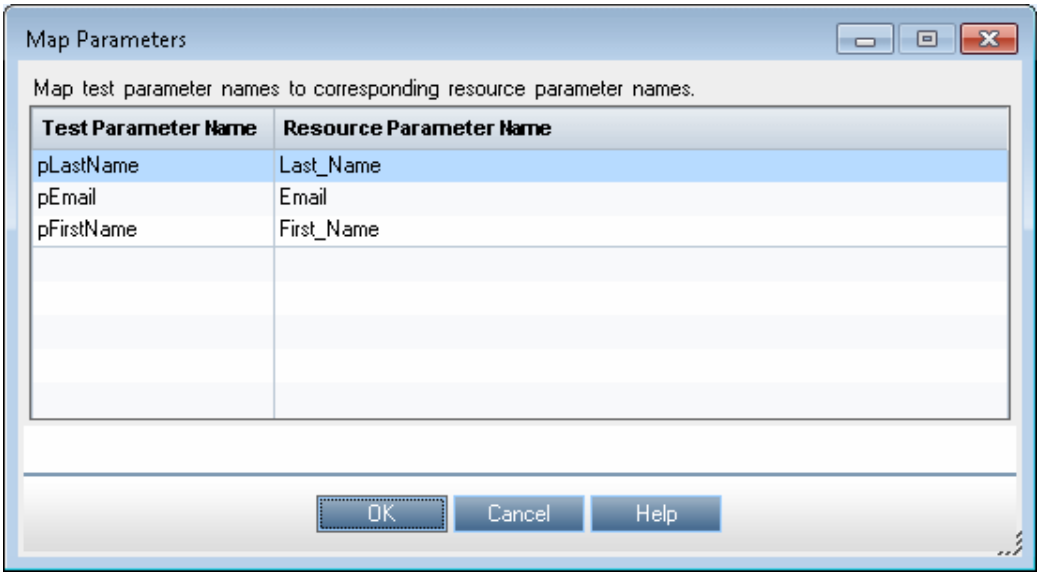
d. In the Resource Viewer tab, click **Upload File**. Then browse to and upload the relevant `.xls`or `.xlsx` file.

> **Note:** You can convert an internal data table from an open test to an uploadable data resource file by right-clicking the Data pane, selecting **File > Export**, saving the data table to the file system as an `.xls` or `.xlsx` file, and then uploading it as described above.

5. ## Specify a default data table resource for all new test configurations

a. In the Parameters tab of the Test Plan module, select the data table resource you want to use as the default for all test configurations.

b. Click the **Map Parameters** button  . In the Map Parameters dialog box, map the data table parameters (column headings) to the test parameters by entering the matching data table parameter names in the **Resource Parameter Name** column, as shown in the example below.



All new configurations use the default mappings unless you specify otherwise in the Data tab of the Configurations tab. For details, see the next step, **Define your test configurations**.

6. ## Define your test configurations

Define test configurations for various run sessions. For each configuration, you specify whether to use the default resource file that you specified in the previous step, or whether to use a different data resource file.

a. In the ALM Test Plan module, browse to and select the test to which you want to associate your data table resource. For details, see the *HP Application Lifecycle Management User Guide*.

b. Click the **Test Configurations** tab. A default configuration is displayed in the grid. This configuration was created when your test was added to the ALM project. For details on configurations, see "Data Awareness in ALM" on page 324.

c. In the bottom pane of the Configurations tab, click the **Data** tab.

d. In the Data tab:

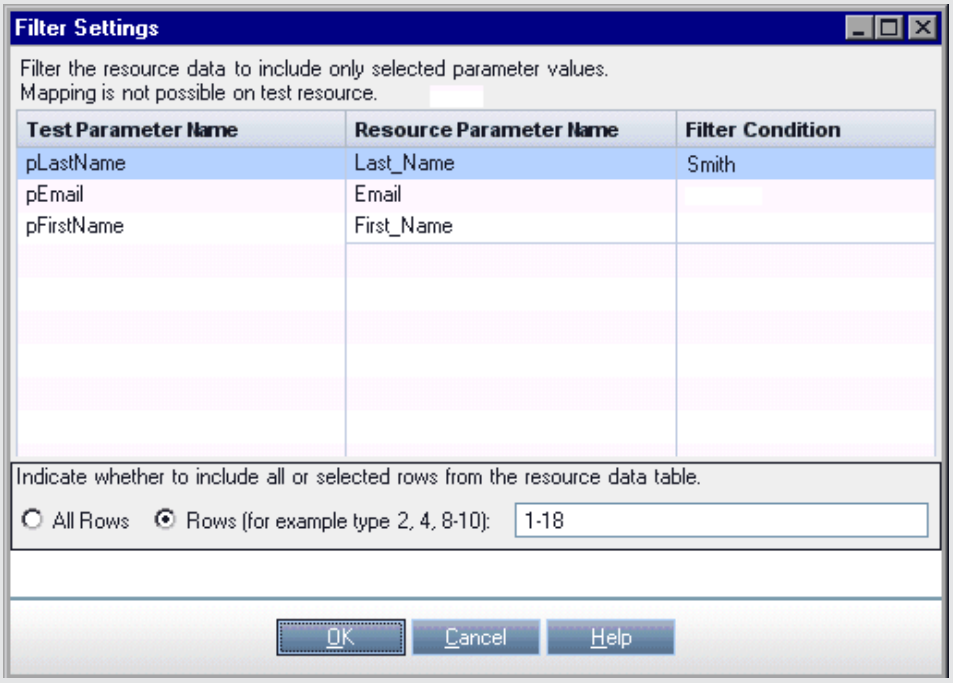
- Select the **Override test data resource** check box to select a different data resource file from the Test Resources module, or leave the check box blank to use the default resource file you selected in the Parameters tab in the previous step.
- In the **Data Resource** box, browse to and select the relevant data resource file to associate with this configuration. (Relevant only if you selected the **Override test data resource** check box)
- Click the **Data Resource Settings** button, and in the Filter Settings dialog box:
  - Map the data table parameters from your test to the column headers in the data table file (Relevant only if you selected a different data resource file in the previous step)
  - Apply filter conditions (text strings), as needed. You can apply one filter condition to each parameter.
  - Specify the rows on which to run iterations. For example, if you run a configuration named `Gold`, and users of this type are listed in rows 2-114, then specify these rows only.

Note: If you apply filter conditions and specify rows, AND logic is used, meaning that the parameter value must equal the filter text value and the parameter value must be located in one of the specified rows.

Example:

The `pLastName` data table parameter is mapped to `Last_Name` and is filtered to include only parameter text values that equal `Smith`. The configuration is set to run

only on rows 1-18 in the `.xls` or `.xlsx` file you uploaded as the data resource.



For details on this dialog box, see the *HP Application Lifecycle Management User Guide*.

7. ## Link your configurations to requirements to create requirements coverage - Optional

   If you want to make sure that your requirements are fully covered, link them to configurations. This enables you to select configurations to run based on requirement coverage when you plan your run session.

   a. In the Test Plan module, click the **Req Coverage** tab.

   b. Click the **Select Req** button. The Requirement Tree tab is displayed in the right pane.

   c. From the Requirement Tree tab, select a requirement to add to the Req Coverage grid. When you add the requirement, the Add Advanced Coverage dialog box opens.

   d. Select the test configurations that cover this requirement.

8. ## Run your test configuration

   You run configurations from ALM.

   a. In the ALM Test Lab module, select or create a test set. For details, see the *HP Application Lifecycle Management User Guide*.

   b. In the right pane, select the **Execution Grid** tab.

   c. Click the **Select Tests** button to display the **Test Plan Tree** and **Requirements Tree** tabs in the right pane.

   d. Do one of the following to select the configurations to run:

- From the Test Plan Tree tab, select a test to add to the Execution Grid. When you add the test, all of its configurations are added to the Execution Grid. (The test itself is not added to the Execution Grid because ALM runs configurations, not tests.)
- Below the Test Plan Tree tab, expand the **Test Configurations** pane, and add the specific configurations that you want to run to the Execution Grid.
- Below the Requirements Tree tab, expand the coverage pane, and select a test to add to the Execution Grid. When you add the test, all of its configurations are added to the Execution Grid. (The test itself is not added to the Execution Grid because ALM runs configurations, not tests.)

e. Click the **Run** button to run the selected configurations.

f. After the run session, click the **Launch Report** button in the Last Run Report tab to view the results.

How to Use ALM Version Control

The following steps describe how to use version control features to manage tests and resources stored in an ALM version-controlled project.

Prerequisite

Connect to your ALM project. Make sure that you are logged into a version controlled project. For details, see "ALM Connection Dialog Box" on page 335.

Check out a test

1. Open the test you want to check out. For details, see "Open/New <Document>/<Resource> Dialog Box" on page 69.

   A dialog opens asking if you want to check out the test. Select **Yes**.

2. In the Check Out dialog box, enter any comments you want to attach with your check out and click **OK**.

   An editable copy of the test opens and is locked in your ALM project.

Check in a test

1. In the document pane or in the Solution Explorer, select your test or resource file.
2. Select **ALM > Check In**. The "Check In Dialog Box" on page 365 opens.
3. In the Check In dialog box, enter any comments you want to attach to your check in and click **OK**.

The test or resource file is checked into your ALM project and is available for other users to check out and edit.

Cancel a check out operation

If you decide you do not want to upload the changes you made to an asset to your ALM project:

1. In the document pane or in the Solution Explorer, select your test or resource file.
2. Select **ALM > Undo Check Out**. Click **Yes** to confirm the cancellation of your check out operation.

   The check out operation is cancelled. The checked out asset closes, and the previously checked in version reopens in read-only mode.

View the version history

1. In the document pane or in the Solution Explorer, select the test or resource file for which you want to view the history.
2. Select **ALM > Version History**.

   The Version History dialog box opens, displaying the information on the versions of the selected asset.

Reference

ALM Data Awareness - Task Breakdown

The following table lists where to perform specific data awareness tasks in ALM:

Module	Tab	Task
Test Plan	**Parameters**	Define the test-level configuration for the current test: • Specify a data resource file. • Map the data table parameters used in your test steps to the column names in your data resource file.

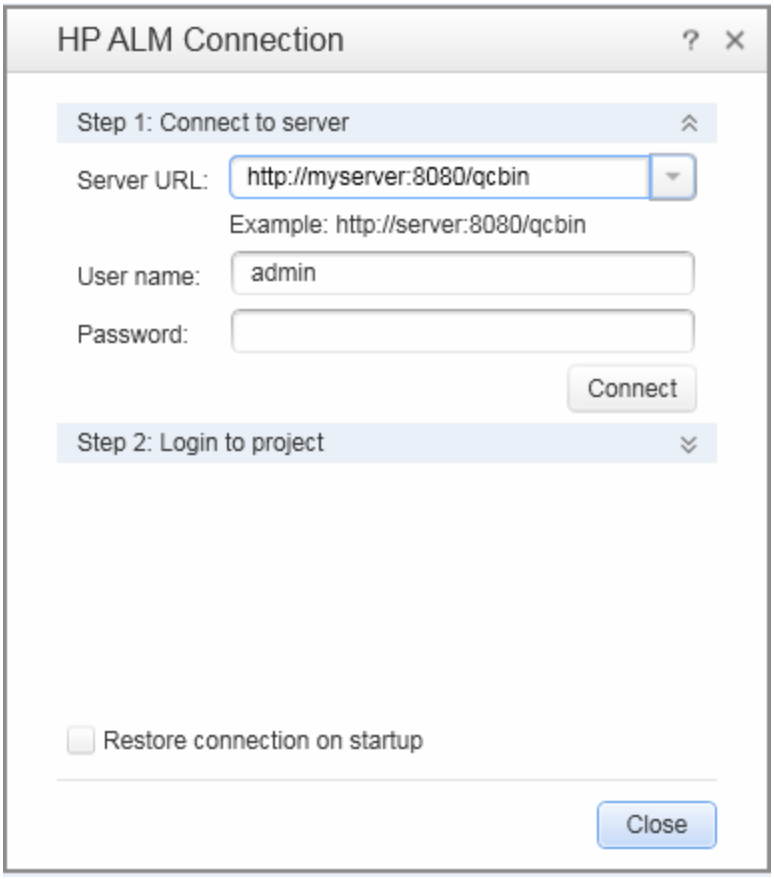
	Test Configurations	Create additional configurations. (Optional)
		Data tab:
		• Select a different data resource. (Optional)
		• Specify the data resource settings:
		  ▪ Specify the rows to be used during a run session.
		  ▪ Specify any text parameters values filters. (Optional)
		  ▪ If you select a different data resource (by overriding the data resource defined in the Parameters tab), map the data table parameters used in your test steps to the column names in your data resource file.

	Req Coverage	If you add requirement coverage to a test that contains multiple configurations, specify which configuration(s) to use for coverage. (Can also be done from the Requirements module)
Test Resources	**Resource Viewer**	• Create a data resource. • Upload a Microsoft Excel (`.xls` or `.xlsx`) file to use as the data resource.
Test Lab	**Execution Grid**	• Add all of the configurations in a test to a test set. • Add only specific configurations to a test set. • Add all of the configurations contained in a tests that cover specific requirements to a test set. • Add only the configurations associated with a specific requirement to a test set.

ALM Connection Dialog Box

This dialog box enables you to connect to or disconnect from a project in any supported version of ALM.

For more details, see "ALM Integration Overview" on page 323.



To access	Use one of the following: • **In Service Test:** Select **ALM > ALM Connection**. • **In the Run Results Viewer:** Select **Tools > ALM Connection**. • Click the **ALM** toolbar button .

Important information	• **1st time connection.** The first time you connect to an ALM server, you must connect as a user with administrator privileges on the computer on which you are connecting. • **Connecting to different ALM servers.** You can simultaneously connect your Web browser to multiple ALM clients and one Quality Center 10.00 client simultaneously. While these clients are open, you can connect to the Quality Center 10.00 client that is currently open on your computer. However, if you want to connect to an ALM client, you must first close the Quality Center 10.00 client. • **Connecting to different versions of Quality Center or ALM.** You cannot connect to multiple versions of Quality Center or ALM in the same Service Test session. Close and reopen Service Test to connect to a different version of Quality Center or ALM. • **Windows Vista, Windows 7, Windows Server 2008, Windows Server 2008 R2 Users, Windows 8, and Windows Server 2012 Users.** See "Guidelines for Windows Vista, Windows 7, Server 2008, Server 2008 R2, Windows 8, and Windows Server 2012 Users" on page 338. • **Connect.** The connection process has two steps. First, you connect to a local or remote ALM server. This server handles the connections between Service Test and the ALM project. You must provide a user name and a password. Next, you choose the project you want to access. The project stores tests and run session information for the application you are testing. • **Disconnect.** You can disconnect from an ALM project or from an ALM server. ▪ If you disconnect from an ALM server without first disconnecting from a project, the Service Test connection to that project database is automatically disconnected. ▪ After you open a Business Process test in Service Test, you cannot log out of a project and log in to another project on the same server. You must first disconnect from the server and reconnect before logging in to another project. • **SSL Certificates.** If you are attempting to connect to an ALM project with a `https://` prefix, but your computer does not have a valid SSL certificate, the connection will fail. If an ALM test or shared file (such as a shared object repository or data table file) is open when you disconnect from ALM, then Service Test closes it in the document pane. The document remains in the Solution Explorer and is marked as unloaded.
Relevant tasks	To view the current connection: • **In Service Test:** Point to the icon on the status bar. A tooltip displays the server name and project to which Service Test is connected. • **In the Run Results Viewer:** The icon on the status bar is labeled with the server name and project to which the Run Results Viewer is connected.

User interface elements are described below:

UI Elements (A-Z)	Description
Close	Closes the ALM Connection dialog box. **Note:** The dialog box does not close automatically. You must click this button to close the dialog box.
Connect / Disconnect	Connects to or disconnects from the selected ALM server. **Note:** After you successfully connect to a server, the button changes to **Disconnect**, and at the top of the dialog box, the **Disconnected** icon  changes to a **Connected** icon  .
Domain	The domain that contains the ALM project.
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**, and the **Logged out** icon  changes to a **Logged in** icon  . • **Logout.** Logs out of the selected domain and project. **Note:** You must click **Close** to close the dialog box after logging into or out of a project.
Password	Your ALM password. **Note:** To enter a password in any CJK (Chinese, Japanese, Korean) language, copy/paste the password into the edit box. (Windows does not support typed CJK characters in password fields.)
Project	The ALM project with which you want to work. **Note:** Only those projects for which you are a defined user are displayed.
Restore connection on startup	Instructs Service Test to automatically reconnect to the ALM server every time you open Service Test.

UI Elements (A-Z)	Description
Server URL	The URL address of the Web server where ALM is installed. You can choose a server that is accessible via a Local Area Network (LAN) or a Wide Area Network (WAN). You can connect to any currently supported version of ALM. For a list of supported versions, see the *HP Service Test Product Availability Matrix*, available from the root folder of the Service Test DVD.
User Name	Your ALM user name.

Guidelines for Windows Vista, Windows 7, Server 2008, Server 2008 R2, Windows 8, and Windows Server 2012 Users

The security settings in the following operating systems may prevent you from connecting to an ALM project:

- Windows Vista
- Windows 7
- Windows Server 2008
- Windows Server 2008 R2
- Windows 8
- Windows Server 2012

This can occur when the UAC (User Account Control) option is set to ON, and you have never connected to an ALM project.

To connect to ALM for the first time, you must disable the UAC option. After you successfully connect to ALM, you can turn the UAC option on again. Thereafter, you should be able to connect to ALM, as needed.

Troubleshooting and Limitations - ALM Integration

This section describes troubleshooting and limitations for working with tests and components and ALM, and includes:

Troubleshooting and Limitations - General ALM Integration

- When working with Quality Center 10.00, set the DEP (Data Execution Prevention) flag to **off**.
- The first time you connect to your ALM server, (either within Service Test or through a browser) connect as administrator. This allows the machine to properly install the ALM client with the required connectivity. For all subsequent connections, you do not need to log in as administrator. This step is also required after installing ALM patches.
- If Service Test is installed on one of the operating systems listed in the table below, you must perform prerequisites before you run tests remotely from ALM:

Operating System	What you should do
▪ Windows XP Service Pack 2 ▪ Windows 2003 Server	You must enable COM+ access. For details, see the Installation Guide.

Operating System	What you should do
▪ Windows Vista ▪ Windows Server 2008 ▪ Windows 7 ▪ Windows Server 2008 R2 ▪ Windows 8 ▪ Windows Server 2012	a. You must enable COM+ access. For details, see the Installation Guide. b. Change DCOM permissions and open firewall ports. For details, see the Installation Guide. c. Run `RmtAgentFix.exe` from the `<Service Test installation>\bin` folder, or use the Additional Requirements Utility, which you open by selecting **Start > Programs > HP Software > HP Service Test > Tools > Additional Installation Requirements**. This is due to a problem with opening DCOM permissions on Windows Vista, Windows Server 2008, Windows 7 and Windows Server 2008 R2. d. Disable User Account Control (UAC) in Windows and restart your computer before you first connect to ALM.

- The security settings in Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, and Windows 2012 may prevent you from performing a Service Test related installation, such as a patch installation, or connecting to an ALM project (either directly or from Service Test). This can occur when the UAC (User Account Control) option is set to ON, and you have not yet connected to an ALM project (if relevant).

  Workaround: Temporarily turned off the UAC option.

  After disabling the UAC option as described above, perform the required installation or connect to ALM as usual. When you are finished, you can turn the User Account Control (UAC) option on again. Hereafter, you should be able to connect to ALM, as needed.

- If you are connected to an ALM server, and you want to connect to a different server, disconnect from the first ALM server, restart Service Test, and connect to the second server.

- If Service Test closes unexpectedly while an asset is open from ALM, the asset may remained locked by ALM for more than fifteen minutes. In some cases, you may be able to reopen Service Test and reopen the test, but when trying to save it, you will receive an error message indicating that the test entity is already locked by you.

  Workaround: Wait fifteen minutes or more and try again.

- Renaming a test or component from ALM may cause the test or component to behave unexpectedly.

  Workaround: To rename a test or component, open it in Service Test and rename it using the **Save As** option. If the test or component has already been renamed from ALM, use the rename option again to restore the old name, and then use the **Save As** option in Service Test. Renaming a test parameter from Service Test causes any run-time parameter values already set for this parameter in ALM to be lost.

- If an ALM user manually changes the status of a test instance run, ALM creates **fast run** results

to record the change of the test status. The **fast run** results are not valid run results files. However, when you try to select results to open or delete in the Run Results Viewer or Run Results Deletion tool, the **fast run** results are available in the list.

- For tests or business components saved on ALM, if you perform a checkout from within ALM, it is not reflected in Service Test for the current test—the test or business component still appears to be checked in.

  Workaround: Close and reopen the test in Service Test.

- When running a test from Service Test with ALM 11.50 and later with an ALM project that supports versioning, the test instance run status will not be updated for tests that are checked out.

- ALM supports non-Unicode projects. If you are working with an ALM project that is not Unicode compliant:

  - You should not use Unicode values (such as the name of the test or component, the default value of a test, action, or component parameter, method argument values, and so forth).
  - Data that is sent to Service Test from ALM (such as values for test, action, or component parameters) is not Unicode compliant.
  - Service Test results containing Unicode characters may appear corrupted in the ALM result grid. You can, however, open and view results containing Unicode characters in the Service TestRun Results Viewer.

- 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 the following location on the ALM client machine: `http:\\<ALM Platform server name>[<:port number>]/qcbin/Apps/`

  b. Click on the **Webgate Customization Tool** link and select **Run**.

  c. 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.

  d. Click **Save** and then **Close**.

- To run Service Test*locally* on a Windows 2003 Server machine, you must perform the following steps in order to run a test stored in ALM.

  a. Select **Start > Run**.

  b. In the Run dialog box, type **`dcomcnfg`**. The **Component Service** console opens.

  c. If the left pane, expand the **Component Services** tree. Select the **Computers > My Computer > COM+ Applications** node.

  d. In the right pane, select **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 required only if ALM is not installed on the same server as Service Test.

- To run Service Test remotely on a Windows 2003 server machine, you must enable network COM+ access. To enable this access, perform these steps on the remote server machine:

  a. Select **Start > Settings > Control Panel**.

  b. In the Control Panel, double-click **Add or Remove Programs**.

  c. In the Add or Remove Programs dialog box, click **Add/Remove Windows Components** in the left pane.

  d. Select **Application Server** and click the **Details** button.

  e. In the **Subcomponents of Application Server** pane, select **Enable network COM+ access**. Click **OK**.

  f. Click **Next** and **Finish** to close all of the dialog boxes. Restart the server machine.

  g. Perform the steps described in the previous bullet for running Service Test locally on a Windows 2003 server machine.

- When running tests with the Remote Agent, the Remote Agent process continues to run after the completion of the test run.

  Workaround: Manually end the `dllhost.exe` process in the Task Manager.

Troubleshooting and Limitations - ALM Integration

- While connected to an ALM project, if you change the available license for an Service Test test, ALM continues to use the original license.

  Workaround: Log out from your ALM project and restart Internet Explorer.

- Service Testconnections to ALM do not support NTLM authentication.

- For Service Test Management: Before importing a WSDL from Service Test Management, you must connect at least once to the Service Test Management-ALM server through Internet Explorer.

- To enable test versioning in Quality Center version 10.00, 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>
```

Chapter 20: Resources and Dependencies Model

Relevant for: GUI tests and components and API testing

Note: The references to ALM in this chapter apply to Quality Center 10.00 and ALM. Note that some features and options may not be supported in the Quality Center or ALM edition you are using. For information on Quality Center or ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

Resources and Dependencies Model Overview

Relevant for: GUI tests and components and API testing

Service Test enables you to use the Resources and Dependencies model to fully integrate your tests and components into ALM projects.

Note: Before you read this section, you may want to familiarize yourself with the "Resources and Dependencies Model Terminology" (described on 346).

- **Replaces the use of attachments with linked assets.** For example, **GUI tests, actions, and application areas** can be linked with function libraries and shared object repositories, respectively or **APItests** can be linked with data tables, user code files, or activities. You store your tests or components in the Test Plan or Business Components module, respectively, and you store your resource files (including application areas) in the Test Resources module. When you associate a resource file to a test or a GUI component's application area, these assets become linked. Linking assets improves runtime performance by decreasing download time. (Using attachments instead of resources increases download time from Quality Center 10.00 and ALM.) Linking assets also helps to ensure that the relationships between dependent assets are maintained.
- **Supports versioning for tests or components and resource files.** You can create versions of these assets in Service Test or in ALM, and you can manage asset versions in ALM. For details, see "Version Control in ALM" on page 356.
- **Supports baselines for tests or components and resource files.** You can view baseline history in Service Test, and you can view and manage baselines in ALM. For details, see "Baseline History Dialog Box" on page 368.
- **Enables you to import and share assets across ALM projects.** For details, see the *HP Application Lifecycle Management User Guide*.

For details about the advantages of working with this model, see "Asset Dependencies - Advantages" below.

Asset Dependencies - Advantages

Relevant for: GUI tests and components and API testing

When you associate a **dependent** resource file with a test's or a GUI component's application area, the assets become integrally linked, and these links can be viewed in ALM (in the Dependencies tab of various modules) and—for external GUI actions—in Service Test (in the Action Properties dialog box).

Assets stay linked

When you move a test, or rename a test, action, component, application area, folder, or resource, dependent assets are automatically updated to reflect the change. This helps ensure that there are no missing resources during a run session.

Resource files are all stored in one ALM module

Resource files are stored in the Test Resources module, enabling you to manage your resources in one central location, and to view at a glance which tests and application areas are using each resource file. For details on the Test Resources module, see the *HP Application Lifecycle Management User Guide*.

Using resources stored in the Test Resources module improves runtime performance

Tests or components open and run faster when the associated resource files are stored in the Test Resources module (instead of being stored as attachments to tests in the Test Plan module).

Version control can also be applied to resource files

When version control is enabled for a project, all of the assets can be checked into the version control database. For details, see "Version Control in ALM" on page 356.

You can create, view, compare, and run baselines

In ALM, you can create baselines that capture the developmental stage for each asset, view and compare these read-only baseline "snapshots", and run baselines from a project. In Service Test, you can view and compare baselines. For details, see "Baseline History Dialog Box" on page 368.

You can share assets with other projects and synchronize them

You can copy assets from other projects. This enables you to reuse your existing assets instead of creating new assets whenever you create a new project. For example, you can create a "template" set of assets to use as a basis for new projects.

You can synchronize these assets in both projects when changes are made, or you can customize your assets to suit the unique needs of each development project. For details, see the sections on importing and synchronizing libraries in the *HP Application Lifecycle Management User Guide*.

Deleting assets is easier

When you delete an asset (a reusable GUI action or component or associated resource file), a warning message informs you if the asset is used by other tests (or more than once in the current test) or is associated with an application area. This message contains a **Details** section that lists the tests or application areas that are associated with this asset or contain calls to this action so you can modify the tests or application areas, as needed. This helps you manage your business process tests and GUI action calls so that tests do not inadvertently fail.

You can verify which tests or components are associated with specific resources and vice versa

In the ALM Test Plan module and Business Components module, you can highlight a test or component and, in the Dependencies tab, see which assets are using the test or component, and see which assets the test or component is using. Similarly, in the ALM Test Resources module, you can highlight a resource file and see the assets with which it is associated.

For details, see "Dependencies Tab (ALM Modules) " on page 350.

Reference

Relative Paths and ALM

Relevant for: GUI tests and components and API testing

Resource files and GUI actions that are associated with tests or application areas using a relative path are not considered dependencies. To ensure that your resource files are recognized as dependencies, they must be saved in the Test Resources module in ALM, and they must be associated using the full ALM path. This enables you to benefit from the features provided by the Resources and Dependencies Model, as described in "Asset Dependencies - Advantages" on page 344.

For GUI testing: Despite this, there may be cases in which you may want to use a relative path. For example, if your application is released in different languages, you may want to use a relative path when associating shared object repositories with your tests or components' application areas, as this enables you to use the same test with localized shared object repositories.

Limitations - Relative Paths and ALM

- Run-time performance times are slower.
- Dependency information for these assets is not displayed in:
  - The Using and Used By grids in the Dependencies tab in ALM.
    See: "Dependencies Tab (ALM Modules) " on page 350
  - The message box that opens when you delete an asset that is associated with other assets.For details, see "Asset Dependencies - Advantages" on page 344
- ALM does not verify that these assets are included during the baseline verification process. For details, see "Viewing Baseline History" on page 360
- When opening or running tests or components from a baseline, any associated external GUI action or resource file that is associated via a relative path but is **not** included in the baseline is considered a missing resource. This may cause a test or component run to fail. (Note that the baseline version of an associated asset is used if the asset associated via a relative path **is** included in the baseline.), For details, see: "Viewing Baseline History" on page 360

Resources and Dependencies Model Terminology

Relevant for: GUI tests and components and API testing

Term	Description
Asset	Any testing document or resource file, including: **For GUI testing:** • tests • actions (GUI testing only) • components • application areas (GUI testing only) • function libraries (GUI testing only) • shared object repositories (GUI testing only) • recovery scenarios (GUI testing only) • data table files • environment variable files **For API testing:** • tests • components • data table files • XML files • testing activities • Web references (done via Service Test Management) • user code files **Note:** In ALM, Service Test assets are referred to by the more general term **entities**.

Term	Description
Resource	Any asset stored in the ALM Test Resources module that is used by a test or component. For example, a **GUI test or component** may contain calls to functions in associated function libraries, and it may reference test objects stored in shared object repositories that are associated with the test or component's application area. An **API test or component** may contain references to a data table file or use a custom testing activity. GUI testing resources include: • application areas • function libraries • shared object repositories • recovery scenarios • data table files • environment variable files API testing resources include: • data table files • XML files • testing activities • Web references (done via Service Test Management) • user code files **Note:** In some cases, a resource can be used by another resource. For example, a GUI test's recovery scenario can use functions in a function library.

Term	Description

Dependency	The linked relationship between a resource or external GUI action and a particular test or application area. Associated resource files and GUI actions are linked to each test or GUI component's application area that uses these resources or calls these GUI actions.
	
	In some cases, a resource can also be linked to another resource. For example, a GUI test's recovery scenario can call functions in a function library.
	
	Assets are considered dependencies if they are associated using absolute paths, and they are stored in the following modules:
	
	• **Test Plan module:** tests
	• **Business Components module:** components
	• **Test Resources module:**
	  ▪ GUI testing resources like application areas, function libraries, shared object repositories, recovery scenarios, data table files, environment variable files
	  ▪ API testing resources like data table files, `.xml` files, testing activities, and user code files
	
	Note: Tests or components stored in the **Unattached or Obsolete** folders in the Test Plan or Business Components module, respectively, are not considered dependencies because they are not associated with any test or component.

Configuration	A test that is set to run from ALM with an optional data resource file and optional data filter settings. In ALM, you can define various configurations for the same test or business process test. For tests, see "Data Awareness in ALM" on page 324 For components, see the *HP Business Process Testing User Guide*.

ALM Resources-Related User Interface

Relevant for: GUI tests and components and API testing

When you create an ALM project in your ALM server, the tests or components that you create in this project are saved to the Test Plan or Business Components module, respectively. You save your resource files to the Test Resources module. By associating resource files with your tests or GUI components' application areas, they become linked dependencies.

This section provides a general overview of the tabs that are relevant for tests, components, or applications areas. For details on using any of these tabs, see the relevant section in the *HP Application Lifecycle Management User Guide*.

This section includes (in alphabetical order):

Dependencies Tab (ALM Modules)

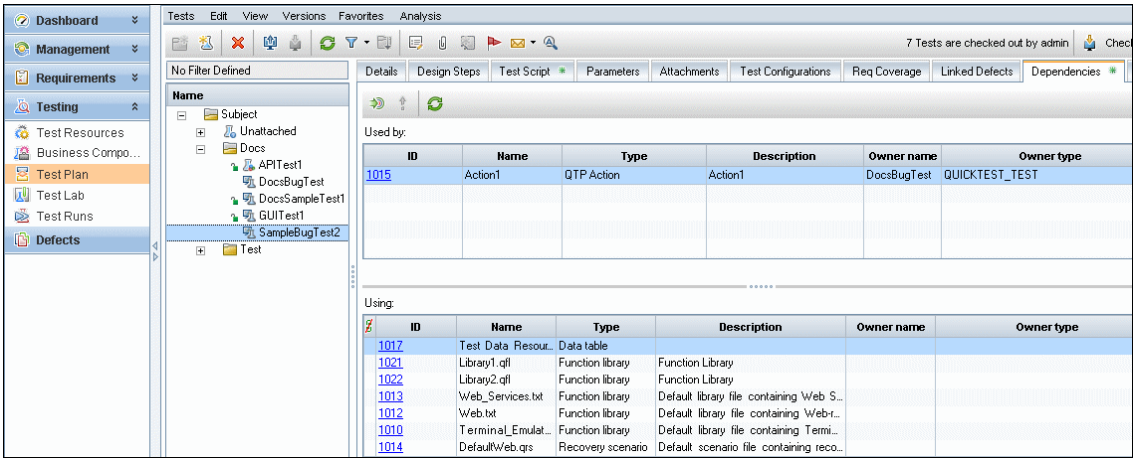
Relevant for: GUI tests and components and API testing

This tab displays the relationship between a selected asset, such as a test or component, and the assets with which it is associated. You use the Dependencies tab to see at a glance which resources are used by a particular asset, and which asset is using a particular resource. This information is displayed in the "Using Grid" on page 352 and "Used By Grid" on next page in the Dependencies tab in the Test Plan and Test Resources modules or the Business Components module.

Usage Example

For GUI testing: Suppose you want to modify the objects in a shared object repository. You can navigate to the shared object repository in the Test Resources module to view a list of the tests or application areas with which it is associated. This enables you to determine which assets this resource file is used by and helps you to analyze the impact that a proposed change may make to the dependent assets.

For API testing: Suppose you want to modify the values in a data table. You navigate to the data table in the Test Resources module to view a list of the tests or components with which it is associated. This enables you to determine which assets this resource file is used by and helps you to analyze the impact that a proposed change may make to the dependent assets.



To access	In ALM, the Dependencies tab is available from the following modules: • Business Components module • Test Plan module • Test Resources module

Important information	For details on using this tab, see the *HP Application Lifecycle Management User Guide*.
Relevant Tasks	• In the Business Components module, the Dependencies contains the following grids: ▪ **Used By.** Displays the business process tests that include this component. ▪ **Using.** Displays the application area with which the component is associated. For more details on the Dependencies tab in the Business Components module, see the *HP Business Process Testing User Guide*.
Business Process Testing	In the Business Components module, the Dependencies contains the following grids: • **Used By.** Displays the business process tests that include this component. • **Using.** Displays the application area with which the component is associated. For more details on the Dependencies tab in the Business Components module, see the *HP Business Process Testing User Guide*.

Service Test-specific user interface elements are described below.

Used By Grid

This grid lists the assets that are using the asset currently selected in the tree. Suppose you are looking at the Used By grid for a shared object repository (for a GUI test) or a data table (for an API test). The Used By grid lists all of the GUI tests or application areas that are associated with the selected shared object repository or all the API tests that are associated with the data table. This list indicates the assets that will be affected if you modify or delete the asset selected in the tree.

Column	Description
ID	A unique numeric ID assigned automatically by ALM. If the **ID** is a link, you can click it to jump to that asset in ALM. **Example:** Suppose you are looking at the Used By grid for a specific function library (GUI testing only) or user code file (API testing only) in the Test Resources module. You can click the **ID** link to jump to the test or application area with which it is associated. (The link takes you to the Test Plan or Business Components module.)

Column	Description
Name	The name of the asset that is using the asset selected in the tree, for example, the name of a test, GUI action, application area, or business process test that is using the asset selected in the tree. **GUI-related** owner names include: • **Main Test Flow.** Indicates the test container called by the top-level action in the currently selected test in the Test Plan module. When **Main Test Flow** is displayed, the **Owner Type** is **Test**. • **Action<#>.** Indicates the internal name of the action that is called by an action in the currently selected test in the Test Plan module. **Action<#>** refers to the sequential number of the action when it was created. **Action<#>** is displayed when the **Owner Type** is **QTP Action**. **Note:Action<#>** is displayed even if an action was renamed in the test. • The actual name of the asset if the asset is not an action. An asset linked to an **API test or component** lists the name of the asset.
Type	The type of asset that is using the asset selected in the tree, for example `ApplicationArea or BUSINESS-PROCESS or Test, QTP Action, QUICKTEST_TEST`, or `SERVICE_TEST`.
Description	The description specified in the **Details** tab for the asset listed in the **Name** column (see description above). **Note: (for GUI testing)** • If the **Type** is **QTP Action**, displays the actual name of the action as shown in Service Test (for example, if the action was renamed, the user-defined name is displayed) and displays its description, if any. • If the **Type** is **QUICKTEST_TEST** or **Test**, this cell is empty.
Owner Name (GUI tests only)	The name of the asset that owns the asset listed in the **Name** column. For example, if the asset listed in the **Name** column is an action, then the **Owner Name** is the name of the test in which that action is stored. (Not relevant for components or API tests)
Owner Type (GUI tests only)	The type of asset that owns the asset listed in the **Name** column. GUI-related owner types include: • **QUICKTEST_TEST.** A GUI test in the Test Plan module. • **QTP Action.** An action that is part of a test in the Test Plan module.

Using Grid

This grid lists all of the dependencies that the selected asset is using. For example, suppose you are looking at a test or component. You can see all of the external actions called by the test, all of the shared object repositories containing test objects used by the test or component, function libraries containing functions called by the test or component, and so on.

Column	Description
ID	A unique numeric ID assigned automatically by ALM.
Name	The name of the associated asset that the selected asset uses, for example, the name of the shared object repository, data table resource, or function library. GUI-related names include: • For **business process tests**, lists the names of the flows and/or components that are included in that test. • For **business process flows**, lists the name of the components that are included in that flow. • For **components**, lists the name of the associated application area. • **Action<#>.** Indicates the internal name of the action that is called by an action in a test in the Test Plan module. **Action<#>** refers to the sequential number of the action when it was created. **Action<#>** is displayed when the **Related Type** is **QTP Action**. **Note:Action<#>** is displayed even if an action was renamed in the test. • For **application areas**, lists the name of all associated resources, including function libraries and shared object repositories. • The actual name of the asset if the asset is not a test.
Type	The type of associated asset that the selected asset uses, for example, **ApplicationArea**, **Component**, **FLOW**, **QTP Action**, **Data table**, **Function library**, **Shared object repository**, and **Recovery scenario**.
Description	The description of the associated asset that the selected asset is using, if any. If the **Type** is **QTP Action**, displays the name of the action as shown in Service Test and displays its description, if any.
Owner Name	The name of the asset that owns the asset listed in the **Name** column. For example, if the asset listed in the **Name** column is an action, then the **Owner Name** is the name of the test in which that action is stored. (Not relevant for components)
Owner Type	The type of asset that owns the asset listed in the **Name** column, for example, **QUICKTEST_TEST**. (Not relevant for components)

History Tab (ALM Modules)

Relevant for: GUI tests and components and API testing

This tab enables you to:

- View version information for a selected file.
- View baseline information for a selected file.

- View and compare file versions.
- View the baseline in which a version is stored (if applicable).
- Check out an earlier version of a file if you want to roll back to that version. (When you check the file back into the version control database, that version becomes the current version.)

To access	In ALM, the History tab is available from the following modules: • Business Components module • Test Plan module • Test Resources module **Note:** The History tab is located in the pane on the right side of the window. You may need to scroll to the right to display it.
Important information	For details on using this tab, see the *HP Application Lifecycle Management User Guide*.
Relevant tasks	In Service Test, you can also view version history and baseline history by selecting one of the following: • **ALM > Version History** • **ALM > Baseline History**
See also	• "Version History Dialog Box" on page 366 • "Viewing Baseline History" on page 360

Libraries Module (ALM)

Relevant for: GUI tests and components

This module enables you to:

- **Create, view, and compare baselines.** For details, see "Viewing Baseline History" on page 360 and the sections describing baselines in the *HP Application Lifecycle Management User Guide*.
- **Import assets from other ALM projects.** This enables you to create a complete copy of the assets that are included in a baseline in another project in any accessible domain. For details, see the *HP Application Lifecycle Management User Guide*.

To access	In the ALM sidebar, under **Management** select **Libraries**.
Important information	For details on using this tab, see the *HP Application Lifecycle Management User Guide*.

Troubleshooting and Limitations - Resources and Dependencies

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for resources and dependencies.

- When you save a resource to ALM (either from Service Test or using the **Upload** option from the ALM Test Resources module), and the resource file has a comma in the file name, the resource appears to be saved successfully, but the file is not actually uploaded to the ALM server.
- **For GUI testing:** If you insert a call to an external action that is associated with a data table, and that data table was previously renamed or moved in the Test Resources module of Quality Center 10.00 or ALM, Service Test tries to locate the data table in its original location.

  Workaround: Save the test, close it, and reopen it.

Chapter 21: Version Control in ALM

Relevant for: GUI tests and components and API testing

Note: The references to ALM in this chapter apply to Quality Center 10.00 and ALM. Note that some features and options may not be supported in the Quality Center or ALM edition you are using. For information on Quality Center or ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

Managing Versions of Assets in ALM Overview

Relevant for: GUI tests and components and API testing

When Service Test is connected to an ALM project with version control support, you can update and revise your Service Test assets while maintaining earlier versions of each asset. This helps you keep track of the changes made to each asset and see what was modified from one version to another. Assets can include tests, components, function libraries, application areas, shared object repositories, recovery scenarios, and external data tables, XML files, user code files, or test activities.

You can check in the asset at any time. For example, you may want to check the asset in every day or when you complete a task. While the asset is checked out to you, other users can view the last checked in version of that asset in read-only mode, but they cannot modify the asset or view your changes until you check in the asset.

If the asset is...	You can...
checked in	• Open the asset in read-only mode using the **Open** option. You cannot modify the asset. • Open the asset and check it out by selecting **ALM > Check Out**. • Open the asset and check it out immediately using the **Open and Check out** option in the "Open/New <Document>/<Resource> Dialog Box" (described on page 69). You can modify the asset as needed. **Tip: (for GUI testing):** If a test, component, function library, or application area is checked into a project with version control, the document tab indicates its **(Read-Only)** status. **Note:** If you check out a test, and that test is associated with a data table that is currently checked in to the ALM project, the data in that data table is read-only, even though the test is editable. To modify the data table, you must first check out the data table in the ALM Test Resources module.
checked out to your ALM user name	Open the asset in read-write mode, using the **Open** option and modify the asset as needed. **Tip:** If a test, component, function library, or application area is checked out to your ALM user name, an unlocked icon  in the Solution Explorer indicates this status.

If the asset is...	You can...
checked out to another ALM user	Open the asset in read-only mode using the **Open** option. Service Test displays a message indicating that the asset is checked out to another ALM user. You can view the last checked in version of the asset now, and you can check out the asset later after the other user checks in the asset. **Tip:** If a test, component, function library, or application area is checked out to another ALM user name, the document tab indicates this status by displaying a locked icon and **(Read-Only)** adjacent to the document's name.

In Service Test, you can check out only the latest version of an asset, although you can view and compare earlier versions. This is because assets that are stored in ALM are often linked to or **dependent on** one another.

For example, if you try to run an earlier version of a test or component with a later version of a shared object repository or a data table, your test or component might fail because the objects in the object repository or data table would not necessarily match the objects or steps in the test or component. For more details, see "Troubleshooting and Limitations - ALM Version Control" on page 370.

Viewing Version Control Information When Opening a Test

Relevant for: GUI tests and components and API testing

When you open a test from an ALM project with version control support, you can view version control information for the test by using the **Details** view in the "Open/New <Document>/<Resource> Dialog Box" (described on 69).

The **Checked Out To** column specifies the user name of the ALM user to whom the test is checked out, if it is checked out. If the test is currently checked in to the version control database, there is no indication in the dialog box.

This section also includes:

How ALM Manages Assets

Relevant for: GUI tests and components and API testing

You manage asset versions by checking assets in and out of the version control database.

You add an asset to the version control database by saving it in an ALM project with version control support. When you save an asset for the first time, Service Test automatically checks the asset into the ALM version control database, assigns it version number 1, and automatically checks the asset out for you so that you can continue working on it. When you check the asset in, the asset retains version number 1, since this is the first version that can contain content. Then, each time the asset is checked out and in again, the version number increases by 1.

Note: If you create an asset directly in ALM, the asset is assigned version number 1 and is immediately checked out to you. In ALM, version number 1 represents the created asset without content. When you next check the asset in, ALM assigns it version number 2.

View and Compare Asset Versions

Relevant for: GUI tests and components and API testing

If your project administrator creates project baseline versions when a milestone is reached during product development, you can view and compare the asset versions stored in these baselines. For details, see "Viewing Baseline History" on next page.

Note: With the exception of the **Baseline History** option, the **ALM Version Control** options in the **ALM** menu are available only when you are connected to an ALM project with version control support, and an asset stored in ALM is open in the Service Test window.

Adding Assets to the Version Control Database in an ALM Project

Relevant for: GUI tests and components and API testing

When you create a new asset or use **Save As** to save an existing asset in an ALM project with version control support, Service Test automatically saves the asset in the project, checks the asset into the version control database with version number `1`, and then checks it out so that you can continue working. This is an administrative version of the asset, similar to a placeholder. The version number indicates that the asset exists in the database. When you later check in the asset, the version number remains version number `1`—the first version that you are checking in. Subsequent checkins increase the version number by 1.

Saving your changes to an existing asset does not check them in. Even if you save and close the asset, the asset remains checked out until you choose to check it in. For details, see "Checking Assets into the Version Control Database" on next page.

Checking Assets Out of the Version Control Database

Relevant for: GUI tests and components and API testing

When you open an asset that is currently checked in to the version control database, it is opened in read-only mode. You can review the checked-in asset. You can also run the asset and view the results.

To modify the asset, you must check it out. When you check out an asset, ALM copies the asset to your unique check-out folder (automatically created the first time you check out an asset), and locks the asset in the project database. This prevents other users of the ALM project from overwriting any changes you make to the asset. However, other users can still run the version that was last checked in to the database.

You can save and close the asset, but it remains locked until you return the asset to the ALM database. To release the asset, either check the asset in, or undo the check out operation. For more details on checking assets in, see "Checking Assets into the Version Control Database" below. For details on undoing the check-out, see "How to Cancel a Check-Out Operation" on page 363.

In Service Test, the check out option accesses the latest version of the asset. In ALM, you can also check out earlier versions of any asset except for application areas. For details, see "Version History Dialog Box" on page 366 and the ALM user guide.

Checking Assets into the Version Control Database

Relevant for: GUI tests and components and API testing

While an asset is checked out, ALM users can run the previously checked-in version of your asset. For example, suppose you check out version 3 of an asset and make a number of changes to it and save the asset. Until you check the asset back into the version control database as version 4, ALM users can continue to run version 3.

When you have finished making changes to an asset and you are ready for ALM users to use your new version, you check it in to the version control database.

Note: If you do not want to check your changes into the ALM database, you can undo the check-out operation. For details, see "How to Cancel a Check-Out Operation" on page 917.

When you check an asset back into the version control database, ALM deletes the asset copy from your checkout folder and unlocks the asset in the database so that the asset version is available to other users of the ALM project.

Viewing Baseline History

Relevant for: GUI tests and components and API testing

In ALM, a project administrator can create baselines that provide "snapshots" of an entire project (or part of a project) at different stages of development. A **baseline** represents a version of a project at a specific point in a project's life cycle. For example, baselines are often created for each milestone or when specific phases in a project are completed.

Baselines can be created for ALM projects that are enabled for version control, and for projects for which version control is not enabled.

The project administrator creates the baseline in the Libraries tab of the Management module in ALM. Creating a baseline is a two-fold process. The administrator first creates a library, which

specifies the root folders from which to import the data. The administrator makes sure to include all of the associated resource files, such as shared object repositories and function libraries. The administrator then creates the actual baseline, which comprises the latest versions of every asset included in the library. If the project is version control-enabled, then these are the latest checked in versions of every asset.

During the creation process, ALM verifies that all of these assets (such as associated resource files) are included in the baseline. If any assets are not included, ALM informs the administrator so that the library and baseline can be modified accordingly. For more details, see the ALM user guide.

In ALM, these baselines can be viewed and compared in their entirety.

In Service Test, you can view and compare the assets saved in these baselines. This enables you to review the content of an asset at a specific phase in the project time line.

You can also run a test or component from a baseline.

Version History Versus Baseline History

Relevant for: GUI tests and components and API testing

This section focuses on the differences between version history and baseline history and describes when to use each.

- You use version control to check in and check out assets as needed. For example, you may want to check in an asset on a daily basis or only when significant results are achieved. This enables you to monitor the asset's development.

  If you want to view the content of an asset on a particular date or after a particular user checked in the asset, use the **Version History** option to view or compare the asset.

- The ALM project administrator creates baselines that represent "snapshots" of a project's assets at various milestones in a project's life cycle. Each baseline links to the assets specified by the administrator when the baseline was created. The asset version represented in the baseline is always the version that was checked in when the baseline was created.

  If you want to view an asset as it was saved for a particular milestone, use the **Baseline History** option.

Tasks

How to Manage Version Control Operations

Relevant for: GUI tests and components and API testing

This task describes how to check in the currently open asset, check out the latest version of an asset in order to edit it, and cancel a check-out operation if needed.

After you edit an asset, you can save changes and close it without checking in the modified asset. However, your changes are not available to other ALM users until you check it in. If you do not want to check your changes in, you can undo the check-out by canceling it.

For more details on checking assets in, see "Checking Assets into the Version Control Database" on page 360.

This task includes the following steps:

- "How to Check In the Currently Open Asset" below
- "How to Check Out the Latest Version of an Asset" below
- "How to Cancel a Check-Out Operation" on next page

How to Check In the Currently Open Asset

1. Confirm that the currently open asset is checked out to you. For details, see "Version History Dialog Box" on page 366.

   Note: If the open asset is currently checked in, the **Check In** option is disabled. If you open an asset that is checked out to another user, all **ALM Version Control** options, except the **Version History** option, are disabled.

2. Select **ALM > Check In, and check in the asset using the Check In dialog box.** For details, see "Check In Dialog Box" on page 365.

How to Check Out the Latest Version of an Asset

1. Make sure the asset you want to check out is currently checked in. If you open an asset that is checked out to you, the **Check Out** option is disabled. If you open an asset that is checked out to another user, all ALM version control options, except the **Version History** option, are disabled.

   Note: Note about version numbers: Prior to Quality Center 10.00, version numbers consisted of three segments separated by periods, for example `1.7.4`. In Quality Center 10.00 and ALM, version numbers consist of a single segment, for example `12`.

2. Open the "Open/New <Document>/<Resource> Dialog Box" as follows:

If the asset is a:	Do this:
Test, Component or Function Library	In the main Service Test window, select **File > Open > Test, BusinessComponent**, **Function Library**, or **Application Area** or click the **Open** down arrow and select the asset type from the list.
Shared Object Repository (**GUI testing only)**	In the Object Repository Manager, select **File > Open** or click the **Open** button.
Recovery Scenario (**GUI testing only)**	In the Recovery Scenario Manager, click the **Open** button.

3. Browse to and open the asset.

   The test opens in the document pane as read-only with a lock icon in the test's tab.

4. Select ALM **> Check Out** to check out the test and edit it.

How to Cancel a Check-Out Operation

1. Open the asset if it is not already open.
2. Select **ALM > Undo Check out.**
3. Click Yes to confirm the cancellation of your check out operation.

   The check out operation is cancelled. The checked out asset closes, and the previously checked in version reopens in read-only mode.

Reference

Version Management Commands

Relevant for: GUI tests and components and API testing

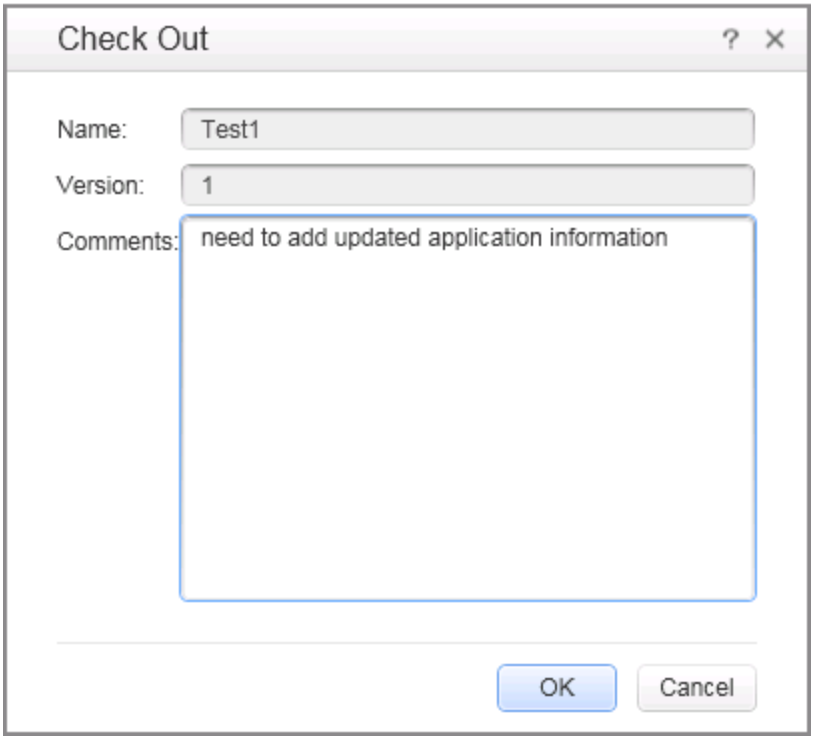
The following version control commands are available in Service Test and can be used when connected to an ALM project that has version control enabled:

- **Check Out.** Enables you to check a version-controlled asset out of the version control database. For details, see "Checking Assets Out of the Version Control Database" on page 359.
- **Undo Check Out.** Enables you to cancel the check out of a version-controlled asset from the version control database. For details, see "How to Cancel a Check-Out Operation" on page 917.
- **Check In.** Enables you to check an asset in to the version control database. For details, see "Checking Assets Out of the Version Control Database" on page 359.
- **Version History.** Enables you to view or compare the versions of a particular asset. For details, see "Managing Versions of Assets in ALM Overview" on page 357.
- **Baseline History.** Enables you to view or compare the versions of a particular asset as it was saved in a project's baselines. For details, see "Viewing Baseline History" on page 360.

Check Out Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to check an ALM asset out of the ALM version control database so that you can edit it.



To access	**ALM > Check Out**
Relevant tasks	"How to Check Out the Latest Version of an Asset" on page 362

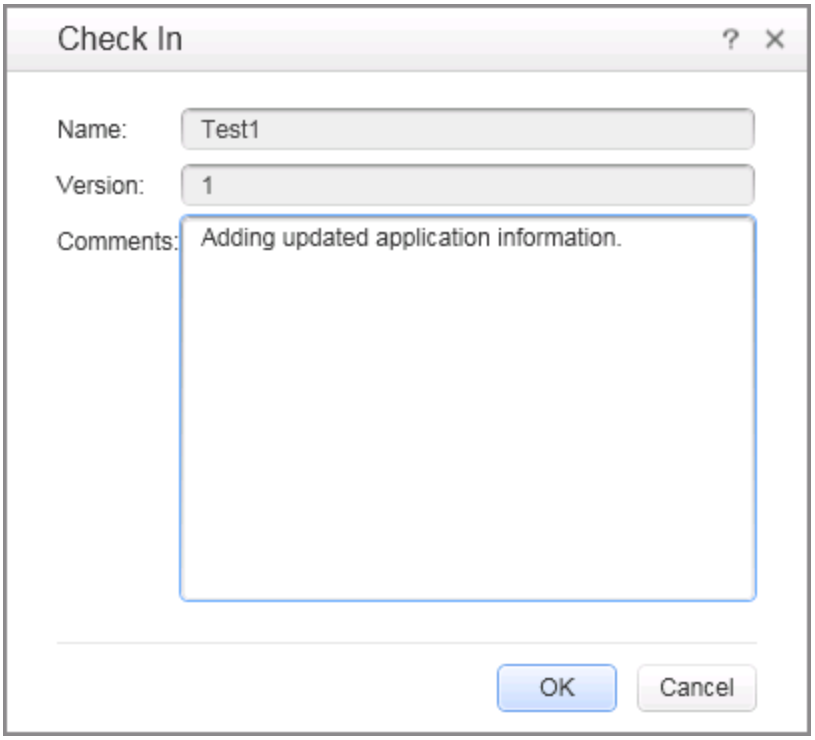
User interface elements are described below:

UI Elements	Description
Name	The name of the asset.
Version	The version number of the asset.
Comments	Enter any comments for the version in the Comments area.

Check In Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to check an ALM asset in to the ALM version control database.



To access	**ALM > Check In**
Relevant tasks	"How to Check In the Currently Open Asset" on page 362

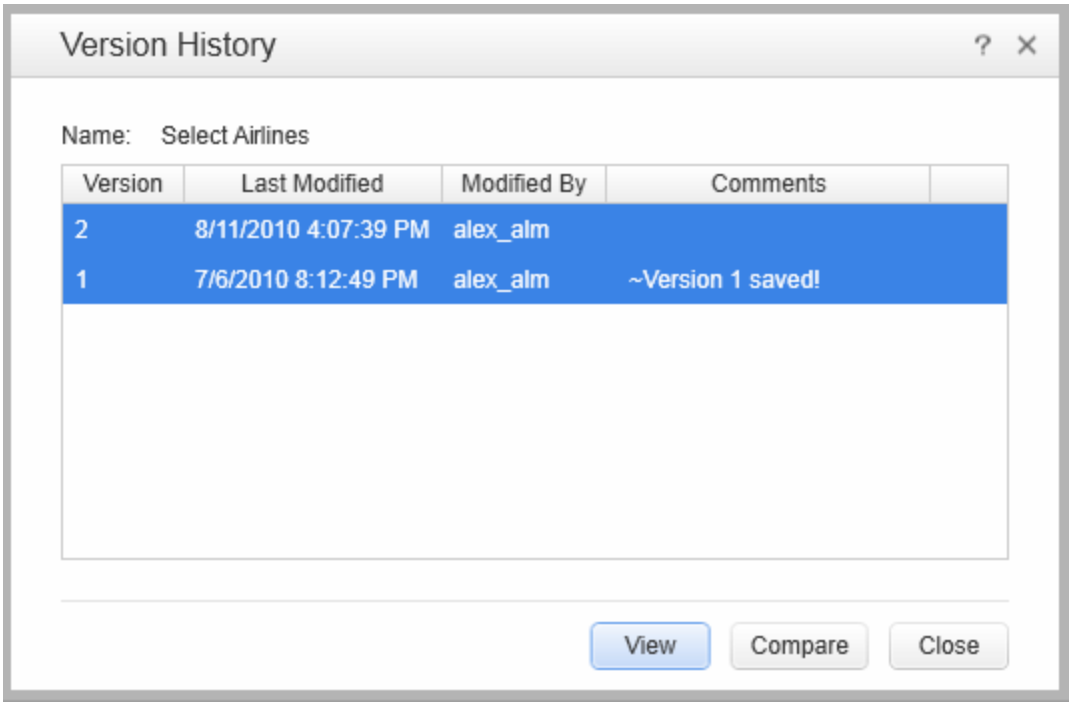
User interface elements are described below:

UI Elements	Description
Name	The asset name.
Version	The new version number. By default, the new version number is one number higher than the previously checked in version.
Comments	If you entered a description of your change when you checked out the asset, the description is displayed in the **Comments** box. You can enter or modify the comments in the box.

Version History Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to view the version history for an asset, view the content of a previous asset version, and compare two asset versions.



To access	• **From most assets:** Open the asset and select the **ALM > Version History** menu command.
Important information	**To view a version for an asset:** Select a version and click **View**. **To compare two versions of an asset:** Select two versions and click **Compare**. You cannot check out an earlier version of an asset from this dialog box. (You can check out earlier version of most assets directly from the ALM project. For details on checking out assets from ALM, see the ALM user guide.)
See also	"Managing Versions of Assets in ALM Overview" on page 357

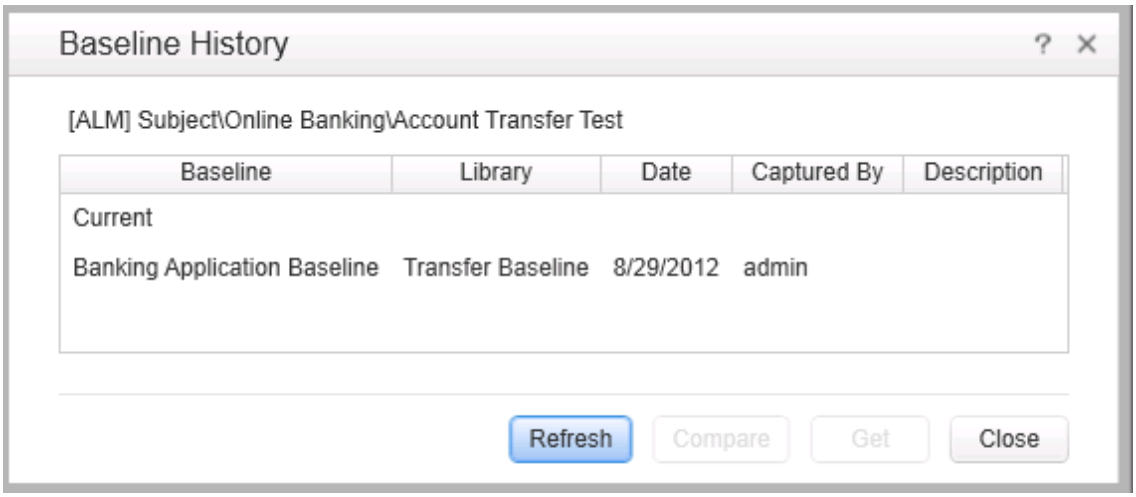
User interface elements are described below:

UI Elements	Description
Name	The name of the currently open asset.
Version column	A list of all versions of the asset.
Last Modified column	The date that each version was checked in.
Modified By column	The user who checked in each listed version.
Comments column	The comments that were entered when the selected asset version was checked in.

Baseline History Dialog Box

Relevant for: GUI tests and components and API testing

This dialog box enables you to view and compare read-only baseline "snapshots" of an asset.



To access	• **Most assets:** Open the asset and select the **ALM > Baseline History** menu command. • **Recovery scenario (GUI testing only):** In the Recovery Scenario Manager, open the recovery scenario, click the **Version Control** down arrow, and select **Baseline History.**
Important information	In the ALM Test Lab module, you can use the **Pin to Baseline** option to run a baseline version of an asset. For more details, see the ALM user guide.
See also	"Viewing Baseline History" on page 360

User interface elements are described below:

UI Elements	Description
Refresh button	Reloads the baselines in the Baseline History dialog box with the latest changes. For example, if a baseline is added while this dialog box is open, clicking **Refresh** updates the list of baselines.
Baseline column	Lists all of the baselines that include this asset. Baselines are defined in the ALM project (**Management** module > **Libraries** tab).
Library column	Lists the libraries from which each baseline was created.
Date column	Lists the date that each baseline was created.

UI Elements	Description
Captured By column	Lists the ALM user who created each listed baseline.
Description column	Displays any comments that were added when the baseline was created.
Compare button	Enables you to view a comparison of the currently open asset in two baselines.
Get button	Enables you to open the current asset from the selected baseline. **To view the asset as it was stored in a baseline:** Select a baseline from the list and click **View**. When you click **Get**, Service Test: • Closes the currently open asset. • Opens the same asset from the baseline you selected. • Loads the baseline version of the external actions and resource files that are associated with the asset, if any, when they are called. **Note:** If an external GUI action or resource file is associated via a relative path, loads the latest version of the action or resource file instead of the version from the baseline.

Troubleshooting and Limitations - ALM Version Control

Relevant for: GUI tests and components and API testing

This section describes troubleshooting and limitations for ALM version control.

- If you need to check out an earlier version of an asset, for example, to roll back to an earlier version, contact your ALM project administrator. Your administrator needs to ensure that the correct versions of all relevant assets become the latest versions.
- When working with a version-control-enabled ALM project, it takes a long time to save a test for the first time (up to twice as long as saving the same test in a project without version control support enabled). This delay does not occur on subsequent saves of the test.
- To enabletest versioning in Quality Center version 10.00, 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>
```

Part 6: Testing Design

Chapter 22: Testing Services

This chapter includes:

Concepts

API Testing Overview

Welcome to Service Test, HP's tool for the construction and execution of functional tests of non-GUI applications.

You create a test by dragging and dropping activities from the Toolbox pane onto a canvas. The Toolbox pane 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 pane by importing WSDLs or using services from the repository. You can also create new custom activities, using the built-in Activity Wizard. For details, see "Extensibility in Service Test" on page 661.

This section includes:

Performing Integrated Testing

To test your applications and services, you perform known activities and observe the response. By checking the response, you determine whether your system is performing as expected.

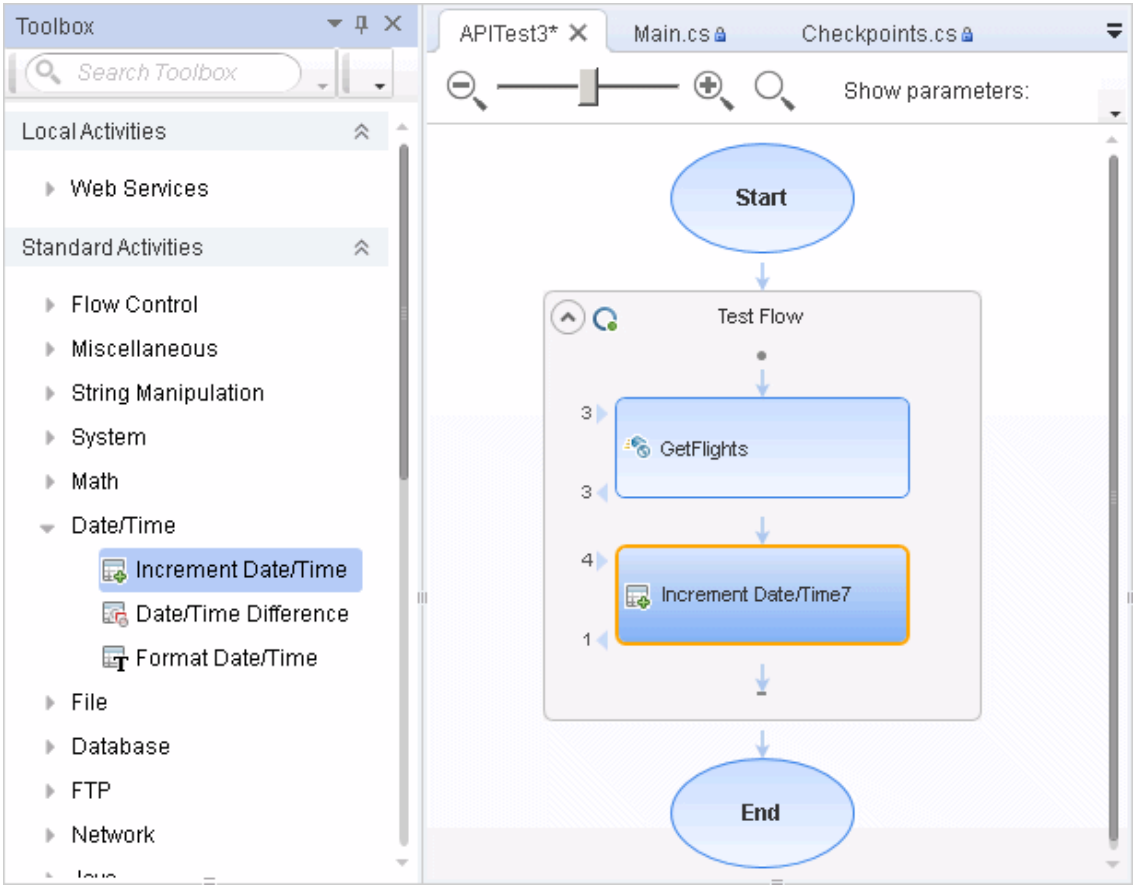
For example, to implement the **Database > Open Connection** activity, you drag the activity onto the canvas and provide the database connection string. You then can drag database activities onto the canvas such as the **Select Data** activity. After running the activities, you check the response to confirm that it generated the expected value.

The steps to create a basic test are:

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

The Canvas

The canvas provides a visual representation of the test flow. You populate the canvas by dragging in activities from the Toolbox pane. For details, see "Toolbox Pane" on page 254.



The Toolbox pane contains many built-in services, such as string and file manipulation activities. For a non built-in service to appear in the Toolbox pane, you must first import it or define it, such as Web and REST services, .NET assemblies and SAP IDocs or RFCs. For details, see "Local Activities" on page 463.

Tab Title

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

Manipulating Steps

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

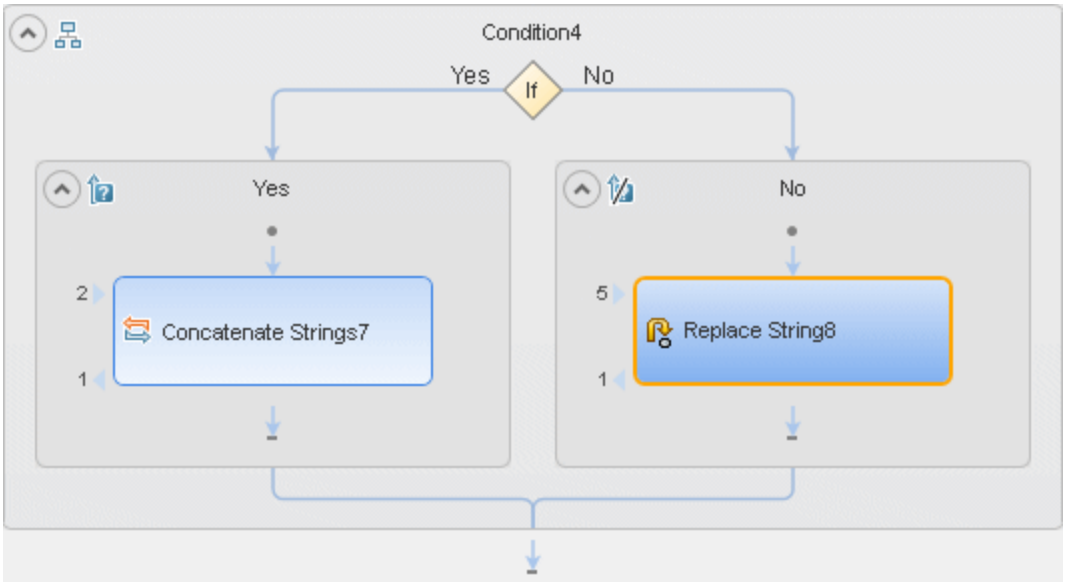
- **Reorder**. Drag a step from one location to another.
- **Copy and Paste**. Select a step and press CTRL+C to copy it to the clipboard. Press CTRL+V to paste it into another location within the Test Flow or to another loop.
- **Delete**. Select a step and press the keyboard's **Delete** button.

Flow Control

The canvas enables 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/Output Properties Tab (Properties Pane)" on page 201.

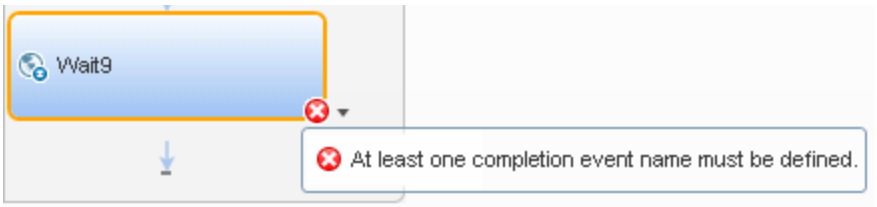
- **Conditional steps.** Adds a two-branched conditional node to the test.



- **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

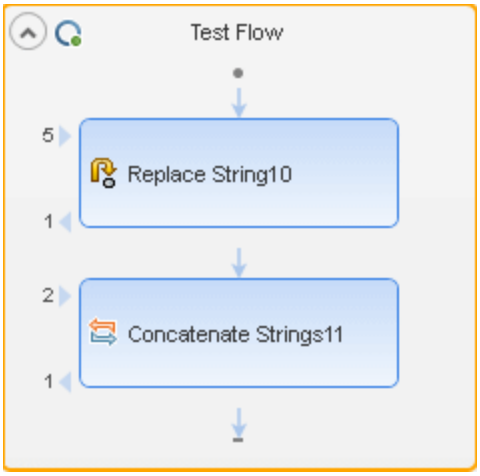
Steps that require intervention in order to run, such as **HTTP**, **Custom Code**, **Wait**, and **SOAP Request**, post an alert in their top right corner. Clicking an alert button displays the action that you are required to do.



You can also view this information in the Errors pane by selecting **View > Errors**. For details about the Errors pane, see "Errors Pane" on page 172.

Connecting Data

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

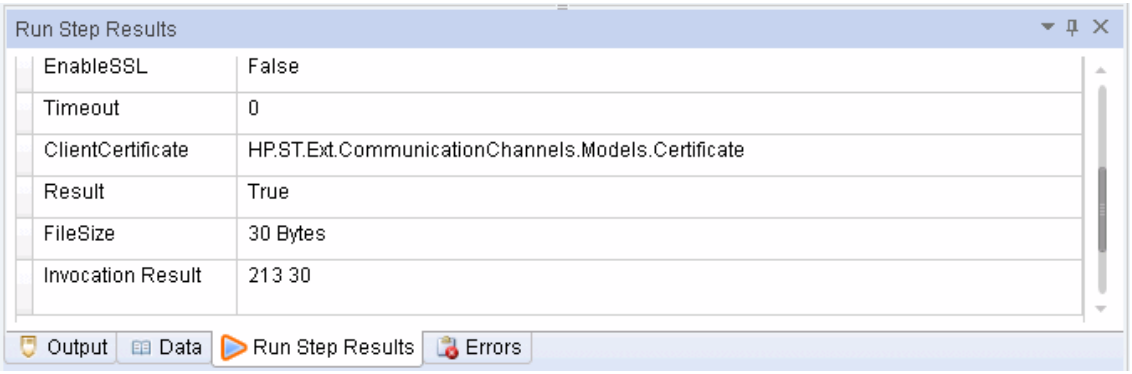


For details, see the "Select Link Source Dialog Box" on page 630.

For details about linking multiple steps to a single result, see "Outgoing Links" on page 612.

Testing Steps

The **Run Step** utility lets you run a single test step without running the entire test. Using the property values from the Properties pane, the **Run Step** operation invokes the step locally and presents the results in a separate pane, the **Run Step Results** pane. The following example shows the results of an **FTP File Get Size** step.



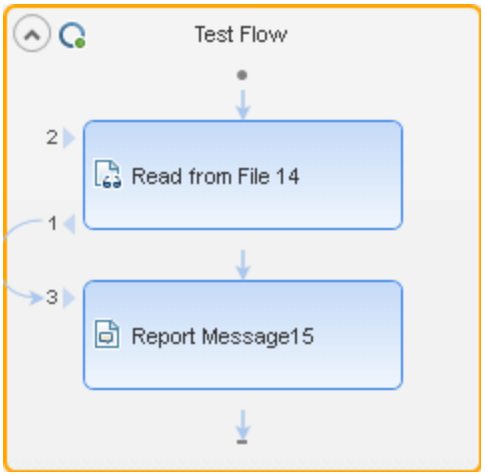
For more information, see the "Run Step Results Pane" on page 219.

For details about the **Run Step** utility, see the "Canvas User Interface" on page 391.

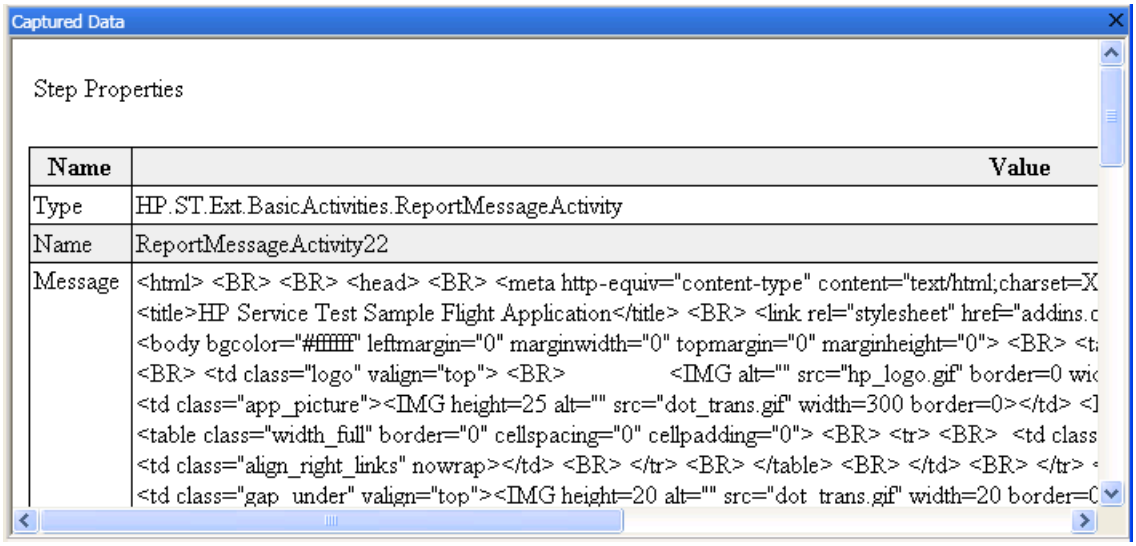
Sending Messages to the Output and Results Viewer

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 Properties pane.

As with any step, you can link its input to the output of a prior step and view the results in the Output pane or Run Results Viewer.



In the following example, the Run Results Viewer shows the **Message** string.



For details, see the Captured Data Pane Contents for / Steps (described in the Run Results Viewer) and "Output Pane" on page 180.

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 details, see "Coding Service Test Events" on page 641.

In addition, through the activity wizard (**Start > All Programs > HP Software > HPService Test> Tools > Activity Wizard**), you can create custom testing activites. The Activity Wizard enables you specify the activity type and properties. It then exports the activity to the Toolbox pane for use in future testing sessions.

For details on adding new activities and custom code, see "Extensibility in Service Test" on page 661.

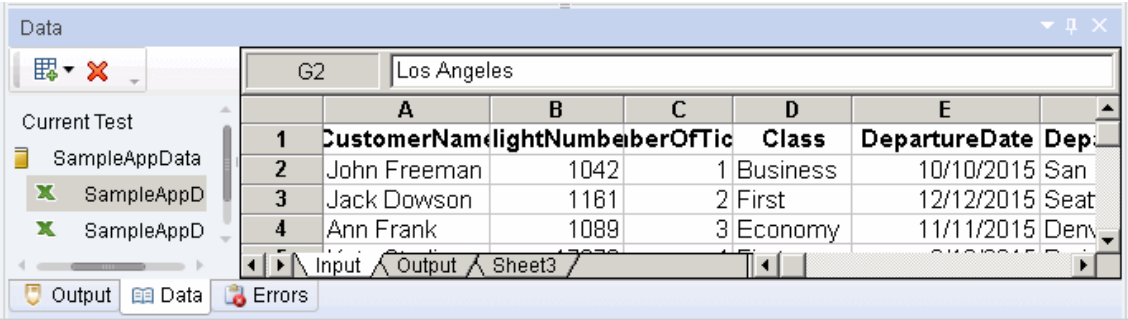
Note: For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

Data Sources

Using the Data Pane, 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 locally define data 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. If you create a referenced data source, you will be unable to edit the values in the data grid

A data grid displays the data.



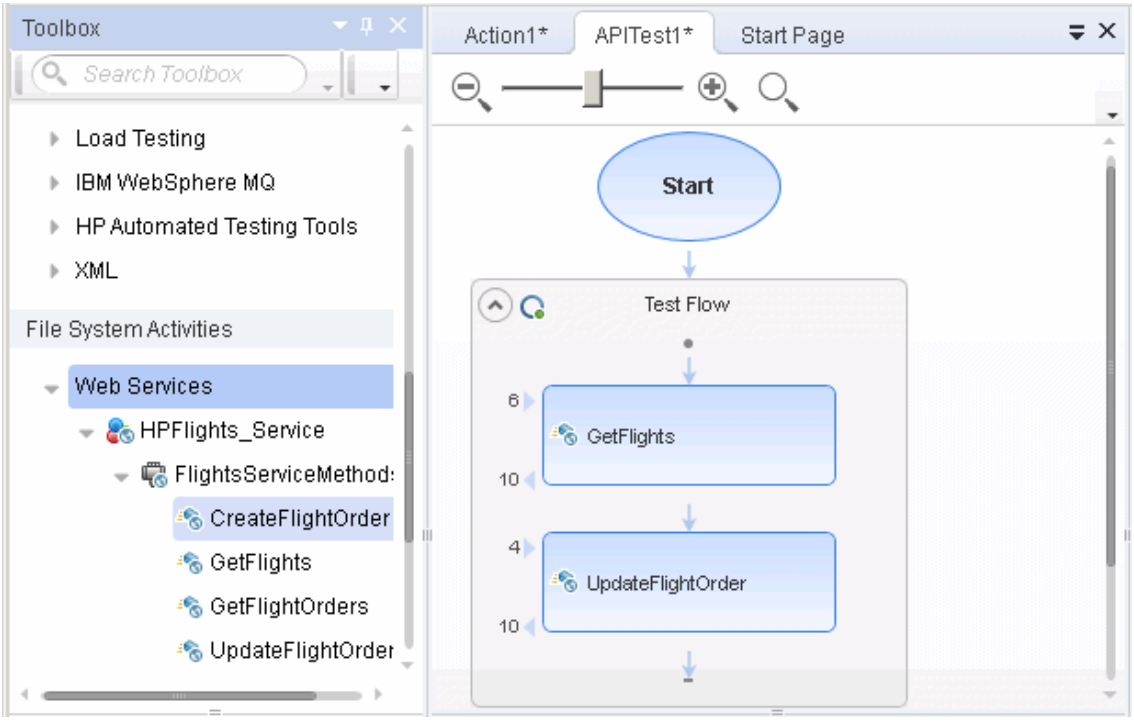
After you load data, you can assign it to properties in your test.

For information on using data in your test, see "How to Assign Data to Test Steps" on page 621.

Testing Web Services

In order to test Web services, you import the WSDL file which describes the service.

After importing a WSDL file, the Toolbox pane displays the service's operations. You then drag the operations directly onto the canvas and provide values for the input properties.



For task details, see "How to Import a WSDL-Based Web Service" on page 472.

Testing REST Services

You can also 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.

You can also create prototype for individual REST methods. The prototype includes all of the request and header information, as well as custom properties. Using this prototype, you can drag the method onto the canvas without having to repeatedly configure its details.

For task details, see "How to Create a REST Method" on page 475.

Java Testing

To call Java classes use the **Call Java Class** activity.

In order to use this activity, 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 property 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:** `<product_installation>\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java\ServiceTestCall.java`
Sample code	• **Path:** `<product_installation>\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 "Java Class Dialog Box" on page 459.

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 402.

Automated Testing Tool Integration

HP Automated Testing Tools activities allow you to create tests with steps that consist of a GUI, API, Service Test tests, or LoadRunner scripts.

You create the tests or scripts in the original application and call them from within your Test flow.

For user interface details, see the "Toolbox Pane" on page 254.

For task details, see "How to Call Tests from Other Applications" on page 411.

Business Process Testing

Business Process Testing is a methodology in which several test steps or components, are combined to create a complete business process. You compose a business process by combining a series of test components with data flow between them.

Business 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 components from within ALM or through Service Test.

In ALM, using the Business Component 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 ALM, see the *HPBusiness Process Testing User Guide*.

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

Benefits of Business Process Testing

Some of the advantages of working with a Business Component model 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
- Enables 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 business components in ALM, 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.

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

- Load Test activities are not supported.
- Automated Testing Tool activities are not supported.
- Business components cannot be saved on the file system—only on ALM.
- 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.
- The business component's name must not contain one of the following characters: \, /, :, ", ?', <, >,|*, %, !, {, or }.

JMS Transport Overview

The JMS transport method 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 Properties pane's Test Settings. For details, see the "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.

For a list of the activities and their properties, see "JMS Activities" on page 431.

For task details, see "Set up the Java environment - optional" on page 385.

Tasks

How to Create an API 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 "How to Define Test Properties or User/System Variables" on page 59.

This task includes the following steps:

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 472.

For REST services, build the metadata for the service. For details, see "How to Create a REST Method" on page 475.

Skip this step when using the built-in operations, such as **String Manipulation** activities.

2. Create the test flow - optional

Expand the Toolbox pane nodes and drag **Flow Control** activities onto the canvas:

- **Loop.** Enables you to add a loop other than the standard **Test Flow**. You specify the loop behavior in the loop's input properties.
- **Condition**. Enables you to define conditional branches.
- **Sleep.** Indicates a time delay in milliseconds.

3. Add activities to the test

Expand the nodes of the Toolbox pane and drag activities into the **Text Flow** or **Loop** box within the canvas. If you added a **Condition** step, drag activities into the condition branches. For a list of the activities, see the "Standard Activity Types" on page 417.

4. **Define data for the test**

   For details, see "How to Assign Data to Test Steps" on page 621 or "How to Data Drive a Test Step" on page 623.

5. **Create a Custom Code activity - optional**

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

   a. Click the Input/Checkpoints tab in the Properties pane.

   b. Click **Add Properties** and create the required input and output properties.

   c. Open the **Events** view in the Properties pane.

   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 the custom code and the test.

   For details and examples, see "Coding Service Test Events" on page 641.

6. **Test the step**

   Select the step in the canvas and choose **Run Step** from the context menu to run the step with its property values. Note the results in the **Run Step Results** pane, in the lower section of the main window. If something needs to be modified, do so at this point. For details, see "Run Step Results Pane" on page 219.

   > **Note:** The Run Step command, for steps whose properties are assigned an XML data source, does not retrieve the latest data shown in the user interface.
   >
   > **Workaround**: Save the test before activating the Run Step command.

7. **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** tab in the Properties pane.

   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 a file as the **Origin** of the attachment using the **Browse** button .

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

8. **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 Properties pane.

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. It calculates the specified file's checksum using the MD5 Hash function.

For user interface details, see "Attachments Tab (Properties Pane)" on page 187. You can also check for received attachments in the test's folder, stored as files with a `.bin` extension.

9. ## Set up the Java environment - optional

For JMS messages and Call Java Class activities, you set up the Java environment before adding a test step.

a. Click in a Start or End step and click the **Test Settings** tab in the Properties pane. Set the Java test settings for the VM and JMS. For details, see the "Test Settings Tab (Properties Pane)" on page 209.

b. Expand the **JMS** node in the Toolbox pane and drag a **JMS** activity onto the canvas.

c. Set the step's properties. Click on the step in the canvas, and select the Input/Checkpoints tab in the Properties pane. Enter a value for Queue, Subscription, Topic name, and any other relevant property.

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 382.

10. ## Configure SOAP Fault information - optional

To apply negative testing to a Web service:

a. Open the **SOAP Fault** tab .

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 Tab (Properties Pane)" on page 209.

11. ## 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 Properties pane.

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 the `TestUserCode.cs` file and the test.

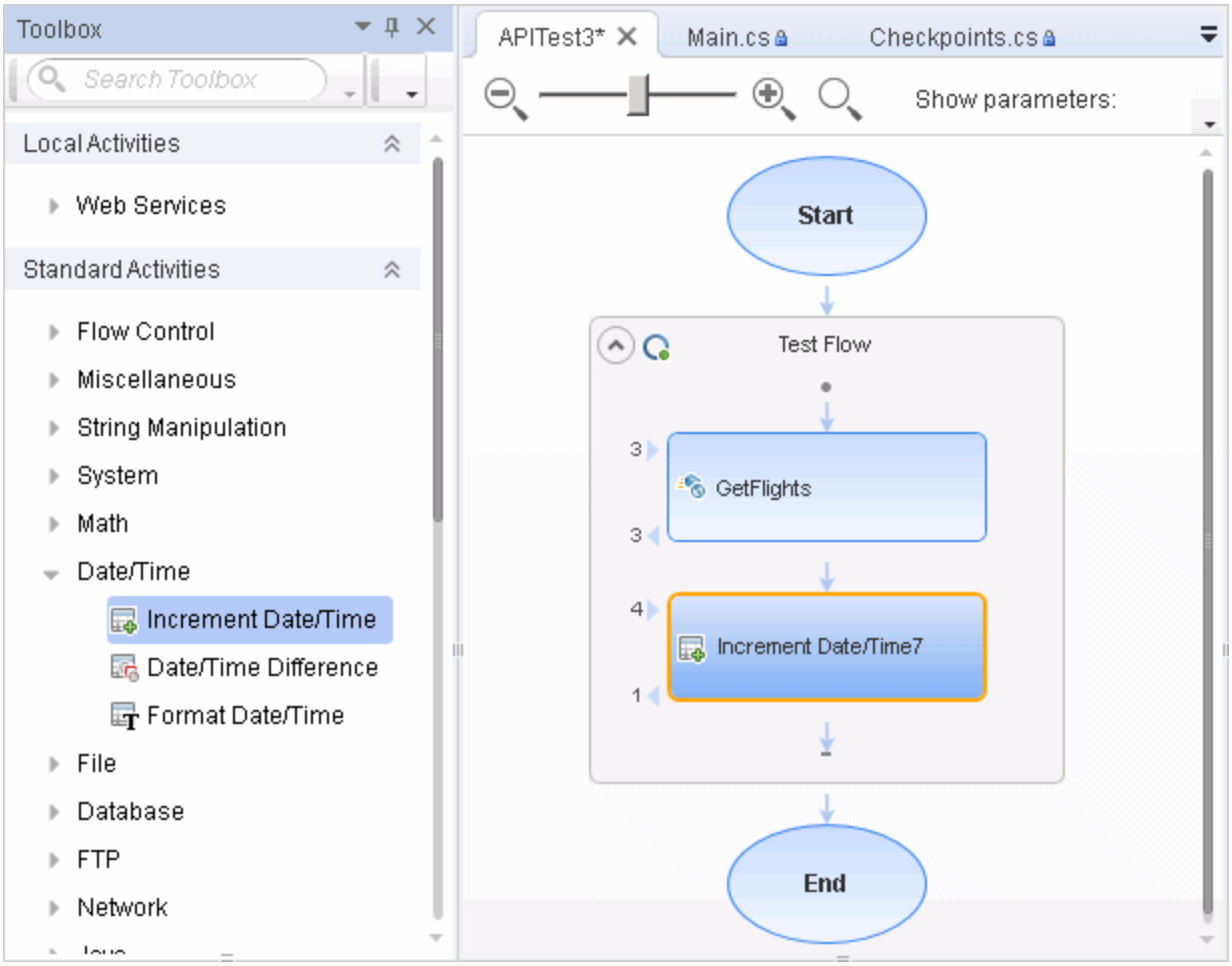
For details and examples, see "Coding Service Test Events" on page 641.

Tip: To ignore all event handlers for a step, select a handler and press DELETE to clear the field in the grid. To remove the handler, you must delete the code manually in the `TestUserCode.cs` tab.

References

Main Window - Service Test

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



To access	1. Select **Start >[All] Programs > HP Service Test > HP Service Test.** **Note:** For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33. 2. Create or open a test.
Relevant tasks	"How to Create an API Test" on page 383

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

UI Elements	Description
	**New.** Opens the "New <Document> Dialog Box" (described on page 67).
	**Open.** Opens an existing test.
	**Add.** Adds a new or existing test to a solution.
	**Save.** Saves the current test. **Tip:** Select **File > Save As** to save the test to a new location.
	**Save All.** Saves all open files.
	**ALM Connection.** Opens the "ALM Connection Dialog Box" (described on page 335) for defining server information and logging into an ALM project.
	**SEV Settings.** Opens the "Virtualized Services Settings Dialog Box" (described on page 598), enabling you to manage virtualized services included with your test.
	**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.** Saves, compiles and runs the current project. • **Run Test in Load Testing Mode.** Saves, compiles and runs the current test in Load Test mode with the **mdrv** driver. **Note:** The **Run Test in Load Testing Mode** 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).
	**Insert Call to New Action.** Opens the "Insert Call to New Action Dialog Box" (described on page 516).

UI Elements	Description
	Opens one of the Import dialog boxes: • **Import WSDL from URL or UDDI.** Opens the Import Service from URL or UDDI dialog box. For user interface details, see "Import/Update WSDL from URL or UDDI Dialog Box" on page 489. • **Import WSDL from File or ALM Application Component.** Opens the Import WSDL dialog box for importing a WSDL from the file system or HP ALM with HP Service Test Management.
	**Import .NET assembly.** Opens the "Import .NET Assembly Dialog Box" described on page 494.
	**Add REST Service.** Opens the "Add/Edit REST Service Dialog Box" described on page 495.
	**Solution Explorer Pane.** Opens the Solution Explorer or brings it into focus.
	**Toolbox Pane.** Opens the Toolbox pane or brings it into focus.
	**Properties Pane.** Opens the Properties pane or brings it into focus.
	**Data Pane.** Opens the Data pane or brings it into focus.
	**Debug Pane.** Brings the Breakpoints pane into focus. Click the down arrow to select a different debug pane.
	**Last Run Results.** Opens the Run Results Viewer.
	**Help.** Opens the Service Test Help.
Document 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`) For details on the document pane, see "Document Pane Overview" on page 132.

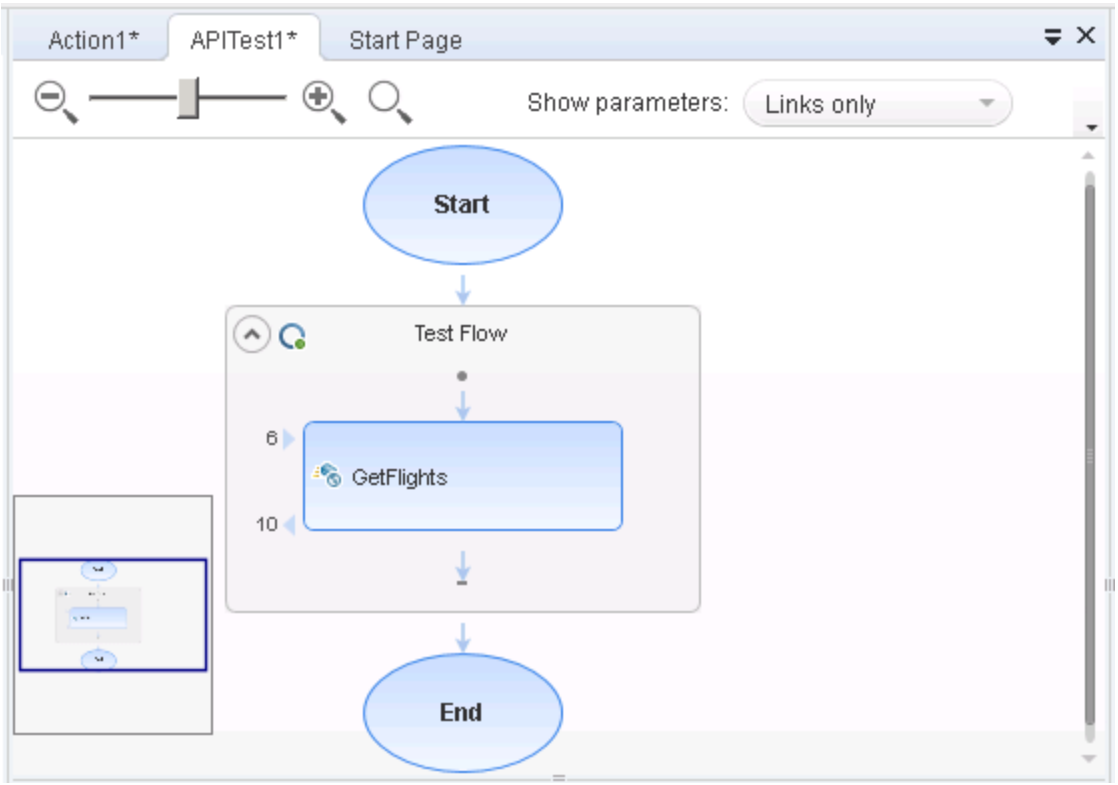
UI Elements	Description
Data Pane	Displays the data from an Excel, XML, or JSON file, a database, or local tables. You can use this data in your tests. • **New.** Creates a new data node in the Data pane through: ▪ **Excel File.** Opens the "New/Change Excel Data Source Dialog Box" (described on page 98) for creating a new set of data. ▪ **Local Table.** Opens the "New Local Table Data Source Dialog Box" (described on page 106). ▪ **XML.** Opens the "New XML Data Source Dialog Box" (described on page 100). ▪ **Database.** Opens the "Add New Database Data Source Wizard" (described on page 102). • **Delete.** Removes the selected data source. For details, see "Data Pane" on page 96.
Debug Panes	Displays panes to assist in debugging your user code files, including the following panes: • **Breakpoints** • **Call Stack** • **Loaded Modules** • **Threads** • **Variables** • **Console** • **Watch** For details, see " Debug Panes Overview" on page 111.
Errors pane	Displays errors, warning, and messages about the test run. For details, see "Errors Pane" on page 172. The default location is the bottom pane.
Output pane	Displays the Output log for the compilation and test run. For details, see "Output Pane" on page 180. The default location is the bottom pane.
Properties pane	A series of screens showing the general properties, schemas, snapshots, and event information for the activity selected in the canvas. For details, see "Properties Pane" on page 184. **Note:** Only active when viewing the canvas.

UI Elements	Description
Solution Explorer pane	Displays the test components in a tree hierarchy. For details, see the "Solution Explorer Pane Overview" on page 235. The default location is the left pane.
Toolbox pane	A tree view of all of the available activities. This includes built-in activities and operations of imported services. For details, see the "Toolbox Pane" on page 254. The default location is the left pane.

Canvas User Interface

The canvas is displayed by default when you create a new test. It provides a visual representation of the test flow and enables you to:

- Display test and action steps
- Manage steps and change their order
- Test individual steps using the Run Step command
- Open wizards and dialog boxes to resolve steps with missing information.



To access	1. Create or open a test or component. 2. Click on the test canvas tab.
See also	"Activity Overview" on page 397

This pane includes the following icons and context menu options:

- "Main User Interface Elements" below
- "Context Menu Options" on next page

Main User Interface Elements

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

UI Element	Description
<Zoom slider>	Enables you to zoom in or out of the test flow image. Do one of the following: • Click the magnifying glass with the plus sign to zoom in on (enlarge) the test flow image. • Click the magnifying glass with the minus sign to zoom out of (shrink) the test flow image. • Click and drag the zoom slider position handle left or right to zoom in or out of the test flow image.
	<Return to default size button>. Returns the test flow image to the default size and the zoom slider position to the center.
Show parameters	Enables you to select the level of information shown in the canvas, including: • **None:** Shows only the number of input and output properties, adjacent to each step or action frame. Clicking the number shows the property names. • **All:** Displays the names of all the input and output properties and their links, with the total number of properties. • **Links only.** Displays only the links to other steps in the flow, and the number of properties. • **Links and names.** Displays the links to other steps, the name of the linked properties. and the number of properties. • **Custom.** Indicates that the **Show parameters** settings for some of the steps were modified manually and not via the drop down. This entry is not selectable.
	<Minimap display button>. Located above the test flow image on the right, and displays or hides the test flow minimap in the lower left corner of the canvas. The minimap display button is highlighted in grey when the minimap is displayed, and is not highlighted at all when the minimap is hidden.
<Test flow>	Displays the test flow in a flow chart format, as well as any parameter details available, as specified in the **Show parameters** drop-down list.

UI Element	Description
<Minimap>	A small, high-level view of the entire test flow, useful when using large flows and/or small screens. **Note:** When you zoom out of the canvas, the area displayed in the canvas becomes larger. This has the effect of shrinking the test flow image in the minimap because the minimap displays the entire test flow.

Context Menu Options

Context menu items are accessible by right-clicking a step in a test or action.

Option	Description
Cut	Cuts the selected step and places it on the clipboard.
Copy	Copies the selected step to the clipboard.
Delete	Deletes the selected step.
Properties	Opens the Properties Pane. For details, see "Properties Pane" on page 184.
Open	Opens the selected action step in its own tab (for action steps inside the Test Flow).
Paste	Pastes the step currently on the clipboard to the location of the cursor. **Note:** • Available only after you add a step to the clipboard using either **Cut** or **Copy**. • Only visible when clicking on the down arrow inside the Test Flow, Loop, or Action frame.
Run Step	Runs the current step using its properties from the Properties pane, and shows the results in the **Run Step Results** pane. For details, see "Run Step Results Pane" on page 219. **Note:** • This command is not available for all steps. • If an input property was linked to an output property of a previous step, the Run Step dialog box opens. This lets you set constant values for the step's properties. For details, see "Run Step Dialog Box" on page 224.

Troubleshooting and Limitations - Testing Services

- XPath aggregate functions are not supported.
- Service Test cannot open with a license that was installed with Service Test 9.53.
  Workaround: Reinstall the license using the Service Test License Manger, available from the Start menu.

  Note: For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

- The following restrictions apply when naming different API features:

Item	Naming Convention
Action (for tests)	▪ Must be unique in the test. ▪ Cannot begin or end with a space. ▪ Cannot exceed 1,023 characters. ▪ Cannot contain the following characters: **\ / : * ? " < > | % ' ! { }**
Action parameter (for tests)	▪ Case-sensitive. ▪ Must begin with a letter ▪ Cannot contain spaces. ▪ Cannot contain the following characters: **! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ' : " , . / < >**
Checkpoint	▪ Cannot begin or end with a space. ▪ Cannot contain **"** (double quotation mark). ▪ Cannot contain the following character combinations: ○ **:=** ○ **@@**
Component (for components)	▪ Cannot exceed 260 characters. ▪ Cannot contain, begin, or end with spaces. ▪ Cannot contain the following characters: **\ / : " ? < > | * ! { } ' % ;** ▪ Cannot contain multibyte punctuation symbols and other multibyte special characters, such as multibyte question marks, multibyte spaces, and multibyte brackets.

Item	Naming Convention
Data Table file	**ALM:** cannot contain the following characters: **! % * { } \ | ' : " / < > ? ; ,**
Data Table > Parameter name (column header)	▪ Must be unique in the sheet. ▪ Must begin with a letter or underscore. ▪ Can only contain the following: ○ Letters ○ Numbers ○ Periods ○ Underscores
ALM file or folder name	Cannot exceed 90 characters (including the path).
Test name (for tests)	▪ Cannot exceed 260 characters (including the path). ▪ Cannot begin or end with a space. ▪ Cannot contain the following characters: **\ / : * ? " < > | % ' ;** ▪ Cannot contain multibyte punctuation symbols and other multibyte special characters, such as multibyte question marks, multibyte spaces, and multibyte brackets.

Chapter 23: Standard Activities

This chapter includes:

Concepts

Activity Overview

You create a test by double-clicking or dragging **activities** from the Toolbox pane into a canvas. The Toolbox pane provides a collection of built-in standard activities for functional testing in areas such as file and string manipulation, and messaging through HTTP, FTP, and JMS.

The activities are divided into several categories:

- **Standard Activities.** This category includes the built-in activities, such as **String Manipulation**, **Database**, **Network**, **File System**. For a complete list of the standard activities, see "Standard Activity Types" on page 417.
- **Local Activities.** This category includes the activities that are stored as part of the test or business component. These can be imported services such as Web or REST service operations, and .NET assemblies. For details, see "Local Activities" on page 463.
- **File System Activities.** This category includes activities that reside in the file system on either local or network drives. These are Local activities that you moved into the file system repository that can be shared between tests.
- **ALM Activities.** This category includes the activities that reside in the ALM repository. These are Local activities that you moved into the ALM repository that can be shared between tests. This category only appears when a connection to an ALM server is open.

You can also create new custom activities, using the extensibility API provided with the product. These will be stored under the Standard Activities. For details, see "Extensibility in Service Test" on page 661.

For general information about the API interface and creating tests, see " Testing Services" on page 372.

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

1. Drag the **Increment Date/Time** activity onto the canvas.
2. Click the Input/Checkpoints tab in the Properties pane.
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 **Checkpoints** 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 onto the canvas.
2. Click the Input/Checkpoints tab in the Properties pane.
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 **Checkpoints** pane. Use the scroller to specify the expected difference.

Format a date or time

1. Drag the **Format Date/Time** activity onto the canvas.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Set the property **Input Date/Time**, the term to which you want to apply formatting. Click the arrow in the **Value** column to open the calendar. Click the date displayed at the top of the calendar window to use the current date, or select another date.
4. If you provide a date/time in a string (non-date) format, select a format in the **Input Date/Time Format** row. Otherwise, leave this row empty.
5. Set the **Format** input property. Select a format from the drop down or type in a custom format for the output expression. If your expression contains milliseconds, use 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 **Checkpoints** 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 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 420.

This task includes:

- "Open a connection" below
- "Add a Select Data step" on next page
- "Add an Execute Command step" on next page
- "Add database transaction activities" on next page
- "Add a Close Connection step" on page 401

Open a connection

This step is mandatory for all of the following steps.

1. Drag the **Database > Open Connection** activity onto the canvas.
2. Click the Input/Checkpoints tab in the Properties pane.
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 453.
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 onto the canvas, below an **Open connection** step.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Set the Input property—**Connection**. Click the **Link to a data source** button by the right side of the **Value** column . In the Select Link 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 455.
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 onto the canvas, below the **Open Connection** step.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Set the Input property—**Connection**. Click the **Link to a data source** button by the right side of the **Value** column . In the Select Link 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 "The Canvas" on page 373.

Add a Close Connection step

1. Drag the **Database > Close Connection** activity on to the canvas, after all of the database steps.
2. Click the Input/Checkpoints tab in the Properties pane.
3. Set the Input property—**Connection**. Click the **Link to a data source** button in the right side of the **Value** column . In the Select Link 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 Send a Multipart HTTP Request

The following describes how to send an HTTP request that uses multiple parts. For details about the properties, see the "Multipart Tab (Properties Pane)" on page 207.

This task includes:

- "Add an HTTP Step" below
- "Set the General properties" below
- "Set the properties for the first part" below
- "Set the properties for the next parts" below

1. ### Add an HTTP Step

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

2. ### Set the General properties

   Open the General tab in the Properties pane. Set the properties as described in the "General Tab (Properties Pane)" on page 195.

3. ### Set the properties for the first part

   Open the Input/Checkpoints tab in the Properties pane. Set the properties for the first part as described in the "Network Activities" on page 426.

4. ### Set the properties for the next parts

   a. Open the Multipart tab in the Properties pane.

   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 Tab (Properties Pane)" on page 207.

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

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 enables you to incorporate existing Java code into your test.

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

1. ## Set the global Java settings - optional

   To set global VM (Virtual Machine) settings, select the Start or End steps in the canvas, and open the **Test Settings** view  in the Properties pane. For details, see the "Test Settings Tab (Properties Pane)" on page 209.

2. ## Implement the Java interface

   Change to the `<installation_folder>\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java` folder and create an implementation for the java interface. For an example, see the `sample` subfolder.

   This interface includes the essential information for the Java call, such as input properties, output properties, and a point of entry.

3. ## Compile the Java source code

   Compile the java files located in the `<installation_folder>\addins\ServiceTest\JavaCall\Java Interface\src\hp\st\ext\java` folder.

   > **Tip:** To determine which JDK to use for, check the version of Java JRE installed with Service Test. Open the `<installation_folder>/jre/bin` folder and right-click the `java.exe` file. Select **Properties** and open the **Version** tab.

4. ## 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.

5. ## Add a Call Java Class activity

Expand the **Java** node in the Toolbox pane, and drag the **Call Java Class** activity onto the canvas.

6. **Open the Java Class Setting dialog box**

   Select the Java step in the canvas, and click the Input/Checkpoints tab in the Properties pane. Click the **Java Class** button to open the "Java Class Dialog Box" (described on page 459.

   If you need to embed the jar in the script, you must select the check box before browsing for and selecting the class file.

   For general guidelines, see "Java Testing" on page 379.

7. **Set the Java Call properties**

   In the Java Class dialog box:

   a. Provide a classpath. If you packaged your Java step, click the **Browse** button adjacent to the **Jar** field and point to a `.jar` file. Alternatively, click the **Browse** button adjacent to the **Package root** field and point to a package root folder. To embed the jar and save it with the test, select **Embed Jar in Test**. Due to a technological limitation, if you intend to specify a class file, you must select the **Embed Jar in Test** option before you browse for the class file.

   b. Click the **Browse** button adjacent to the **Class file** field 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 and 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 "Java Class Dialog Box" on page 459.

8. **Provide Input property values**

   In the Properties pane's Input/Checkpoints tab , provide values for the step's input properties.

How to Retrieve Messages from a JMS Queue

This task describes how to send, receive and browse JMS messages on a JMS queue.

This task includes the following steps:

- "Set checkpoints- optional" on page 406
- "Run the test and view the run results" on page 407

1. ## Define the global JMS settings

   Define global JMS settings such as the JNDI details according to the specifications of your JMS messaging server. Select the **Start** or **End** step in the canvas and open the **Test Settings**  view in the Properties pane. For details, see the "Test Settings Tab (Properties Pane)" on page 209.

2. ## Prepare a message to send to the queue

   Prepare the message you want to send to the queue in one of the following ways:

   - Prepare an expression and type it into the **Message** property area.
   - Use other activities to generate the string, for example **Concatenate String**, **Trim String**, and so forth.

3. ## Add a Send Message step to your test

   a. In the Toolbox pane, expand the **Standard Activities > JMS** node. Drag a **Send Message to JMS Queue** or a **Send and Receive Message from JMS Queue** activity onto the canvas.

   b. Select the step and click the Properties pane's **Input/Checkpoints** tab. Set the send-related input properties:

      - **Send queue name**
      - **Message**
      - **JMS send properties**

   c. If you used another step to generate the message text, link its output to the **Message** property.

4. ## Add a Send Message with security and attachments - optional

   To send a Web service message via JMS with attachments and security setting, such as tokens and signatures:

   a. Drag in or select the Web service step in the canvas and disable its **Send Request to Service** option in the Properties pane. For details, see "Properties Pane Tabs" on page 186.

   b. Set the security options for the service. For details, see "How to Set Security for a Web Service on the Port Level" on page 547.

   c. Include any attachments. For details, see "Attachments Tab (Properties Pane)" on page 187.

   d. Drag in a Send Message to JMS Queue activity onto the canvas.

   e. In the Properties pane's Input/Checkpoints tab, select the link icon  for the **Message** property. The Select Link Source dialog box opens. For details, see the "Select Link Source Dialog Box" on page 630.

f. In the Select Link Source dialog box, select the Web service step in the left pane. In the Output property section, double-click on the **Raw Request** property. This attaches the security and attachment data to the message.

g. Define any other JMS properties as you normally would. For property details, see "JMS Activities" on page 431.

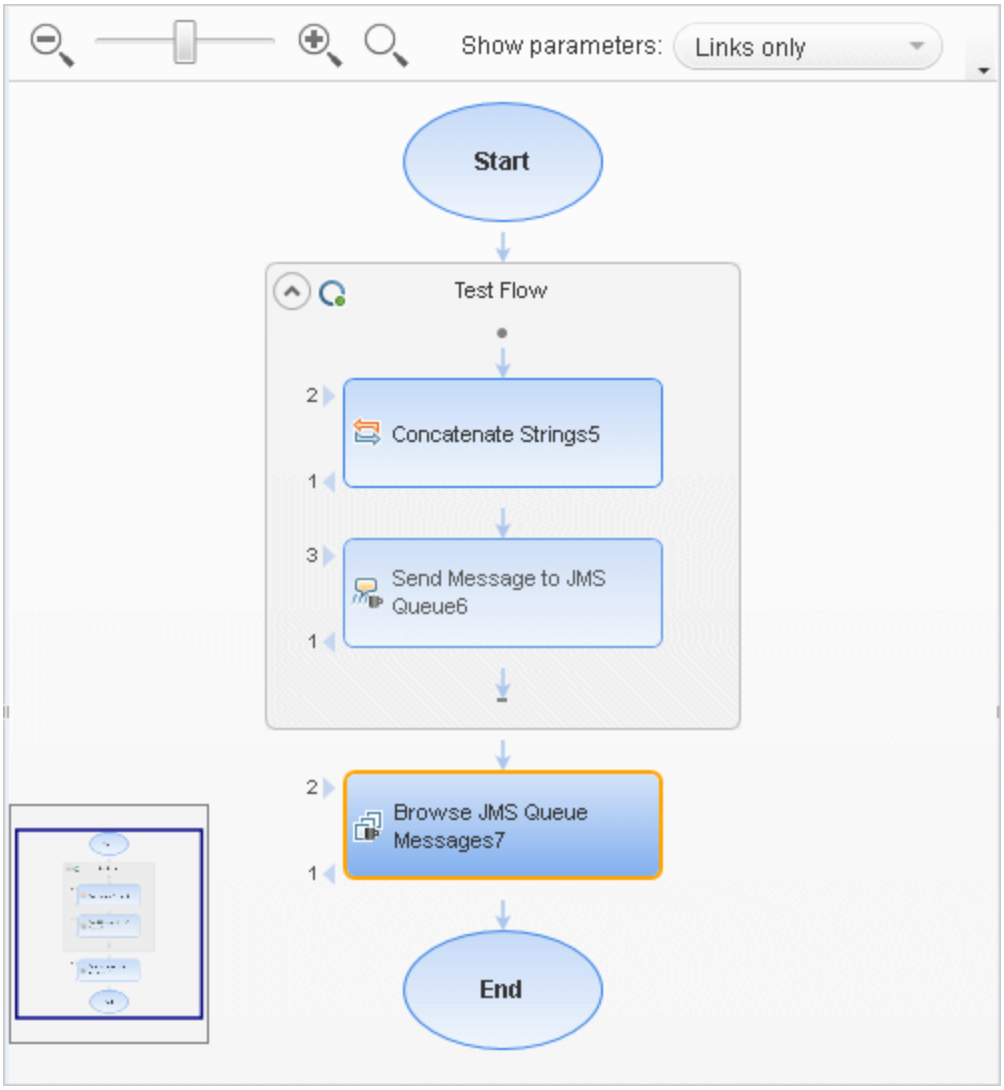
5. ## Add a Receive Message step to your test - optional

a. To retrieve a message from the queue, drag a **Receive Message from JMS Queue** activity onto the canvas. If you added a **Send and Receive Message to JMS Queue** activity in the previous step, you can skip this step.

b. Select the **Receive Message from JMS Queue** step and open the Properties pane's Input/Checkpoints tab. Set the receive-related input properties: **Receive queue name** and **JMS receive message selector**. For property details, see "JMS Activities" on page 431.

6. ## Browse the message queue - optional

a. To browse the messages in the queue without consuming them, drag a **Browse JMS Queue Messages** activity onto the canvas, outside of the Test Flow. Make sure that a **Send (and Receive) Message from JMS Queue** step precedes the **Browse JMS Queue Messages** step.



b. Click the Properties pane's Input/Checkpoints tab, and select the **Browse JMS Queue Messages** step. Set the browse-related input properties:

- **Queue name.** Provide a queue name or link to the queue name from an earlier step. For IBM Websphere's MQ, if you specify a queue name that does not exist, a new queue will be created during the test run.
- **JMS receive message selector.** The selector lets you filter the messages list. For details about these properties, see "JMS Activities" on page 431.

7. ## Set checkpoints- optional

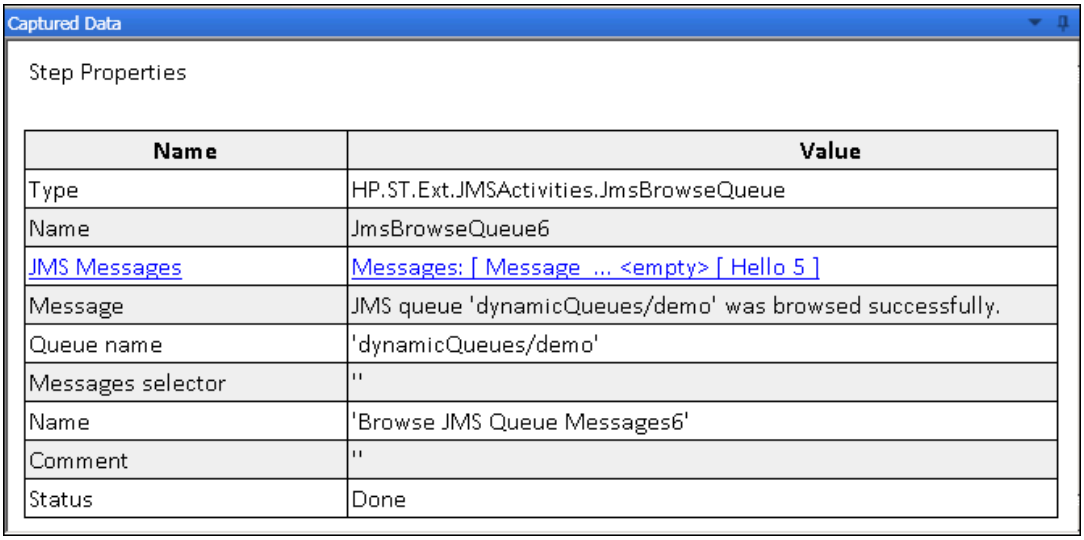
a. Select the **Browse JMS Queue Messages** step and click in the Properties pane's **Checkpoints** section.

b. Select the properties you want to validate and provide values.

c. Use the **Browse JMS queue messages** output properties to validate the messages on the queue. For example, you can check the value of a specific property or check the number of messages on the queue.

8. **Run the test and view the run results**

   Run the test. In the Run Results Viewer, expand the checkpoint nodes and verify that the actual values match the expected values. Click the link in the report to display the JMS queue messages in a separate browser. Compare these messages with those shown in the queue on the JMS application server's console.


Captured Data

Step Properties

Name	Value
Type	HP.ST.Ext.JMSActivities.JmsBrowseQueue
Name	JmsBrowseQueue6
JMS Messages	Messages: [Message ... <empty> [Hello 5]
Message	JMS queue 'dynamicQueues/demo' was browsed successfully.
Queue name	'dynamicQueues/demo'
Messages selector	''
Name	'Browse JMS Queue Messages6'
Comment	''
Status	Done

How to Receive Messages Through JMS Topics

This task describes how to work with subscriptions to JMS topics.

This task includes the following steps:

- "Set the global JMS settings" below
- "Subscribe to a topic" below
- "Publish a message to the topic" below
- "Include Web service security and attachments - optional" on next page
- "Receive the message through the topic" on next page

1. **Set the global JMS settings**

   Set global JMS settings such as the JNDI details according to the specifications of your JMS messaging server. Click in the canvas outside of the Test Flow, and open the **Test Settings** tab in the Properties pane. For details, see the "Test Settings Tab (Properties Pane)" on page 209.

2. **Subscribe to a topic**

   Add a **Subscribe to JMS Topic** step and provide values for its input properties as described in "JMS Activities" on page 431.

3. **Publish a message to the topic**

   Add a **Publish Message to JMS Topic** step and provide values for its input properties.

4. **Include Web service security and attachments - optional**

   To publish a Web service message via JMS with attachments and security setting, such as tokens and signatures:

   a. Drag in or select the Web service step in the canvas and disable its **Send Request to Service** option in the Properties pane. For details, see "Properties Pane Tabs" on page 186.

   b. Set the security options for the service. For details, see "How to Set Security for a Web Service on the Port Level" on page 547.

   c. Include any attachments. For details, see "Attachments Tab (Properties Pane)" on page 187.

   d. Drag a **Publish Message to JMS Topic** step onto the canvas.

   e. In the Properties pane's Input/Checkpoints tab, select the link icon for the **Message** property. The Select Link Source dialog box opens. For details, see the "Select Link Source Dialog Box" on page 630.

   f. In the Select Link Source dialog box, select the Web service step in the left pane. In the Output property section, double-click on the **Raw Request** property. This attaches the security and attachment data to the message.

   g. Define any other JMS properties as you normally would. For property details, see "JMS Activities" on page 431.

5. **Receive the message through the topic**

   Drag a **Receive Message from JMS Topic** activity onto the canvas. Use the topic and subscription name that you defined in the previous steps.

How to Retrieve Messages from an MQ Queue

This task describes how to work with Put and Get on an IBM Websphere MQ queue.

This task includes the following steps:

1. **Prerequisite**

   You must have the MQ client installed on all machines upon which you want to run the test.

2. **Connect to the MQ Queue Manager**

   A connection step must precede all steps that use the specified Queue Manager. Drag an **IBM Websphere MQ > Connect to MQ Queue Manager** step onto the canvas and provide the connection details in the Properties pane's Input/Checkpoints tab. For details see "Connect to MQ Queue Manager - Input Properties" on page 438.

3. **Put a message on the queue**

   Drag a **Put Message to MQ Queue** step onto the canvas, below the connection step. The canvas automatically links the **MQManager** property to the most recent **Connect to MQ Queue Manager** step. Set input values as described in "Put Messages to MQ Queue - Input Properties" on page 441.

4. **Browse the message queue - optional**

   To browse the messages in the queue without consuming them:

   a. Drag a **Browse Messages in MQ Queue** activity onto the canvas.

   b. Select the step in the canvas and open the Properties pane's **Input/Checkpoints** tab.

   c. Set the browse-related properties. For details, see "Browse Messages in MQ Queue - Input Properties" on page 440.

5. **Get a message from the queue**

   Drag a **Get Message from MQ Queue** step onto the canvas, below the connection step. Set the input values as described in "Get Messages from MQ Queue - Input Properties" on page 442.

6. **Commit or Backout the actions - optional**

   To commit the actions performed until a certain point, drag a **Commit MQ Pending Messages** activity onto the canvas. To roll back changes drag a **Backout MQ Pending Messages** activity onto the canvas at the relevant location.

7. **Disconnect from the MQ Queue Manager**

   A disconnection step must follow all steps that use the specified Queue Manager. Drag a **Disconnect from MQ Queue Manager** step onto the canvas, and provide the connection details in the Properties pane's **Input/Checkpoints** tab. For details see "Disconnect from MQ Queue Manager - Input Properties" on page 439.

8. **Define event handlers - optional**

   To customize or automate the actions against the MQ server, use the MQ event handlers. Click the Events button in the Properties pane and double-click the appropriate event—a standard event or one of the MQ-specific events, such as `BeforeMQGetMessage` or `AfterMQGetMessage`. Customize the code in the `TestUserCode.cs` tab. For details, see "Coding Service Test Events" on page 641 and the "Events Tab (Properties Pane)" on page 193.

9. **Set checkpoints - optional**

To validate the response data, select the step and click within the Properties pane's **Checkpoints** section. Indicate the properties you want to validate and provide values. Run the test and view the run results.

How to Receive Messages Through MQ Topics

This task describes how to retrieve messages published to MQ topics.

This task includes the following steps:

1. ### Prerequisite

   You must have the MQ client installed on all machines upon which you want to run the test.

2. ### Connect to the MQ Queue Manager

   A connection step must precede all steps that use the specified Queue Manager. Drag a **Connect to MQ Queue Manager** step onto the canvas, and provide the connection details in the Properties pane's **Input/Checkpoints** tab. For details see "Connect to MQ Queue Manager - Input Properties" on page 438.

3. ### Subscribe to a topic

   Add a **Subscribe to MQ Topic** step. The canvas automatically links the **MQManager** property to the most recent **Connect to MQ Queue Manager** step. Provide values for its input properties as described in "Subscribe to MQ Topic - Input Properties" on page 446.

4. ### Publish a message to the topic

   Add a **Publish Message to MQ Topic** step and provide values for its input properties as described in "Publish Message to MQ Topic - Input Properties" on page 447.

5. ### Receive the message through the topic

   Drag a **Receive Message from MQ Topic** activity onto the canvas. Use the topic and subscription name that you set in the previous steps. For details, see "Receive Message from MQ Topic - Input Properties" on page 448.

6. ### Unsubscribe from a topic

   Add an **Unsubscribe from MQ Topic** step and provide values for its input properties as described in "Unsubscribe from MQ Topic - Input Properties" on page 446.

7. **Disconnect from the MQ Queue manager**

   A disconnection step must follow all steps that use the specified Queue Manager. Drag a **Disconnect from MQ Queue Manager** activity onto the canvas. The canvas automatically links to the most recent opened connection.

8. **Define event handlers - optional**

   To customize or automate the actions against the MQ server, use the MQ event handlers. Click the **Events** button in the Properties pane and double-click the appropriate event—a standard event or one of the MQ-specific ones. Customize the code in the `TestUserCode.cs` tab. For details, see "Coding Service Test Events" on page 641 and the "Events Tab (Properties Pane)" on page 193.

How to Call Tests from Other Applications

This task describes how to incorporate tests from other HP applications to provide a unified testing solution. These applications include HP Unified Function Testing, earlier versions of Service Test and QuickTest Professional, and LoadRunner's Virtual User Generator. For details, see "Automated Testing Tool Integration" on page 380.

This task includes the following steps:

1. **Prerequisites**

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

   - **Service Test.** To call a test last modified in a previous version of Service Test, make sure the test was last modified with Service Test 11.10 or higher.
   - **LoadRunner's Virtual User Generator.** Make sure you have HP LoadRunner version 11.00 or later. Generate a test with the Virtual User Generator, or use a test that you enabled for load testing in this application.

2. **Call an API Test action or test**

   a. Make sure the action or test you want to call has been saved and run successfully at least once.

   b. Expand the **HP Automated Testing Tools** node in the Toolbox pane. Drag the **Call API Action or Test** activity onto the canvas.

   c. Open the Properties pane's Input/Output Properties tab and click the **Select Action or Test** button. Select a test last modified with Service Test 11.10 or higher.

   d. To update a test or action that has already been loaded, click **Refresh**.

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

   f. Click the Properties pane's Input/Output Properties tab and set the property values, if needed.

Note:

- The property list remains empty until you select a test.
- If the test you are calling has no input or output parameters, the Input/Output Properties tab will be empty.

g. Add other relevant steps to your test. You can link subsequent input properties to the output properties of the step. Click the **Link to data source** button in the property's row to create the link. For details, see "Select Link Source Dialog Box" on page 630.

h. If the API Test step input must be a string (when the result of a previous step was XML), add an **XML to String** activity after the call to the Service Test action or test. For details, see "XML Activities" on page 451.

i. Save and run the test.

3. Add a LoadRunner script activity

a. In the Toolbox pane, expand the **HP Automated Testing Tools** node. Drag the **Call Virtual User Generator Script** activity onto the canvas.

b. Open the Properties pane's General view, and click the script selection button . Select a VuGen (Virtual User Generator) `.usr` script file.

c. Add other relevant steps to your test.

d. Save and run the test.

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" below
- "Add a Validate XML activity" on next page
- "Connect the steps - if using Read from File" on next page
- "Specify an XSD file" on next page
- "Set a checkpoint - optional " on next page
- "Run the test" on next page

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 onto the canvas.

b. Click the Input/Checkpoints tab in the Properties pane. Browse to the XML file you want to validate.

2. ### Add a Validate XML activity

Drag the **XML > Validate XML** activity onto 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. Click the Input/Checkpoints tab in the Properties pane. Click in the **Value** column of the **XML String** property and click the **Link to a data source** button .

b. In the Select Link 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. Click the Input/Checkpoints tab in the Properties pane, 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 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. In the Input/Checkpoints tab in the Properties pane, 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:

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 onto the canvas.

   b. Click the Input/Checkpoints tab in the Properties pane. Browse to the `.xml` file you want to transform.

2. ## Add a Transform XML activity

   Drag the **XML > Transform XML** activity onto 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 the next step.

   a. Click the Input/Checkpoints tab in the Properties pane. Click in the **Value** column of the **XML String** property and click the **Link to a data source** button .

   b. In the Select Link 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 XSLT file

   a. Select the **Transform XML** step in the canvas. Click the **Input/Checkpoints** tab in the Properties pane. Browse to an `.xslt` file.

   b. To import the `.xslt` file into the test's folder, set the **XSLT file to test folder** option to `true`. This is useful if you plan on saving the test on ALM. Setting this option will make the `.xslt` 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 Compare XML Strings

This task describes how to compare two XML strings and view the differences.

This task includes the following steps:

1. **Add an Compare XMLs step to the canvas**

   Drag the **XML > Compare XMLs** activity onto the canvas.

2. **Enter the original XML string**

   Copy the original XML string onto the clipboard.

   Click the Input/Checkpoints tab in the Properties pane. Click the down arrow in the **Value** column of the `XML string 1` property and paste the original XML string into the box.

3. **Enter the new XML string**

   Copy the modified XML onto the clipboard and paste it into the `XML string 2` row.

4. **Set the ignore options - optional**

   a. To ignore any differences in the order of the elements, set **Ignore element order** to `true`.

   b. To ignore any differences in namespaces, set **Ignore namespaces** to `true`.

   c. To ignore differences in a specific node, expand the **Ignore XPaths** array and add one or more array elements with the XPath expression you want to ignore.

5. **Save the settings - optional**

   To save all of your ignore settings for future use, click the **Save options** button.

6. **Set a checkpoint - optional**

   To set a checkpoint which indicates whether or not there was a change, click the **Validate** check box in the Checkpoints area.

- To verify that there was a change, set **Are Equal** to `false`.
- To verify that there was no change, set **Are Equal** to `true`.

7. **View the results**
   a. Run the test. In the Run Results Viewer, expand the results.
   b. Select the **Compare XMLs** node in the results tree in the left pane.
   c. In the **Captured Data** pane, click the **Report File** link. The report shows both XML strings. The report's legend indicates the nature of the change: added, removed, changed, and so forth.
   d. Scroll down in the **Captured Data** pane for the XML Comparison Results table. This table lists the changes, showing the old and new element and attribute names with their values.

References

Standard Activity Types

This section describes the Standard activity groups:

Flow Control Activities

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

- **Condition.** Enables 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.
- **Wait.** Waits a specified amount of time for a trigger event before releasing a step for execution

For details, see "The Canvas" on page 373.

Loop Activity

Loop - Input Properties

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

UI Elements	Description
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.
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, it 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).
For Loop	Performs the Test Flow or custom loop the number of times that you specify in the **Number of Iterations** box.

Wait Activity

For asynchronous steps, the **Wait** activity waits a specified amount of time for a trigger event before releasing the associated step. This activity is used in conjunction with the **HTTP Receiver** steps and Web Service calls imported as server response steps. For details, see "Asynchronous Service Calls" on page 601.

Wait - Input Properties

To view the Input properties, click the Input/Checkpoints tab in the Properties pane.

Property (A-Z)	Description
Action on timeout	The action to take when the timeout is reached without the completion events: **Fail** or **Continue**.
Completion event names	A list of the completion events that were configured in prior receiver steps.
Start timeout after step	The time after step execution from when to begin measuring the timeout interval.
Timeout	The timeout in milliseconds. Negative values indicate that there is no timeout.

Miscellaneous Activities

This activity group provides access to the following activities:

- **Custom Code.** Enables 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 on the current 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. You can provide an array of replacement strings for use with iterations.
- **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 system 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 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.

Open/ Close Connection

This activity opens or closes a connection to a database using the specified connection string.

Open/ Close Connection - Input Properties

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 453. • 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.

Open/ Close Connection and Transaction Activities - Checkpoints

Property (A-Z)	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

This activity executes a SELECT type SQL statement that retrieves data.

Select Data - Input Properties

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 455. • Use the **Link to a data source** button  to create a link to the string.

Property	Description
Timeout	The maximum time to allow for executing the database command. A value of zero indicates no time limit. **Default:** 30 seconds

Select Data - Checkpoints

Property (A-Z)	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: • 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. **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:

Execute Command

This activity executes a SQL statement that does not retrieve data from the database, such as UPDATE, DELETE, and so forth.

Execute Command - Input Properties

Property (A-Z)	Description
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.
Connection	A link to the output of a previously opened connection string. Use the **Link to a data source** button to create the link.

Property (A-Z)	Description
Timeout	The maximum time to allow for executing the database command. A value of zero indicates no time limit. **Default:** 30 seconds

Execute Command - Checkpoints

Property (A-Z)	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.

Begin Transaction

This activity begins a database transaction.

Begin Transaction - Input Properties

Property (A-Z)	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.

Begin Transaction - Checkpoints

Property (A-Z)	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.

Commit/Rollback Transaction

These activities commit a database activity and roll back a database activity from a pending state.

Commit/Rollback Transaction - Input Properties

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.

Commit/Rollback Transaction - Checkpoints

Property (A-Z)	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.

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 Properties pane. The following table describes the FTP-specific properties.

Property (A-Z)	Description
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 579. • 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.
Enable SSL	Indicates whether to establish a secure FTP connection: **true** or **false**.
Proxy	The proxy server information: **Server**, **Username** and **Password**.
Timeout	The FTP request timeout in milliseconds.

FTP - Input Properties

To view the Input properties, click the **Input/Checkpoints** tab in the Properties pane.

Property (A-Z)	Activity	Description
Directory name	**FTP Directory Create**	A name for the new directory.
Directory URL path	**FTP Upload**	The URL in which the file or directory will be created.
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.
Password	All	The password for accessing the FTP server.
Transfer mode	**FTP Download** and **FTP Upload**	The data transfer method: **ASCII** (default) or **Binary**.
URL path	All except **FTP Upload**	The URL of the file or directory on the FTP server.
User ID	All	The username for accessing the FTP server.

FTP - Checkpoints

To view the checkpoint properties, select the Input/Checkpoints tab in the Properties pane and click in the lower **Checkpoints** pane.

Property (A-Z)	Description
File size	The size of the specified file. (**FTP File Get Size** activity)
Result	An indicator whether the operation succeeded: `true` or `false`.
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 enable you to send and receive HTTP and SOAP requests.

The **HTTP Receiver** activity is useful for asynchronous testing. For task details, see "How to Create a Test for an Asynchronous Web Service" on page 605.

This activity group contains the following activities:

- **HTTP Request.** This activity enables you to send an HTTP request over the network.
- **HTTP Receiver.** This activity enables you to receive an HTTP response from a server.
- **SOAP Request.** This activity enables you to send a SOAP request over the network.

HTTP Request Activity

The HTTP Request activity enables you to send an HTTP request over the network.

HTTP Request - General Properties

To view the General properties, click the **General** view button in the Properties pane.

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** or **Close**. **Default:** `Keep-Alive`
Timeout	The HTTP timeout in milliseconds.

Property	Description
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. **Note:** To use a certificate, make sure to the **Use Client Certificate** property to `true`.
Use Client Certificate	Enables the use of a client certificate. **Default:** `false`
Maximum automatic redirections	The number of times to attempt accessing a page, including redirection. **Default:** `3`
Allow redirections	Indicates whether to allow the step to redirect to another URL. **Default:** `true`
Reuse cookies	Enables the reusing of cookies for the current step. **Default:** `false`.

HTTP Receiver Activity

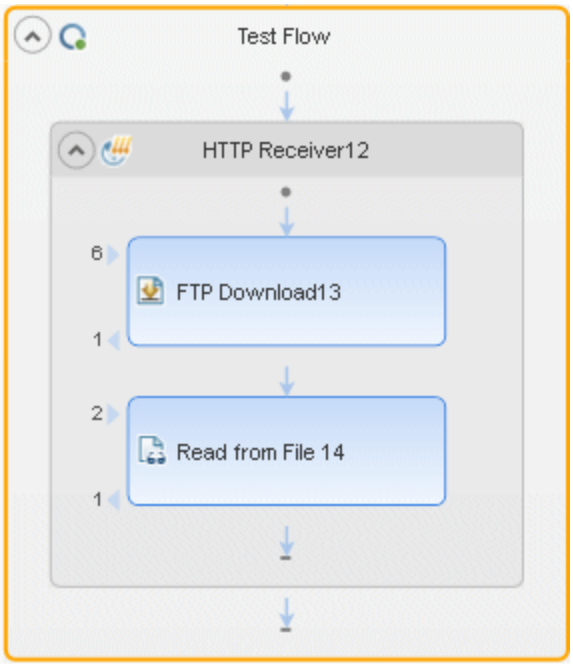
Overview

The **HTTP Receiver** activity receives an HTTP message from a server. This activity is used with asynchronous messaging. For details, see "Asynchronous Service Calls" on page 601.

Receiver activities are activities that act as receiver for a server response. This category includes the built-in **HTTP Receiver** activity and Web Service operations that were imported as a server response. For more details, see the "Select WSDL Dialog Box" on page 488 or "Select WSDL Dialog Box" on page 488.

All receiver activities are composite activities. This means that they serve as a wrapper for all activities inside the container whose responses are sent to the HTTP Receiver step—not the Test Flow.

For example, if an HTTP Receiver step contains the **FTP Download** and **Read from File** steps, these internal steps send their response to the HTTP Receiver step—not to the Test Flow loop.



HTTP Receiver - General Properties

To view the General properties, click the **General** view button in the Properties pane.

Property	Description
Listen on port	The port upon which to listen for the server request.
Completion event name	The name of the event associated with a following **Wait** step, allowing the step to receive the server request.
Use SSL	Secures the connection with SSL: **true** or **false.**
SSL Certificate	Information about the SSL certificate: • **Subject.** The certificate's subject string. • **Store location.** The location of the certificate—LocalMachine or Windows store. • **Store name.** The name of the store hosting the certificate. • **Thumbprint.** The certificate's thumbprint or hash code.
Save response body with this extension	Saves the response body for messages with the specified extension.

HTTP Receiver - Output Properties

You can link to the **HTTP Receiver** output property, its response body, through the Select Link Source dialog box. For details, see "Select Link Source Dialog Box" on page 630.

Property	Description
Response body as a UFT-8 string	The body of the HTTP response in UFT-8 format.

SOAP Request Activity

The **SOAP Request** activity 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/Output Properties Tab (Properties Pane)" on page 201.

SOAP Request - General Properties

To view the General properties, click the **General** button in the Properties pane.

Property	Description
Transport	The type of transport for the SOAP request: **HTTP** or **JMS**.
Transport > HTTP	Use the HTTP transport method for this call with the following properties: • **Endpoint Address.** The endpoint location of the service as derived from the WSDL • **SoapAction.** An expression indicating the SOAP action **Example:** `HP.SOAQ.SampleApp/IHPFlights_Service/DeleteFlightOrder.` • **ContentType.** The type of content: For example `text/xml; charset=utf-8`. • **Timeout.** The maximum time in milliseconds to wait for the response. Enter -1 to disable the timeout and wait for the response as long as required.
Transport > JMS	Use the JMS transport method for this call with the following properties: • **Send queue name.** The name of the queue from which to send the message. • **Receive queue name.** The name of the queue from which to receive the message. • **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://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html. For example, a key `JMSCorrelationID`, with a value of `4566636`, receives only messages whose correlation ID is 4566636.
IsOneWay	Indicates whether the service is a one way service: `true` or `false`. A one way service only sends a request. A non-one way service sends a request and receives a response.

SOAP Request - Input Properties

To view the Input properties, click the **XML Body** button in the Properties pane. The Input properties have the form of a SOAP envelope schema, with a header and body. You can add array elements at all levels using the shortcut menu.

SOAP Request - Checkpoints

To view the checkpoints, click the **XML Body** button in the Properties pane. You can view the expected output schema, or display a list of the XPath checkpoints. You can add array elements at all levels using the shortcut menu.

For details, see the "XML Body Tab (Properties Pane)" on page 213.

JSON Activities

This activity group contains activities related to JSON files:

- **JSON to String.** Converts a JSON file to a text string.
- **String to JSON.** Converts a string to a JSON structure.

This **JSON to String** and **String to JSON** activities are useful when you need to access the output in a specific format—either as string or JSON.

For example, suppose your test contains a Web Service call whose output is JSON. However, you need to use the output as an input for the next step, such as aService Test test step, which requires a textual string—not JSON.

You can use the **JSON to String** activity to create a string that is compatible with the Service Test test step.

After the Service Test test step, you can use a **String to JSON** activity to convert the output to JSON, making it available for linking and checkpoints.

JSON - Input Properties

To view the Input properties, click the Input/Checkpoints tab in the Properties pane.

Property (A-Z)	Activity	Description
<JSON document>	**JSON to string**	The JSON document to convert to a string. This can be displayed in Text or Grid views. To add a document, click **Load JSON** to import a JSON file.
Source string	**String to JSON**	The string to convert to JSON.

JSON - Checkpoints

To view the checkpoint properties, select the **Input/Checkpoints** tab button in the Properties pane and click in the lower **Checkpoints** pane.

Property (A-Z)	Activity	Description
<JSON document>	**String to JSON**	The JSON representation of the expected response. To see the JSON document, click **Load JSON** to import a JSON file.

Property (A-Z)	Activity	Description
Output string	**JSON to String**	The expected string representation of the conversion. **Tip:** To view the actual string: • See the **Captured Data** pane in the Run Results Viewer. • Select **Run Step** from the step's right-click menu and view the Run Step Results pane.

The **String to JSON** step contains an additional output property, **Result**, visible from the **Result** view in the Properties pane or Select Link Source dialog box. For details, see "Result Tab (Properties Pane)" on page 207.

Property	Description
Result	A boolean variable indicating whether the string had the correct structure to be converted into JSON.

Java Activities

This activity group contains the **Call Java Class** activity. It enables you to set up and configure a call to a Java class.

For task details, see " How to Create a Call to a Java Class" on page 402.

For user interface details, see the "Java Class Dialog Box" on page 459.

JMS Activities

This activity group provides steps that handle JMS messages. For details, see the "JMS Transport Overview" on page 382.

For task details, see "How to Retrieve Messages from a JMS Queue" on page 403 or "How to Receive Messages Through JMS Topics" on page 407.

This activity group contains the following activities:

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

- **Subscribe to JMS Topic.** Creates a subscription for a JMS topic. Use this step before any **Receive Message from Topic** steps for the subscription.
- **Browse JMS Queue Messages**. Retrieves messages from the JMS queue without consuming them.

Publish Message to JMS Topic

This activity publishes a message to multiple subscribers using a JMS topic.

Publish Message to JMS Topic - Input Properties

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

Publish Message to JMS Topic - Checkpoints

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Receive Message from JMS Queue

This activity receives a message from a JMS queue.

Receive Message from JMS Queue - Input Properties

Property (A-Z)	Description
JMS receive message selector	An SQL expression to filter the received message. For details on the syntax for defining selectors, see the **Message Properties** section in http://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html. For example, a key `JMSCorrelationID`, with a value of `4566636`, receives only messages whose correlation ID is `4566636`.
Receive queue name	The name of the queue from which to receive the message. If the value is empty, it uses a temporary queue.

Receive Message from JMS Queue - Checkpoints

Property (A-Z)	Description
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.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSDestination	The message's destination.
JMSExpiration	The message's expiration date.
JMSMessageID	A unique ID for the message.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. `False` by default.
JMSReplyTo	The contents of the **Reply To** field.
JMSTimeStamp	The time and date of the message.
JMSType	A string describing the JMS type.
Message body	The message content.

Receive Message from JMS Topic

This activity receives published messages to a specific JMS topic for a subscription. Place this step after a **Subscribe to JMS Topic** step.

Receive Message from JMS Topic - Input Properties

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

Receive Message from JMS Topic - Checkpoints

Property (A-Z)	Description
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.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSDestination	The message's destination.
JMSExpiration	The message's expiration date.
JMSMessageID	A unique ID for the message.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. `False` by default.
JMSReplyTo	The contents of the **Reply To** field.
JMSTimeStamp	The time and date of the message.
JMSType	A string describing the JMS type.
Message body	The message content.

Send Message to JMS Queue

This activity sends a message to a JMS queue.

Send Message to JMS Queue - Input Properties

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

Send Message to JMS Queue - Checkpoints

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Send and Receive Message from JMS Queue

This activity sends a message to a specified queue and receives a message from a specified queue.

Send and Receive Message from JMS Queue - Input Properties

Property (A-Z)	Description
JMS receive message selector	An SQL expression to filter the received message. For details on the syntax for defining selectors, see the **Message Properties** section in http://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html. 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 Properties pane's **Test Settings** tab. This selector uses a correlation ID that corresponds to the request's message ID.
JMS send properties	JMS properties with the key/value form. You can use the key to modify JMS header or message properties.
Message	The message to send.
Receive queue name	The name of the queue from which to receive the message. If the value is empty, it uses a temporary queue.
Send queue name	The name of the queue to which to send the message.

Send and Receive Message from JMS Queue - Checkpoints

Property (A-Z)	Description
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.
JMSCorrelationID	The message's correlation ID. **Tip:** Use in conjunction with the JMS receive message selector.
JMSDeliveryMode	The message's delivery mode. Use the scroller to select a value.
JMSDestination	The message's destination.
JMSExpiration	The message's expiration date.

Property (A-Z)	Description
JMSMessageID	A unique ID for the message.
JMSPriority	The message's priority in comparison to other messages. Use the scroller to select a value.
JMSRedelivered	Indicates whether the message was redelivered. `False` by default.
JMSReplyTo	The contents of the **Reply To** field.
JMSTimeStamp	The time and date of the message.
JMSType	A string describing the JMS type.
Message body	The message content.

Subscribe to JMS Topic

This activity creates a subscription for a JMS topic. Use this step before any **Receive Message from Topic** steps for the subscription.

Subscribe to JMS Topic - Input Properties

Property (A-Z)	Description
JMS receive message selector	An SQL expression to filter the received message. For details on the syntax for defining selectors, see the **Message Properties** section in http://docs.oracle.com/javaee/1.3/api/javax/jms/Message.html.
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.
Topic name	The name of the topic from which to receive the message.

Subscribe to JMS Topic - Checkpoints

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Browse JMS Queue Messages

This activity retrieves messages from the JMS queue without removing them from the queue.

Browse JMS Queue Messages - Input Properties

Property (A-Z)	Description
Messages selector	An optional SQL expression to filter the received messages. For details on the syntax for defining selectors, see the **Message Properties** section in http://www.oracle.com/technetwork/java/jms-101-spec-150080.pdf.
Queue name	The name of the queue upon which to browse for a collection of messages. This field is mandatory. **Note:** When using IBM ActiveMQ, you must use the format `dynamicQueues/<queue_name>`. If the specified queue does not exist, it will be created dynamically.

Browse JMS Queue Messages - Checkpoints

Property	Description
JMS messages	An array of JMS messages and their properties on the specified queue. The array of messages corresponds to the filter used in the **Messages selector** property.

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 594.

IBM WebSphere MQ Activities

This activity group enables you to perform actions on an IBM Websphere MQ application server.

The activities in this group are:

- **Connect to MQ Queue Manager.** Connects to a specific MQ Queue Manager and maintains an open connection with the server.
- **Disconnect from MQ Queue Manager.** Disconnects from the specific MQ Queue Manager and closes the connection with the MQ server.
- **Commit MQ Pending Messages.** Commits the last changes made for all messages since the last point of synchronization.
- **Backout MQ Pending Messages.** Performs a Backout operation from the last point of synchronization. All PUT messages will be cancelled and GET messages will be reinstated to the queue.

- **Browse Messages in MQ Queue.** Browses the messages on the MQ Queue via the Queue Manager.
- **Put Message to MQ Queue.** Puts a message on the MQ queue via the Queue Manager.
- **Get Message from MQ Queue.** Gets the most recent message from the MQ queue via the Queue Manager.
- **Put and Get Message from MQ Queue.** Puts a message on the MQ queue and then gets a message from the MQ queue via the Queue Manager.
- **Subscribe to MQ Topic.** Creates a subscription for an MQ topic. From this point on, a subscriber can receive messages from a specified topic.
- **Unsubscribe to MQ Topic.** Cancels a subscription for the specified MQ topic.
- **Publish Message to MQ Topic.** Publishes a message to subscribers using an MQ topic.
- **Receive Message from MQ Topic.** Receives messages for the active subscription, that were published to a specific MQ topic.

Service Test enables you to set the data format of the message for both queues and topics. The **String** format indicates Unicode and non- Unicode strings without a prefix. The **UTF** format represents a UTF string with a 2-byte prefix.

For details about each of the activities and a description of their input and output properties, see the sections below.

For task details, see "How to Retrieve Messages from an MQ Queue" on page 408 or "How to Receive Messages Through MQ Topics" on page 410.

Connect to MQ Queue Manager

This activity connects to the specified MQ Queue Manager and maintains an open connection until it is closed or until the end of the test.

Connect to MQ Queue Manager - Input Properties

Property (A-Z)	Description
Channel	The channel through which to connect. For most configurations, use `SYSTEM.DEF.SVRCONN`.
Host name	The name or IP address of the Queue Manager's host machine.
Password	An optional password for connecting to the host machine.
Port	The port through which to connect, by default **1414**.
Queue manager name	An property indicating the name of the Queue Manager.

Property (A-Z)	Description
SSL	Enables or disables SSL. When you enable SSL, you can set values for the following properties: • **SSLCipherSpec.** The SSL Cipher Specification, SSLCIPH. This specification indicates which data encryption algorithm and key size to use, such as `DES_SHA_EXPORT` or `RC2_MD5_EXPORT`. • **SSLKeyRepository.** The path of the SSL key repository on the client machine.
User ID	An optional identification property for connecting to the host machine.

Connect to MQ Queue Manager - Checkpoints

Property (A-Z)	Description
MQManager	An automatically generated connection expression. **Note:** • Although this does not appear in the **Checkpoints** grid, it is visible as an output property in the Select Link Source dialog box. For details, see the "Select Link Source Dialog Box" on page 630. • When you add steps, they automatically use the connection expression from the most recent connection step.
Result	A boolean variable indicating whether the step succeeded.

Disconnect from MQ Queue Manager

This activity disconnects from the specified MQ Queue Manager and closes the connection with the MQ server.

Disconnect from MQ Queue Manager - Input Properties

Property	Description
MQManager	A connection expression linked to a **Connect to MQ Queue Manager** step. **Note:** When you add a **Disconnect to MQ Queue Manager** step, it automatically creates a link to the most recent connection step.

Disconnect from MQ Queue Manager - Checkpoints

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Commit/Backout MQ Pending Messages

These activities commit or rollback the changes made for all messages since the last point of synchronization.

Commit/Backout MQ Pending Messages - Input Properties

Property	Description
MQManager	A connection expression linked to a **Connect to MQ Queue Manager** step. **Note:** When you add this step, it automatically creates a link to the most recent connection step.

Commit/Backout MQ Pending Messages - Checkpoints

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Browse Messages in MQ Queue

This activity browses the messages on the MQ queue via the Queue Manager, without removing them from the queue.

Browse Messages in MQ Queue - Input Properties

Property(A-Z)	Description
Match Conditions	An array of match conditions in a `key/value` format. Select a key from the drop down box in the Key's **Value** column and provide a value. `MQMO_NONE` does not require a value.
Message data format	The format of the message being retrieved: **String** or **UTF**.
Message Get Options	A tree with several built-in MQ Get options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Queue Access Options	A tree with several built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.

Property(A-Z)	Description
Queue name	The name of the MQ queue to browse.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • `-1`: Unlimited wait time • `0`: No wait • A positive number: The time to wait in milliseconds.

Browse Messages in MQ Queue - Checkpoints

Property	Description
Message	An array of the message and its properties on the MQ message queue, that matched the specified conditions. Each array element contains the following message properties: • **MessageId.** The message identifier of the retrieved message. • **CorrelationId.** The correlation identifier of the retrieved message. • **GroupId.** An identifier of the message group to which the physical message belongs. • **MsgSeqNumber.** The sequence number of a logical message within a group. • **Put Time.** The date and time the PUT action was executed for the message. • **Priority.** The priority of the message from 0 to 9. • **UserId.** The UserId of the user who submitted the message. • **MessageBody.** A string representing the UTF content of the retrieved message.

Put Message to MQ Queue

This activity puts a message on the MQ queue via the Queue Manager.

Put Messages to MQ Queue - Input Properties

Property (A-Z)	Description
Correlation ID	The correlation ID of the message.
Message body	The body of the message to put on the queue.

Property (A-Z)	Description
Message data format	The format of the message being sent: **String** or **UTF**.
Message Properties	An array of Put message properties in a `Key/Value` format.
Message Put Options	A drop down of built-in message put options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
Put queue name	The name of the MQ queue on which to put the message.
Queue Access Options	A drop down of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.

Put Messages to MQ Queue - Checkpoints

Property (A-Z)	Description
MessageId	A message identifier of the message being sent.
PutTime	The date and time the message was put.
UserId	The sender's user ID.

Get Message from MQ Queue

This activity gets a message from the MQ queue via the Queue Manager.

Get Messages from MQ Queue - Input Properties

Property (A-Z)	Description
Get queue name	The name of the MQ queue from which to get the message.

Property (A-Z)	Description
Match Condition	An array of match conditions in a `key/value` format. Select a key from the drop down box in the Key's **Value** column and provide a value. `MQMO_NONE` does not require a value.
Message data format	The format of the message being retrieved: **String** or **UTF**.
Message Get Options	A collection of built-in Message get options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Queue Access Options	A drop down of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • `-1`: Unlimited wait time • `0`: No wait • A positive number: The time to wait in milliseconds.

Get Messages from MQ Queue - Checkpoints

Property (A-Z)	Description
CorrelationId	The correlation identifier of the retrieved message.
GroupId	An identifier of the message group to which the physical message belongs.
MessageBody	A string representing the UTF content of the retrieved message.
MessageId	The message identifier of the retrieved message.
MsgSeqNumber	The sequence number of a logical message within a group.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
PutTime	The date and time the PUT action was executed for the message.
UserId	The User ID of the user who submitted the message.

Put and Get Message from MQ Queue

This activity puts a message on the MQ queue and retrieves a message from the MQ queue.

Put and Get Messages from MQ Queue - Input Properties

Property (A-Z)	Description
Auto Match Correlation ID	Correlates the sent and received messages by automatically adding a match condition. This condition indicates that the **Correlation ID** of the received message should equal the **Message ID** of the sent message.
Get message data format	The format of the received message: **String** or **UTF**.
Get Queue Access Options	A collection of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
Get queue name	The name of the MQ queue from which to get the message.
Match Condition	An array of Get Message match conditions n a `key/value` format. Select a key from the drop down box in the Key's **Value** column and provide a value. `MQMO_NONE` does not require a value.
Message body	A string representing the UTF content of the submitted message.
Message Get Options	A collection of built-in options that control the Get operation. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
Message properties	An array of Put message properties in a `key/value` format.
Message Put Options	A collection of built-in message put options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.

Property (A-Z)	Description
Put correlation ID	A correlation identifier for the message to be submitted.
Put message format	The format of the message: being sent: **String** or **UTF**.
Put priority	The priority of the message to be submitted.
Put Queue Access Options	A collection of built-in MQ queue access options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
Put queue name	The name of the MQ queue upon which to put the message.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • `-1`: Unlimited wait time • `0`: No wait • A positive number: The time to wait in milliseconds.

Put and Get Messages from MQ Queue - Checkpoints

Property (A-Z)	Description
CorrelkationId	The correlation identifier of the retrieved message.
GroupId	An identifier of the message group to which the physical message belongs.
MessageBody	A string representing the UTF content of the retrieved message.
MessageId	The message identifier of the retrieved message.
MsgSeqNumber	The sequence number of a logical message within a group.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
PutTime	The date and time the PUT action was executed for the message.
UserId	The Userid of the user who submitted the message.

Subscribe to MQ Topic

This activity creates a subscription for an MQ topic. From this point on, a subscriber can receive

messages that were published to a specific topic using the **Receive Message from MQ Topic** activity.

Subscribe to MQ Topic - Input Properties

Property (A-Z)	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Subscription name	An automatically generated subscription name.
Topic Access Options	A collection of built-in MQ topic access options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
Topic name	The name of the MQ topic from which subscribers can access the message.

Subscribe to MQ Topic - Checkpoints

Property	Description
Result	The Subscription name entered manually or generated automatically. **Notes:** • Although this does not appear in the **Checkpoints** grid, it is visible as an output property in the "Select Link Source Dialog Box". • This property can be used as an input value for the subscription name in the **Unsubscribe from MQ Topic** and **Receive Message from MQ Topic** steps.

Unsubscribe from MQ Topic

This activity cancels the subscription to an MQ topic.

Unsubscribe from MQ Topic - Input Properties

Property	Description
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Subscription name	The subscription from which to unsubscribe. This can be a literal string or an expression linked to the output of a **Subscribe to MQ Topic** step.

Unsubscribe from MQ Topic - Checkpoints

Property	Description
Result	A boolean variable indicating whether the step succeeded.

Publish Message to MQ Topic

This activity publishes a message to subscribers using an MQ topic. The **Receive Message from MQ Topic** activity retrieves the message.

Publish Message to MQ Topic - Input Properties

Property (A-Z)	Description
Correlation ID	The UserId of the user who submitted the message.
Message body	The message to publish.
Message data format	The format of the message being published: **String** or **UTF**.
Message Properties	An array of message properties in a `Key/Value` format.
Message Put Options	A drop down of built-in options that control the Put operation, such as `MQPMO_SYNCPOINT`. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
Topic Access Options	A drop down of the built-in options that control the opening of the topic. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
Topic name	The name of the topic on which to publish the message.

Publish Message to MQ Topic - Checkpoints

Property (A-Z)	Description
MessageId	A message identifier of the submitted message.
PutTime	The date and time the message was put.
UserId	The UserId of the user who submitted the message.

Receive Message from MQ Topic

This activity receives messages for subscribers, that were published to a specific MQ topic.

Receive Message from MQ Topic - Input Properties

Property (A-Z)	Description
Match Condition	An array of match conditions n a `key/value` format. Select a key from the drop down box in the Key's **Value** column and provide a value. `MQMO_NONE` does not require a value.
Message data format	The format of the message being retrieved: **String** or **UTF**.
Message Get Options	A drop down of built-in message get options. **Note:** To access additional options, use an event handler. For details, see "Coding Service Test Events" on page 641.
MQManager	A connection expression linked to the most recent **Connect to MQ Queue Manager** step.
Subscription name	The subscription through which to receive topic messages. This can be a literal string or an expression linked to the output of a **Subscribe to MQ Topic** step.
Wait interval	The maximum time in milliseconds to wait for the message to arrive, using the following values: • `-1`: Unlimited wait time • `0`: No wait • A positive number: The time to wait in milliseconds.

Receive Message from MQ Topic - Checkpoints

Property (A-Z)	Description
CorrellationId	The correlation identifier of the retrieved message.
GroupId	An identifier of the message group to which the physical message belongs.
MessageBody	A string representing the UTF content of the retrieved message.
MessageId	The message identifier of the retrieved message.
MsgSeqNumber	The sequence number of a logical message within a group.
Priority	The priority of the message rated from 0 through 9 (highest). A message with a higher priority will be taken from the queue before messages with a lower priority.
PutTime	The date and time the message was put.
UserId	The User ID of the user who submitted the message.

HP Automated Testing Tools Activities

This activity group contains activities that allow you implement unified testing plan. For task details, see "How to Call Tests from Other Applications" on page 411.

- **Call API Action or Test.** Invokes a Service Test Test action or test when the current Test Flow or loop reaches this step.
- **Call Virtual User Generator script.** Runs a LoadRunner Virtual User Generator (VuGen) script.

HP Automated Testing Tools - General Properties

The following section describes the general properties for the HP Automated Testing Tools activities.

Property (A-Z)	Tool	Description
Action name	**API (Service Test) and GUI**	The name of the action being called. This field is read-only. To select an action, click the **Select Action or Test** button in the Input/Checkpoint view.
Description	**All**	An editable field describing the call.
Result Directory	**API (Service Test)**	The location in which to store the run results. This field is editable.
Script path	**Virtual User Generator**	The path of the Virtual User Generator-LoadRunner script. This field is read-only. To select a script, click the **Select Virtual User Generator Script** button in the Input/Checkpoint view.
Test path	**API (Service Test)**	The path of the test. This field is read-only. To select a test, click the **Select Action or Test** button in the Input/Checkpoints tab.

SAP Activities

This activity group contains the activity related to SAP IDoc status.

- **Get IDOC Status.** Retrieves the status of an IDoc.

This section describes the built-in SAP activity. For information about imported SAP activities representing RFCs and IDocs, see "How to Create an SAP Test Step" on page 485.

Get IDoc Status - General Properties

To view these properties, click the **General** tab in the Properties pane.

Property	Description
Timeout	The maximum time allowed for retrieving the IDoc status in milliseconds.
Connection Info	The connection information for the SAP server. If you created a step based on an imported IDoc, you can link to these properties using the **Link to data source** button. For details, see "How to Create an SAP Test Step" on page 485.

Get IDoc Status - Input Properties

To view the input properties, click the Input/Checkpoints tab in the Properties pane.

Property	Description
IDoc Number	The IDoc number for the IDoc whose status you want to retrieve.

Get IDoc Status - Checkpoints

To view the checkpoint properties, select the Input/Checkpoints tab button in the Properties pane and click in the lower **Checkpoints** area.

The checkpoint name corresponds to the SAP property ID.

Property	Description
MANDT	The client.
DOCNUM	The IDoc number.
LOGDAT	The date of the status information.
LOGTIM	The time of the status information.
COUNTR	The IDoc status counter.
CREDAT	The creation date of the status record.
CRETIM	The creation time of the status record.
STATUS	The status of the IDoc.
UNAME	The user name.
REPID	The program name.
ROUTID	The name of the subroutine or function module.
STACOD	The status code.
STATXT	The status code textual representation.
SEGNUM	The SAP segment number.
SEGFLD	The field name in the SAP segment.

Property	Description
STAPA1,2,3,4	Status parameters 1, 2, 3, and 4.
STATYP	The system message type: A, W, E. S, or I.
STAMQU	The status message qualifier.
STAMID	The status message ID.
STAMNO	The status message number.
TID	The transaction ID.
APPL_LOG	The application log.

XML Activities

This activity group contains activities related to XML files:

- **Transform XML.** Converts an XML file to another structure based on the specified XSLT.
- **Compare XMLs.** Performs a comparison between two XML strings.
- **XML to String.** Converts an XML file to a text string.
- **String to XML.** Converts a string to XML structure.
- **Validate XML.** Validates an XML file against an XSD schema.

The **XML to String** and **String to XML** activities are useful when you need to access the output in a specific format—either as string or XML.

For example, suppose your test contains a Web Service call whose output is XML. However, you need to use the output as an input for the next step, such as aService Test test step, which requires a textual string—not XML.

You can use the **XML to String** activity to create a string that is compatible with the Service Test test step.

After the Service Test test step, you can use a **String to XML** activity to convert the output to XML, making it available for linking, and checkpoints.

For task information, see:

- "How to Validate an XML file" on page 412
- "How to Transform an XML File" on page 413
- "How to Compare XML Strings" on page 415

XML - Input Properties

To view the Input properties, click the Input/Checkpoints tab in the Properties pane.

Property	Activity	Description
<XML>	**XML to string**	The XML to convert to a string. This can be displayed in Text or Grid views.
XML string 1 **XML string 2**	**Compare XMLs**	The XML strings upon which to perform a Diff operation. **XML string 2** denotes the newer string.
Ignore element order	**Compare XMLs**	Ignores differences between the order of the elements in the XML strings.
Ignore namespaces	**Compare XMLs**	Ignores differences between namespaces in the XML strings.
Ignore XPaths	**Compare XMLs**	Ignores specific XPath nodes, listed as array elements of this property.
Import XSD/XSLT to folder	**Validate XML** and **Transform XML**	Copies the XSD/XSLT to the test's folder, contributing to the test's portability.
Source string	**String to XML**	The string to convert to XML.
XML string	**Validate XML** and **Transform XML**	The XML string to validate or transform.
XSD/XSLT file	**Validate XML** and **Transform XML**	The path of the source XSD/XSLT file.

XML - Checkpoints

To view the checkpoint properties, select the Input/Checkpoints tab in the Properties pane and click in the lower **Checkpoints** pane.

Property (A-Z)	Activity	Description
<XML>	**String to XML**	The XML representation of the response. For details, see "Properties Pane Tabs" on page 186.
Are Equal	**Compare XMLs**	No differences were found between the compared XML strings, based on the Input properties ignore settings.
Output string	**XML to String**	The expected string representation of the XML document. **Tip:** To view the actual string: • See the **Captured Data** pane in the Run Results Viewer. • Select **Run Step** from the step's right-click menu and view the Run Step Results pane.

Property (A-Z)	Activity	Description
Result array	**Validate XML**	An array with the following properties: • **Valid.** A boolean variable indicating whether the source XML was valid. • **Errors.** The contents of an error string, if one was issued.
Transformed XML	**Transform XML**	The expected XML in its transformed layout.

The **String to XML** step contains an additional output property, **Result**, visible from the **Result** view in the Properties pane or Select Link Source dialog box. For details, see "Result Tab (Properties Pane)" on page 207.

Property	Description
Result	A boolean variable indicating whether the string had the correct structure to be converted into XML.

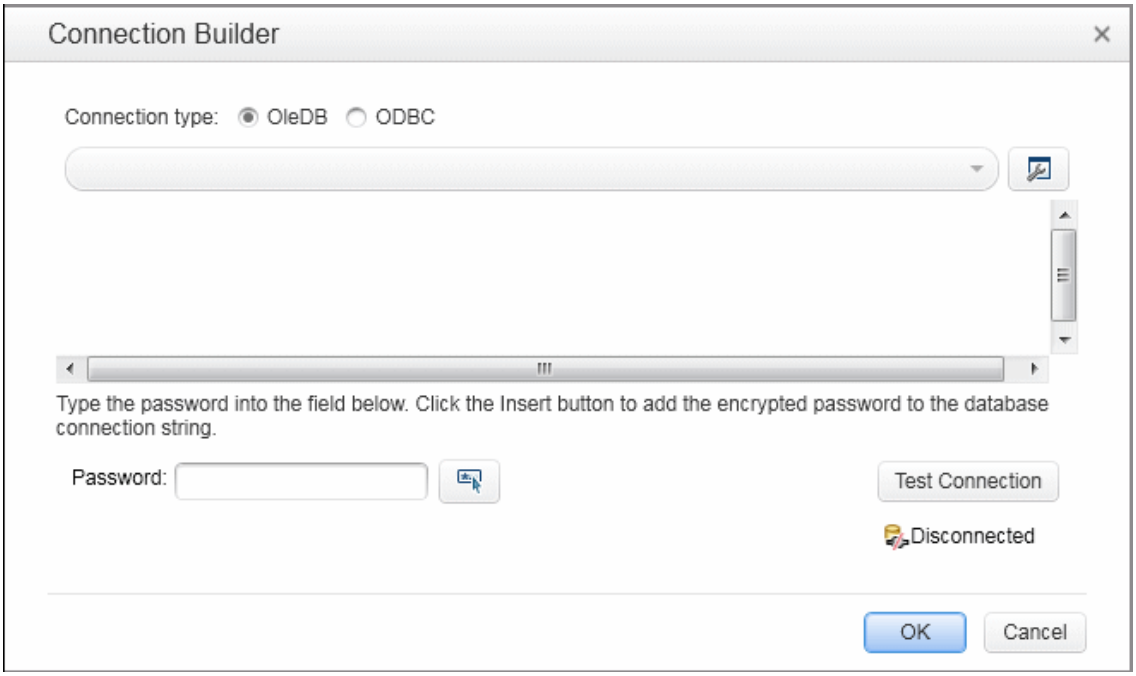
Activity-Specific Dialog Boxes

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

This section includes:

Connection Builder Dialog Box

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 onto the canvas. 2. Click the **Input/Checkpoints** tab in the Properties pane. 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 399

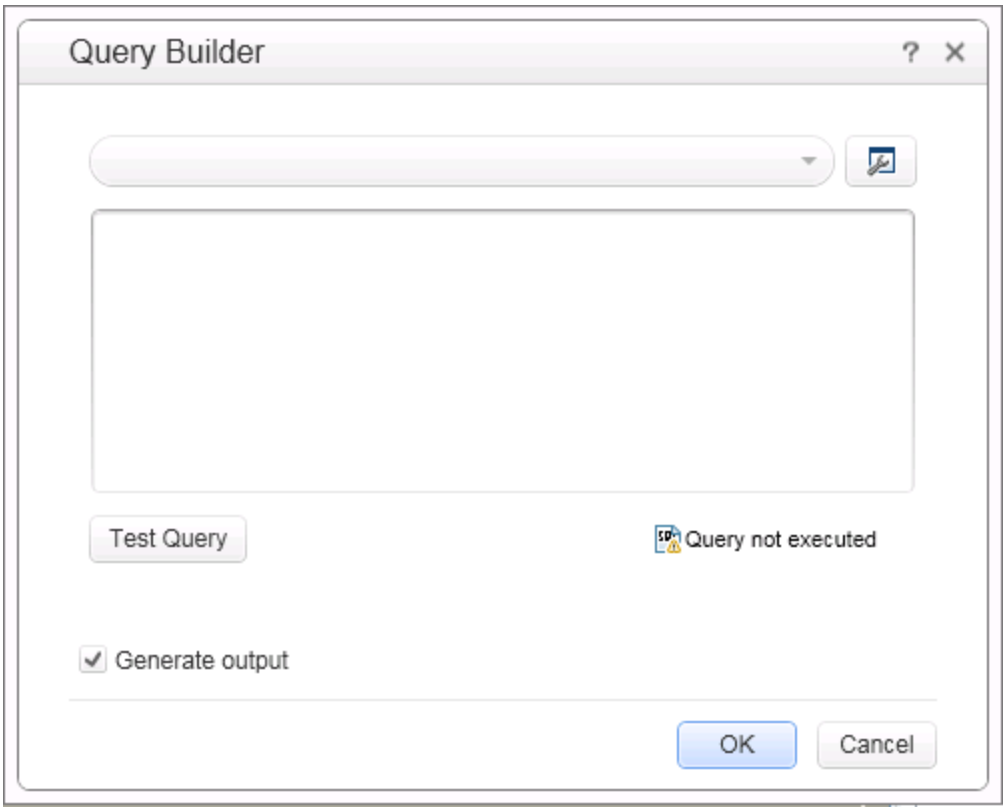
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Connection type	The type of database connection: **OleDB** or **ODBC**.
	Edit Connection String. • **For OleDB.** Opens the Microsoft Data Link Properties dialog box to help you create a new string or edit an existing one. • **For ODBC.** Opens the Select Data Source Name dialog box. For details, see "Select Data Source Name Dialog Box" on page 105.
<connection string area>	An editable area for pasting in existing strings, or for editing the connection string selected in the drop down list.
Connection string	A drop down list of previously defined connection strings.

UI Elements	Description
Password	The password required to connect to the database. The password is displayed in encrypted form.
	**Insert.** Inserts a parameter representing the password at the location of the cursor in the Connection String area.
Test Connection	Checks the database connection using the string shown in the text area. If it succeeds to connect to the database, the status message beneath this button changes from **Disconnected** to **Connected**. 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.
Connected/Disconnected	The current connection status.

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 onto the canvas. 2. Click the Input/Checkpoints tab in the Properties pane. 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 399

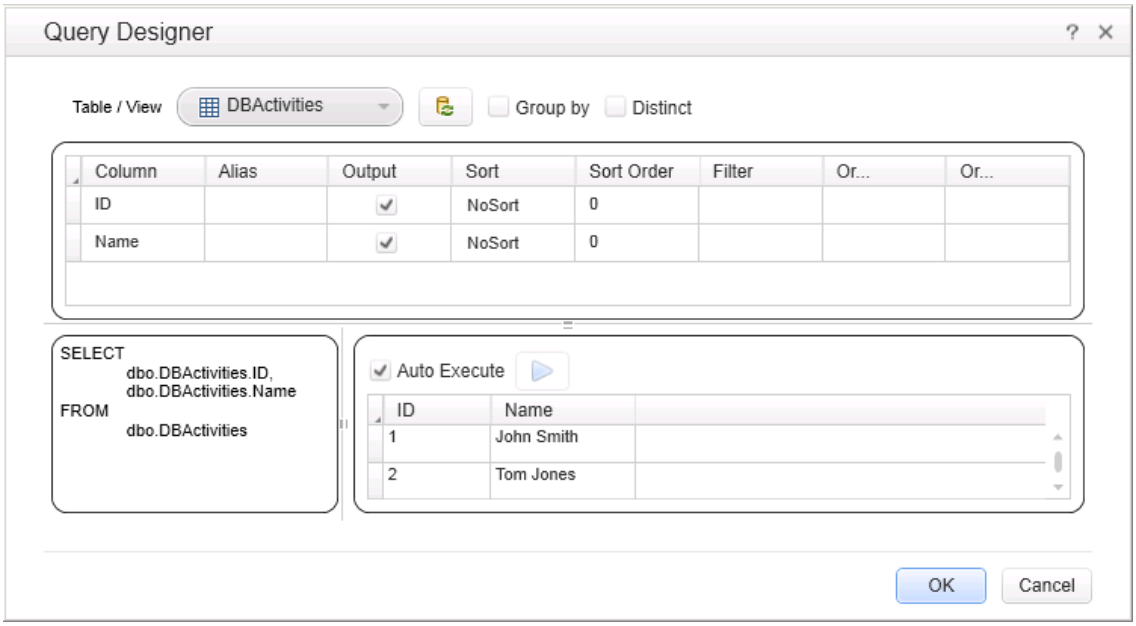
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
	**Query Designer.** Opens the Query Designer dialog box.
Query string	A drop-down list of previously defined query strings.
<query string area>	An editable area for pasting in existing strings, or for editing the query string selected in the drop-down list.
Test Query	Tests the query against the database to which you are connected.
<query status>	The displayed query's status. The possible values are: • **Query not executed** • **Query executed successfully** • **Query completed with errors**
Generate output	Automatically retrieves table information from the database when you click **OK,** and sets the structure of the output in the Properties pane'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.
- creating a new database data source in the Data Pane.



To access	In the Input/Checkpoints tab, for a **Select Data** step: 1. Select the Input property—**Query string**. 2. Click the **Build an SQL query string** button in the right side of the **Value** column to open the "Query Builder Dialog Box". 3. Click the **Query Designer** button . In the Data Pane, when defining a new Database type data source using the "Add New Database Data Source Wizard" on page 102: 1. In the Add New Database Data Source wizard, proceed to the second screen, **Set SQL 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 399 • "Add a database data source" on page 91

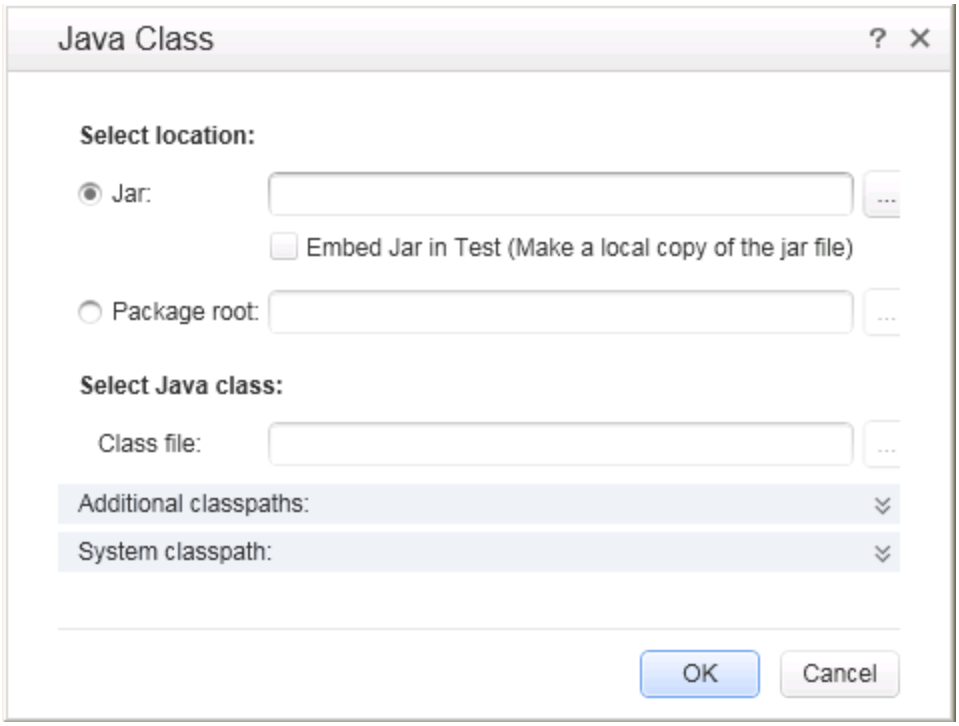
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Table/View	A drop-down list of the tables and views in the database. The tables and views are distinguishable by different icons. **Note:** The number of items shown in the list is limited to 1000. You can also enter text in the field to display a list of relevant data sources.
	Reset Query. Resets the query to its original state and discards all changes.

UI Elements	Description
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.
<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.** Enables 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.** Enables you to filter the results of the query, by adding a `WHERE <column_name> = <filter text>` to the SQL statement.
<SQL statement preview pane> (bottom left)	A preview of the SQL statement generated by your selections.
<returned rows pane> (bottom right)	The rows returned by the query. **Tip:** If you enable **Auto Execute**, these rows display the current query results.
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 the **Execute query** button  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.

Java Class 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 Properties pane, click the Input/Checkpoints tab . 3. Click the **Java Class** button.
Relevant tasks	" How to Create a Call to a Java Class" on page 402

User interface elements are described below:

UI Elements	Description
Jar	The Jar (Java Archive) file to use in the Java call.
Embed Jar in Test	Copies the Jar file to the test's **EmbeddedJavaResources** folder. This allows the test to be ported to other machines. (Due to a limitation, you must select this check box before you browse for the class file.)
Package root	The root folder containing the classes to be called by the step.

UI Elements	Description
Class file	The Java class to associate with the **Call Java Class** step. When you click **Browse**, it shows the classes in the specified `.jar` file.
Additional classpaths	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. • **Add.** Adds the contents of the **Additional Classpath** box to the additional classpath list. • **Remove.** Removes the selected entry from the list.
System classpath	The paths you defined in the Test Settings for system classpaths. To modify this setting, click on the **Start** or **End** step and open the Properties pane's **General** view. Modify the Test Setting's **JVM > Classpath** node.

Troubleshooting and Limitations - Activities

This section describes troubleshooting and limitations for working with activities.

System Activities

- **End Program.** You cannot specify **Window Title** as an input method for a windowless process, even if you are using a wildcard expression.
- **End Program.** If you are running on a 64-bit machine, the **End Program** activity will not be able to terminate 64-bit applications. However, it can terminate other 32-bit applications.

Java Activities

- **Call Java Class.** Supports only Java primitive types.
- **Call Java Class.** Once you select a Java file for the call, the **Java Class** button is disabled. As a result, you cannot replace or update the Java file.

  Workaround: Remove the step containing the **Call Java Class** step and add a new one using the new Java file.
- **Call Java Class.** Java code loaded by the JVM (Java Virtual Machine) cannot be modified or updated when the JVM is running.

Network Activities

- **HTTP Request/Receiver.** Nested transactions are not support ed.

  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.
- **HTTP Request/SOAP Request**. XML file that use XSL, are not supported.
- **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: Open the **Text** view to view the correct XML.

Database Activities

- Nested transactions are not supported in database transaction activities.

  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.
- Databases which are supported by ODBC, but not by OLEDB, cannot be accessed by Service Test.

FTP Activities

- **FTP:** When working with FTP activities that specify paths, you need to enter the full path.
- **FTP Download:** When downloading from Linux servers, you cannot download an empty folder.

IBM Websphere MQ Activities

- **Get Message from MQ Queue.** The MQGMO_MARK_SKIP_BACKOUT option is not supported.

- **Put Message to MQ Queue, Publish Message to MQ Topic.** The message body should not exceed 64K bytes. If it exceeds this size, the execution issues a MQRC_CONVERTED_ STRING_TOO_BIG status. This is a limitation of IBM MQ.
- **MQ Steps.** Automatic linking of IBM Websphere MQ steps to the most recent **Connect to MQ Queue Manager** connection, is supported only when the steps are on the same level in the container, or if the connection step is in a parent container. If the connection step is in a leaf container and the step using the connection is in a parent container or in another leaf, Service Test does not create an automatic link.

  Workaround: Manually link to a MQManager property using the Select Link Source dialog box.

Chapter 24: Local Activities

This chapter includes:

Concepts

Local Activity Overview

In addition to the **Standard** activities built in to Service Test, you can create or import other services and activities. These activities, displayed under the Toolbox pane's **Local Activities** node, are Web Service, REST Service, Web Application Services, .NET assembly operations, SAP IDocs/RFCs whose contract/document or DLL you import.

The activity groups—**REST Services**, **Web Services**,**Web Application Services**, **.NET Assemblies,** and **SAP**—are displayed only after importing or defining at least one entity.

This section includes:

Web Service Activities

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 details on importing a WSDL, see "How to Import a WSDL-Based Web Service" on page 472.

If your service changed during development, you can use the update mechanism for updating the service. For details, "Updating Services and Assemblies" on page 520.

The import supports both Document/Literal and RPC type Web services.

In **Document/Literal** 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 type Web services, the Properties pane displays 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 Properties pane does not display the input and output properties for RPC type services.

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.

The WS-I validation tool lets you validate imported services by checking their conformance with the WS-I standard. For details, see "Validate the WSDL file - optional" on page 473.

After you import a Web service, the tree hierarchy displays its components in several levels:

- **Level 1.** Service name
- **Level 2.** Port number
- **Level 3.** Operation name

For details about importing a Web service, see "How to Import a WSDL-Based Web Service" on page 472.

REST Service Activities

You can define REST services, their resources, and methods. You define the metadata manually by modeling the service and defining your own resources and methods. The services, resources, and methods are then stored as a prototype activity within your test.

After you define a REST service and its resources and methods, Service Test creates a hierarchy of **Service**, **Resource**, and **Method**, depending upon how you defined your REST service.

Service Test provides a designer window for creating and configuring REST service method activities. You drag the REST methods onto the canvas, just as you would with other Toolbox pane activities.

After creating REST methods, you can reuse them for multiple steps. For details, see "How to Create a REST Method" on page 475.

Service Test also enables you to update REST service definitions through the "Resolve REST Method Steps Wizard" on page 535.

In addition, you can also define properties and parameters for your REST service at all levels of the hierarchy and pass these properties or parameters from the service and resource levels to the resource and method levels. For details, see "Passing REST Service Properties" on page 467.

Web-Application Services

Web Application services provide a description of a HTTP-based Web application in an XML format. This description is contained in a Web Application Description Language (WADL) file. This WADL file describes the resources provided by a service and the methods used to access the service.

Like a Web Service, you import a Web application service into Service Test. The resources and methods are then displayed in a hierarchy like a REST service, with a Service, Resource, and Method hiearchy.

The URL for a Web application is defined in the XML of the WADL file. However, you can define other HTTP properties as well as add and define input and output parameters for an activity's methods.

Like REST services, you can define parameters and parameters values at all levels of the Web Application hierarchy and pass these parameter values to lower levels of the hierarchy.

The imported Web Application service methods serve as a prototype for test steps. You can modify the parameter values of a method after dragging the method to the canvas.

.NET Assembly Activities

The .NET importer lets you create activities for testing APIs in the form of .NET assemblies. You can interface with the types defined in the assembly.

You begin by importing the .NET assembly into your test. The Toolbox pane displays the assembly as an activity. You drag the .NET activity on to the canvas as you would with any other activity.

You can define input and output properties and link them to other steps within your script. For example, you can design a test that retrieves an object through the .NET assembly. This object can be an input property for a Web service step.

When you import a .NET assembly, it saves a local copy of the assembly with the test, making the test portable should you copy it to another machine. If the assembly, however, calls other assemblies, the test may not run until you copy the additional assemblies to the new machine. When running the test, it uses the local copy of the assembly, and any dependent assemblies are referenced at their original locations.

For task details, see "How to Create a .NET Assembly API Test Step" on page 473.

For user interface details, see "Import .NET Assembly Dialog Box" on page 494.

SAP Activities

The SAP importer lets you create SAP activities in the Toolbox pane, by importing SAP Intermediate Documents (IDoc) and Remote Function Calls (RFC).

These activities can be useful for testing the SAP server response in several common scenarios:

- Send an IDoc to an SAP server, and confirm that the IDoc was sent
- Check an IDoc's status on the SAP server
- Call an RFC in SAP and ensure that it returned the expected results

These activities can be also be useful when upgrading a system to verify that the integration patterns are still functional, such as Aggregator, Enricher, Router, Translation, Bridge, Splitter, and so forth.

First you define one or more SAP Connections in the SAP Connections pane in the Options dialog box (**Tools > Options > API Testing** tab **> SAP Connections** node). For details, see "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 275.

Once the connection is established, you can search for an IDoc or RFC and import them as activities into the Toolbox.

Once they are in the Toolbox, you can add them to a test or action and assign property values as with all other steps.

For task details, see "How to Create an SAP Test Step" on page 485.

For user interface details, see "Import from SAP/Update Dialog Box" on page 502.

Activity Sharing

The activity sharing feature enables you to save locally stored services to a repository, so that they will be available for other tests.

You can specify a repository on the file system or in ALM. The next time you create a test, you can access the service's activities from the repository instead of reimporting or recreating them.

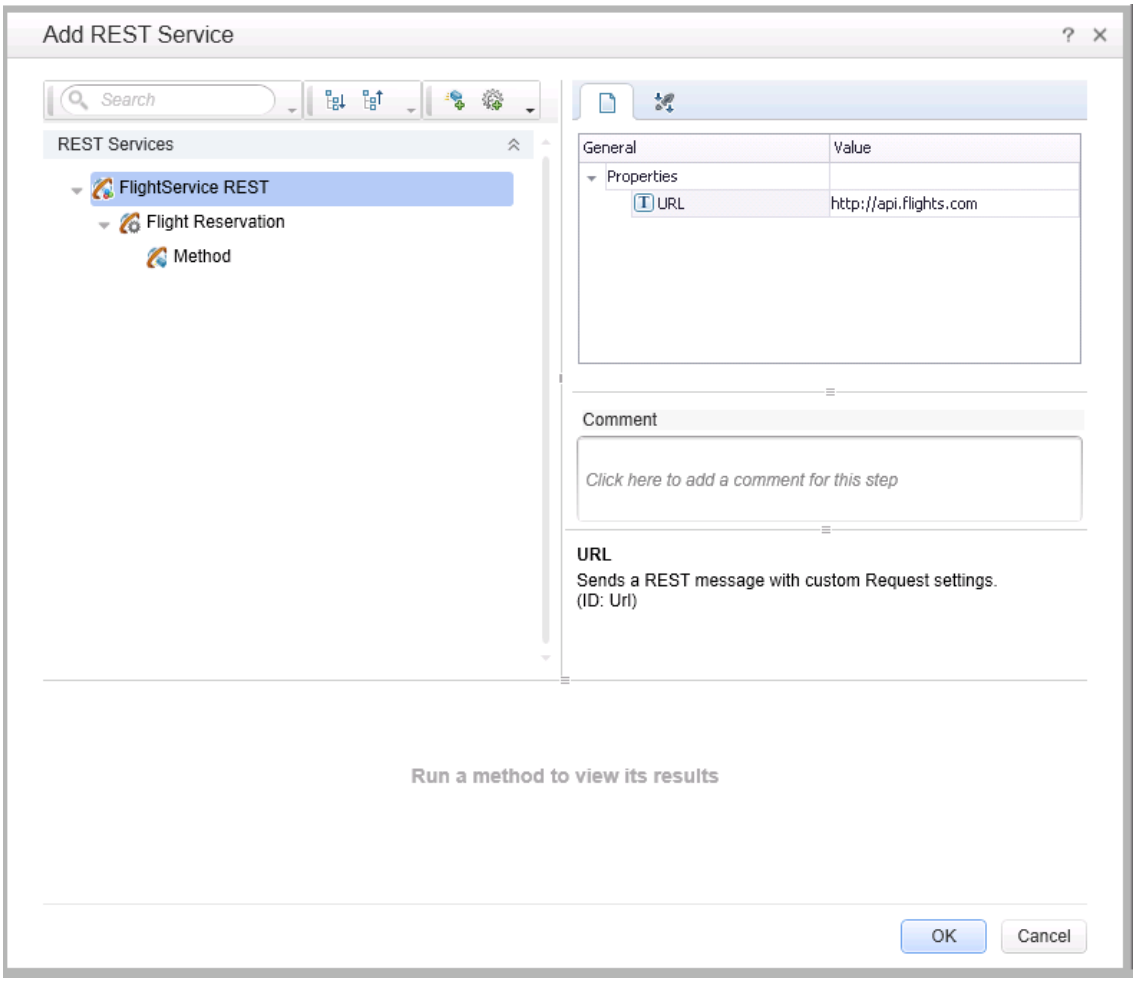
If the resource (WSDL or REST service) becomes unavailable, the canvas displays alerts on the steps that use the resource. If you run the test when its resource is not available from its original source, it uses a copy of the test stored in its cache. By clicking the alert, you can reload the steps when the resource becomes available again.

The Toolbox pane detects version updates for activities stored in a repository. When it detects a discrepancy between the step and the Toolbox pane activity, it displays an alert for the step. By clicking the alert, you can automatically update the resource from its source.

Passing REST Service Properties

When creating or editing a REST service, you may want to define the value of a URL property or a custom input or output property at the service or resource level in order to make this property or parameter available for all resources and methods included in the service or resource. For example, if your REST service resources and methods are all referencing the same URL prefix during run time, you can define the value of URL property at the service level of the REST service and pass the URL property value to all its resources and methods.

You can also define custom input and output property names and values at all levels of the hierarchy and pass these input and output properties and their values through the REST service hierarchy. For details on creating custom input and output properties, see "Define custom properties - optional" on page 477.



After a URL property value is defined at a higher level, you can define other (relative) additions to the URL property value for any of the resources and methods below it. These adjusted relative URL values are then concatenated to the URL property value that was received from higher levels of the hierarchy and the adjusted values are passed to all levels below it. For example, if you add a URL property value at the service level, you can append relative URL paths for a selected resource or method. The URL property value passed from the service level is then concatenated with the relative URL property value added at the resource or method levels.

Note: You cannot modify URL property values passed down from a higher level of the REST service hierarchy. You can only add to them with a relative URL value.

After you add additional relative URL property values at lower levels of the hierarchy, such as at the resource or method level, you must assure that the full URL is a proper URL. For example, if you define the URL property value at the resource level with `http://`, the relative URL property value appended at the method level should not also use a `http://` prefix.

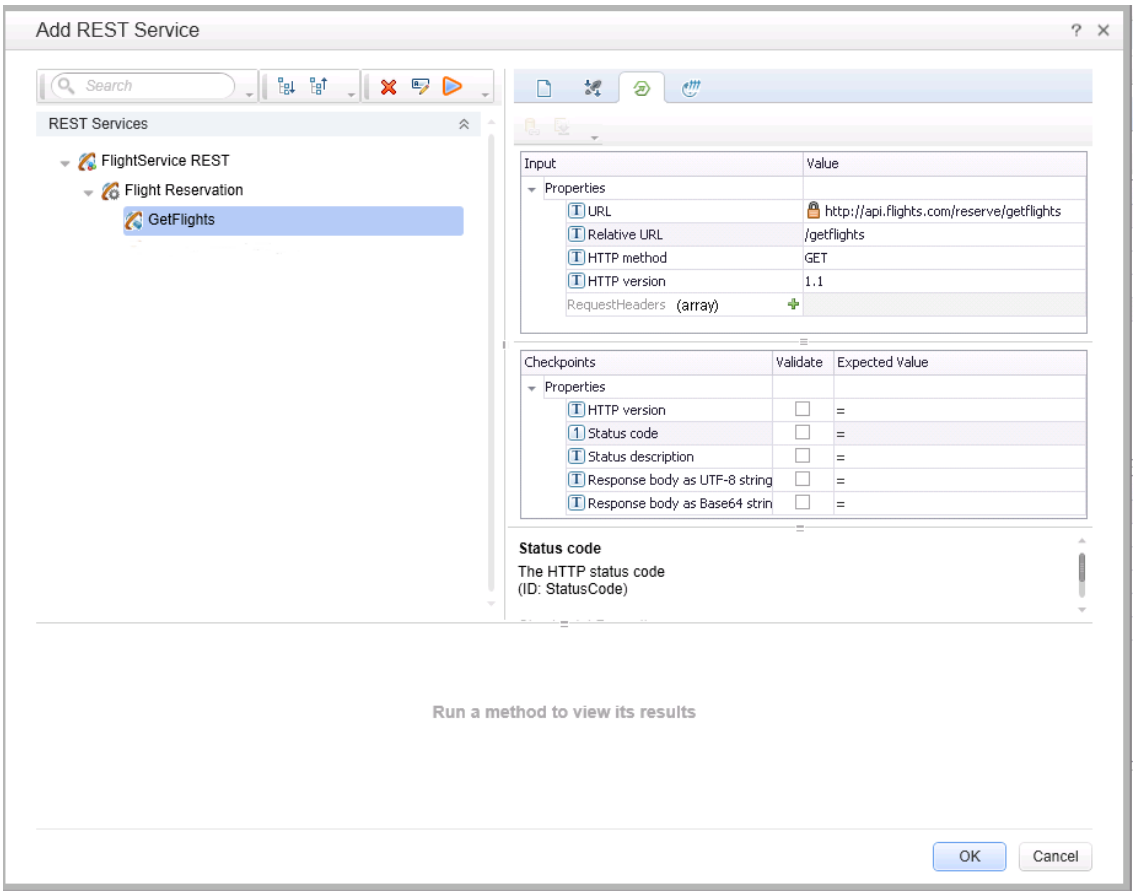
Example

You define a URL property value for the REST service at the service (top) level: `http://flights.api.com`. This value is then passed to any resources and methods within this REST service.

You also create a resource for this service, called **Flight Reservation**. You define the relative URL value as `/reserve`. This relative URL property value is then concatenated with the URL property value passed from the service level to create a URL for the resource: `http://flights.api.com/reserve`. This URL property value can also pass to any methods created for the resource.

You then create a method for the Flight Reservation resource, called **GetFlights** and define the relative URL property value for this method as `/getflights`. This value is then further concatenated with the URL passed from the resource to make a complete URL for the method: `http://flights.api.com/reserve/getflights`.

The example below shows the URL property value passed from the service level (`http://api.flights.com`) and the relative URL property value defined at the resource level (`/reserve`), with the relative URL property value for the method (`/getflights`) also defined. These URL parts are then concatenated in the **URL** property field to make the complete URL for the method.



These properties and custom input or output properties then serve as a prototype template for the REST service, and you can edit the URL property values or custom input and output property values after dragging the method activities in the canvas.

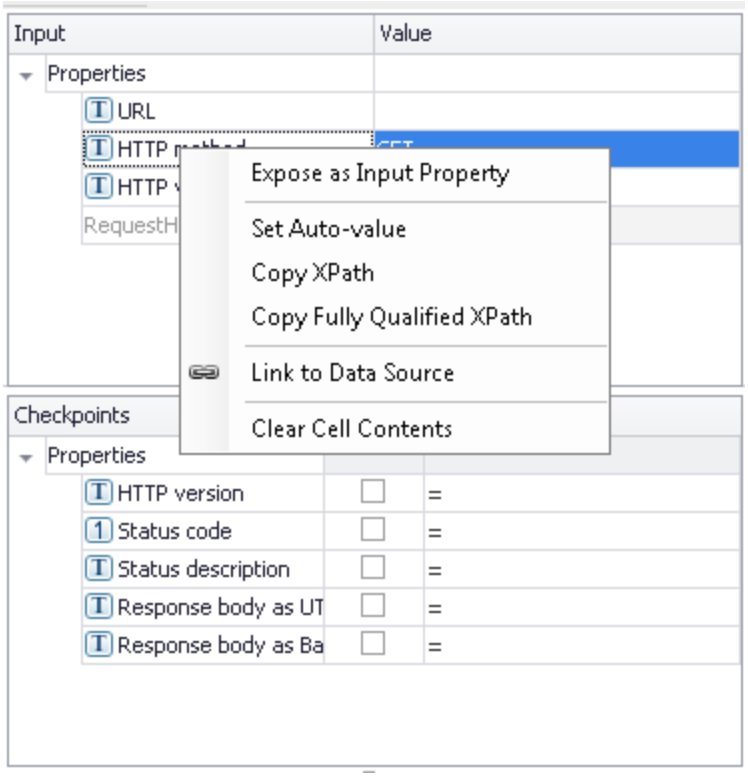
Note: The relative URL is not displayed on the REST method when it is included in a test in the canvas. Only the full, concatenated URL for the method displayed in the Add REST Service dialog box is displayed in the Properties pane for the selected method.

Exposing Properties

When working with REST service methods saved in tests created in Service Test 11.51 or earlier, the Properties pane enables you to expose input and output HTTP properties. Exposing HTTP properties means that you make them available at the REST method wrapper level instead of just the inner HTTP level. You can expose properties that are available from the **General**, **Input/ Checkpoint**, **HTTP**, and **Multipart** views.

Note: You can only expose properties if you are working with tests created in Service Test 11.51 or earlier.

The following example shows the shortcut menu item, **Expose as an input property**. This option prompts you to provide a name for the property as it should appear in the REST wrapper level.



The property, **HTTP method**, will be available in the REST method wrapper. To see the exposed property, select the REST method wrapper in the canvas —not the inner HTTP Request.

Note:

- When exposing a property with incoming links, the links are redirected to the newly created property.
- When exposing a complex property, the new property will be created as a **String** type.

For user interface details, see "New Exposed Property Dialog Box" on page 638.

Tasks

How to Import a WSDL-Based Web Service

This task describes how to import a WSDL file into your test.

This task includes the following steps:

1. **Prerequisites**

   If you intend to import a WSDL stored on ALM, the ALM project to which you are connecting must have the Service Test Management extension enabled. For details, see the "ALM Connection Dialog Box" on page 335.

2. **Open the Import WSDL dialog box**

   - To import a WSDL from the file system or ALM, click Import WSDL **> Import WSDL from File or ALM Application Component**. For details, see the "Select WSDL Dialog Box" on page 488.
   - To import a WSDL from a Web site or UDDI repository, click Import WSDL **> Import WSDL from URL or UDDI**.
     - For a URL, select **Import from: URL**
     - For a UDDI, select **Import from: UDDI**

     For details, see the "Import/Update WSDL from URL or UDDI Dialog Box" on page 489.

3. **Select a source**

   - For **File System** or **ALM Application Components** imports, click the appropriate button in the left pane. Navigate to the WSDL in the file system or the ALM 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. This automatically places the URL in the **Address** field.
   - For a **UDDI** import, 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 490.

4. **Set connection settings for URL/UDDI imports - optional**

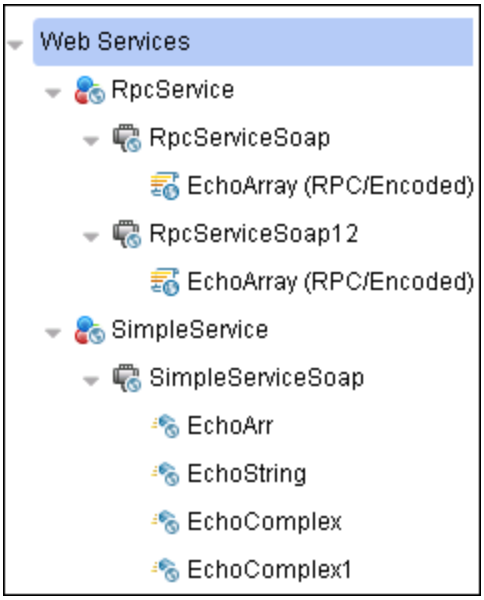
For URL or UDDI imports, if your WSDL must be accessed through a secure server or proxy machine, click **Advanced Settings** to expand the dialog box. Enter the authentication information or the proxy server details.

5. **Mark the WSDL as a server response - optional**

   You can mark the imported WSDL as a server response method. This is useful when working with asynchronous Web services. Click **Advanced Settings** to expand the dialog box and select the **Import as server** option. For details about working with asynchronous services, see "Asynchronous Services" on page 602.

6. **Complete the import**

   Click **OK**. If the import succeeds, the **Toolbox** displays the service, its ports, and operations under the **Web Services** node. The Toolbox pane differentiates between `Document/Literal` and `RPC` encoded services with different icons.



7. **Validate the WSDL file - optional**

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

How to Create a .NET Assembly API Test Step

This task describes how to create a test step for verifying the functionality of a .NET assembly.

This task includes the following steps:

- "Prerequisite " on next page
- "Open the Import .NET Assembly dialog box" on next page
- "Select a GAC assembly - optional" on next page

- "Import a .NET assembly" below
- "Add a .NET assembly step" below
- "Add properties - optional" below
- "Customize the code - optional" below

1. ## Prerequisite

   Prepare a `.dll` assembly file and store it in a location accessible from your machine.

2. ## Open the Import .NET Assembly dialog box

   Click the **Import .NET Assembly** button  on the toolbar. For user interface details, see "Import .NET Assembly Dialog Box" on page 494.

3. ## Select a GAC assembly - optional

   Each computer where the common language runtime is installed has a machine-wide code cache called the GAC (Global Assembly Cache). The GAC stores assemblies specifically designated to be shared by several applications on the computer.

   In the **GAC** tab, select one or more GAC assemblies and click **Select** to add them to the **Selected References** list.

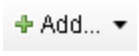
4. ## Import a .NET assembly

   Click the **.NET Assembly Browser** tab. Locate the assembly `.dll` or `.exe` file. Select the file and click **Select** to add it to the **Selected References** list. Click **OK** to add the .NET assemblies to the Toolbox pane.

5. ## Add a .NET assembly step

   Expand the **.NET Assemblies** category in the Toolbox pane and drag a .NET activity onto the canvas.

6. ## Add properties - optional

   a. In the Properties pane's Input/Output tab, click the **Add Input/Output Property** button  to define new input and output properties.

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

      - To add a simple type property, select a type from the drop down.
      - To add an advanced type, click **Advanced** and select a type. All .NET types and `.dll` files referenced by the test, and all types from the imported assemblies, are available. To filter the types, enter a keyword into the **Type** field.

   c. Click **OK** to add the property. Repeat this for all of the required input and output properties.

7. ## Customize the code - optional

   In the Properties pane, click the **Events** button  . Double-click in the ExecuteEvent's **Handler** column. The `TestUserCode.cs` tab opens.

   Edit the `Todo` area. You can access input and output properties that you defined earlier, using autocompletion.

The comments indicate the syntax to use for accessing the properties. For example:

```
/// Use this.CodeActivity4 to access the CodeActivity4
Activity's context, including input and output properties.}
public void CodeActivity4_OnExecuteEvent(object sender,
STActivityBaseEventArgs args)
{
this.CodeActivity4.Input.Property1...
...
```

How to Create a REST Method

This task describes how to set up a REST method service. You define the request body and properties in order to create a prototype method that could be reused by multiple test steps.

This task includes the following steps:

1. ## Prerequisite

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

2. **Open the REST service designer**

   Click the **Add REST Service** toolbar button . The Add REST Service dialog box opens. For details, see the "Add/Edit REST Service Dialog Box" on page 495.

3. **Name the REST service**

   Click on the **New Service** node and provide a meaningful name for the service in the left pane.

4. **Add a resource**

   Click the **Add Resource** toolbar button to add a resource to the REST service. Click on the **REST Resource** node and provide a meaningful name for the resource.

5. **Add a method**

   Click the **Add Method** toolbar button to add a method. Click on the **Method** node and provide a meaningful name for the method.

6. **Set the General properties**

   a. In the right pane's **Properties** list, open the **General** tab.

   b. Enter values for the properties. For details, see the "General Tab (Properties Pane)" on page 195.

7. **Set the URL property values**

   a. In the REST Service editor, select a REST service, resource, or method.

   b. In the General tab, enter the URL property value:

   - If you are entering a URL property value for a REST service, enter the URL prefix in the **URL** property row, beginning with a `http://`
   - If you are entering a URL property value for a REST resource or method, enter the URL property value in the Relative URL property row.

   Note: If you enter a URL property value for the service or resource of your REST service, the URL property values are passed to all resources or methods included in the service or resource. For details, see "Passing REST Service Properties" on page 467

8. **Define the method's HTTP properties**

   a. Select a REST service method.

   b. In the REST service editor's right pane,click the **HTTP Input/Checkpoints** tab.

   c. For each method, modify the **HTTP method** and **HTTP version**.

   d. Use the plus sign in the **RequestHeaders** parent node to add name and value pairs for the request header array.

   For details, see the "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.

9. **Define custom properties - optional**

   For methods that require input, such as `PUT` or `POST`, you can add custom input properties. For methods that provide an output such as `GET`, you add the required output properties. To create these properties:

   a. In the right pane's **Properties** list, click the **Custom Input/Checkpoints** tab .

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

   c. Provide a name, data type, and description (optional) for each property.

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

10. **Enter the request body directly - optional**

    This step describes how to enter the request body directly. (For details on linking to a data source, 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 right pane of the Add REST Service window, open the **HTTP** tab. Make sure the **Body** type in the drop down is set to **Text**, and click within the **Body** section. Press **CTRL+V** to paste the contents of the request into the pane. Make sure that there are no extra spaces before or after the body text.

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

11. **Link the body request to a data source - optional**

    This step describes how to enter the Request body through a data source. (For details on entering the Request body directly, see the previous step.)

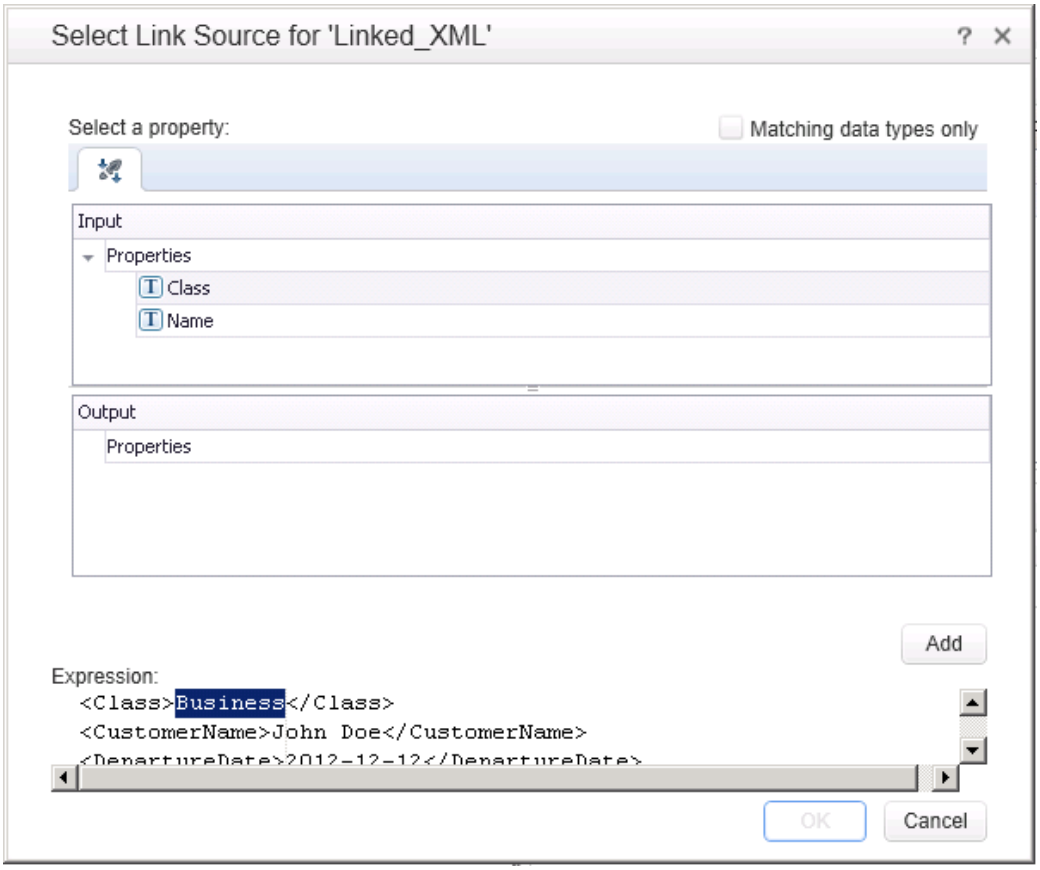
    a. Return to the design document for the REST service and copy the request body to the clipboard.

    b. In the right pane of the Add REST Service dialog box, open the **HTTP** tab . Make sure the **Body** type in the drop down is set to **Text**, and click within the **Body** section.

    c. Click the **Link Body** button to the right of the pane, to open the Select Link Source dialog box. For details, see "Select Link Source Dialog Box" on page 630.

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

    e. In the **Expression** area, highlight the value of the element you want to link. In the upper pane, double-click on the custom property you defined earlier in the properties list. The property is now linked to a value.



f. The modified expression appears in the **Expression** area. In the following example the custom `Class` property is linked to `Class` element in the REST document.

```
<Class>{Step.InputProperties.RestMethod.Class}</Class>
```

g. Click **OK**. Service Test places the data in the **Body** area.

12. ## Test the method

Click the Run Method button on the toolbar to test the method with its property values. Note the results in the lower section of the window. If something needs to be modified, do so at this point.

13. ## Add the method to the toolbox

Click **OK** in the Design REST Service dialog box. Service Test adds the REST service along with its resources and methods to the Toolbox pane, under the **Local Activities** category.

14. ## Use the REST activity

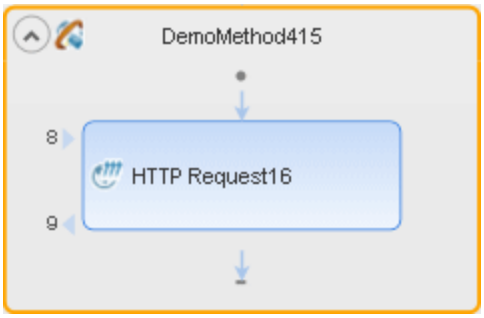
You have now created a prototype for a REST method, complete with input parameters and HTTP information. You can now drag the activity, whose primary properties are already defined, onto the canvas. A wizard helps you resolve any conflicts between the original prototype and a test step. For details, see "How to Resolve Conflicts in a REST Steps" on page 525.

15. **Expose input and output properties - optional (for API tests created in previous versions)**

    You can forward built-in HTTP properties to the REST method wrapper. This is useful for making specific HTTP properties available at the wrapper level.

    Note: If you created a REST method in Service Test 11.52 or Service Test 11.52 or later, all REST properties are incorporated into the REST activity step itself without a wrapper.

    a. Double-click a REST method from the Toolbox pane to add it to the canvas. Expand the method to show the **HTTP Request** frame.

    b. Click in the **HTTP Request** frame. Select a property in the Properties pane and select **Expose as Input Property** or **Expose as Output Property** from the right-click menu.

    c. Provide a name for the property in the New Exposed Property dialog box.

    d. Click the REST method wrapper in the canvas, and view the newly exposed property in the Properties pane's **Custom Input/Checkpoints** tab .

    For details, see "Exposing Properties" on page 470.

16. **Edit the prototype service**

    To edit a REST service method's properties, select it in the Toolbox pane and choose **Edit Service** from the right-click menu. For details, see the "Add/Edit REST Service Dialog Box" on page 495.

17. **Resolve conflicts between the prototype and the test step - optional**

    If you created a test step using a template REST method and then changed the template method, the canvas adds an alert icon adjacent to the affected steps. Click the alert icon to open the Resolve REST Method wizard to resolve any conflicts. For details, see the "Resolve REST Method Steps Wizard" on page 535.

18. **Share the service - optional**

    To add a REST service to the test repository, to make it available for other tests, share it. Select the parent service node in the Toolbox and choose Move to from the right-click menu. For details, see "How to Perform Activity Sharing" on page 486.

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

How to Send a JSON Request to a REST Service

This task describes how to send a JSON request for a REST service.

This task includes the following steps:

- "Set the HTTP properties" below
- "Prepare to load the request body" below
- "Load the request body" below
- "Modify the JSON body - optional" below
- "Data drive or link to fields - optional" on next page

Note: The following tasks show how to send a single JSON request. To create a reusable model, create a prototype. For details, see "How to Create a REST Method" on page 475.

1. ### Set the HTTP properties

   Note: If you are working with test created in Service Test 11.51 or earlier, you must expand the REST activity's wrapper and enter these properties in the HTTP Request step found inside the REST activity wrapper.

   In the Properties pane's Input/Checkpoints tab  , set the destination **URL** and the **HTTP method**, usually `POST` or `PUT`.

2. ### Prepare to load the request body

   Open the Properties pane's **HTTP** view  . Select a **Body** type **JSON**.

3. ### Load the request body

   Click the **Load JSON** button and locate the `.json` file. Click **OK**.

4. ### Modify the JSON body - optional

   If you intend to dynamically assign values to the JSON body from a data source, you need to add escape characters. Open the **Text** view of the JSON body and add the escape character, \, for each occurrence of a square or curly bracket (`{, }, [, and ]`). Do not use an escape character for link expressions enclosed by curly brackets. For example:

   ```
   \{"results":
   \[
   \{"name": "John", "id": 873829904, location: "NY"\},
   \{"name": "Linda", "id": 726371109, location: "LA"\},
   \{"name": "Mike", "id": 711029345, location: "NY"\},
   \]
   \}
   ```

   When no links are used, this is not required.

5. **Data drive or link to fields - optional**

   To data drive or link to specific JSON fields, highlight the value in the body text, click the **Link Body** button , and select a data source. For details, see "Select Link Source Dialog Box" on page 630.

How to Receive a JSON Response from a REST Service

This task describes how to receive a JSON response from a REST service. All HTTP Request activities are capable of receiving JSON responses.

This task includes the following steps:

- "Add a request header - optional" below
- "Add checkpoints - optional" below

Note: The following tasks show how to receive a single JSON request. To create a reusable model, create a prototype. For details, see "How to Create a REST Method" on page 475.

1. **Add a request header - optional**

   Note: If you are working with test created in Service Test 11.51 or earlier, you must expand the REST activity's wrapper and enter these properties in the HTTP Request step found inside the REST activity wrapper.

   If your server requires you to specify JSON during content negotiation, you need to set the request header. In the Properties pane's Input/Checkpoints tab , click the plus sign to add a **RequestHeader** array element. Add a custom request header named `Accept` with the value `application/json`.

2. **Add checkpoints - optional**

   If you require checkpoints to validate the response:

   a. Open the Properties pane's **HTTP** tab . Click in the lower section of the tab, in the checkpoint area.

   b. To validate against regular expressions, set the **Body** type to **Text**. Add regular expressions for each value that you want to validate.

   c. To validate against a JSON response, set the **Body** type set to **JSON**. Click **Load JSON** and locate the `.json` file with the expected values.

How to Import a Web Application Service

This task describes how to import a Web Application service (WADL) into your test.

This task includes the following steps

- "Prerequisite " on next page
- "Import the WADL document" on next page

1. ## Prerequisite

   Before importing, study the structure of your WADL document, as specific elements from the document are imported into the WADL hierarchy inside Service Test.

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <resources base="http://example.com/api">
    <resource path="books">
      <method name="GET"/>
      <resource path="{bookId}">
        <param required="true" style="template" name="bookId"/>
        <method name="GET"/>
        <method name="DELETE"/>
        <resource path="reviews">
          <method name="GET">
            <request>
              <param name="page" required="false" default="1"
style="query"/>
              <param name="size" required="false" default="20"
style="query"/>
            </request>
          </method>
                <resource path="{index}">
                      <method name="GET" id="get index"/>
                      <param name="index" style="template"/>
                </resource>
        </resource>
      </resource>
    </resource>
    <resource path="readers">
      <method name="GET"/>
    </resource>
  </resources>
</application>
```

   For details on WADL elements, see http://www.w3.org/Submission/wadl/.

2. ## Import the WADL document

   In the toolbar, click on the **Add REST Service** button and select **Import WADL** or select **Tools > Add REST Service > Import WADL**. In the Choose WADL file dialog box, navigate to the directory in which the WADL is saved, and select the WADL file.

   Note: You can import only WADLs saved on the file system.

The WADL service is imported to the **Local Activities** node of your test with its resources and methods. The WADL service, resource, and method hierarchy is created based on the XML description provided in the WADL file, described below:

XML Element	Service Test WADL Activity Hierarchy Element
`doc xml:lang="en" title="RestService"`	**WADL service name**
`resource path="<resource name>"`	**WADL resource** **Note:** If multiple resources have the same name, Service Test numbers the resources sequentially to differentiate between resources.
`method name="<method name>"`	**WADL method** ▪ Service Test assigns the WADL hierarchy name using the following criteria: ▪ If a method name has an `"id"` attribute, the name is taken from the value of the `"id"` attribute. ▪ If a method name does not a have an `"id"` attribute, then the name is defined as the value of the `"method name"`. For example, if the `"method name"` is defined as `<"method name="GET"/>`, UFT defines the method name in the WADL hierarchy as `GET Method`. The method name always contains the HTTP method as part of its value. This method (GET, POST, PUT, DELETE, TRACE, OPTIONS, or HEAD is also used as the HTTP method in the HTTP tab of the WADL service. For details on the HTTP tab, see "HTTP Tab (Properties Pane)" on page 199 **Note:** If there are multiple methods of the same name using the default values, then the methods are defined with increasing sequential numbers.

`param name="<parameter name>"`	**WADL resource or method parameters.** These parameters are displayed until the Custom Input/Checkpoints tab in the Edit REST Service dialog and in Input/Checkpoints tab for methods on the canvas. If a `"param name = <name>"` string also contains a `"default=<value>"` string, the value defined in the XML is displayed with the parameter.
`resources base="http://example.com/api"`	**WADL Service URL.** This is displayed on the HTTP tab for the service. The URL is also passed to all resources and methods included in the service. For details, see "Passing REST Service Properties" on page 467. **Note:** You cannot change the URL property for an individual resource or method.

3. ## Edit the service's properties - optional

   To edit the WADL service properties, right-click the service name in the Toolbox and select **Edit Service**.

   In the Edit REST Service dialog box, you can define general properties, parameters, HTTP properties, and HTTP request and response information for the service, resources, and individual methods. For details, see "How to Create a REST Method" on page 475.

   For user interface details, see "Add/Edit REST Service Dialog Box" on page 495.

4. ## Use the WADL service's methods

   After importing a WADL, you have created a prototype for the service, with methods, HTTP properties, and parameters. Drag a method to the canvas to use it in your test.

   Note: You can edit all method properties after dragging the activity to the canvas.

   Caution: If you modify the service's properties after dragging a method to the canvas, the properties of your service will differ from the service's prototype (as defined in the Toolbox). The Resolve REST Steps Wizard is not available to resolve differences for WADL service methods.

5. ## Share the service - optional

   To add a WADL service to the test repository, to make it available for other tests, share it. Select the parent service node in the Toolbox and choose **Move to** from the right-click menu. For details, see "How to Perform Activity Sharing" on page 486.

How to Create an SAP Test Step

This task describes how to create a test step for an SAP IDoc or RFC.

This task includes the following steps:

- "Prerequisite" below
- "Define an SAP Connection" below
- "Open the Import from SAP dialog box" below
- "Select a connection" below
- "Provide credentials - optional" below
- "Search for an IDoc or RFC" below
- "Add the IDoc or RFC to the Toolbox pane" below
- "Create a test/action step" below

1. **Prerequisite**

   You must have the SAP .NET Connector installed on your machine. The installation is available on the SAP Help portal, at http://help.sap.com/saphelp_NW04/helpdata/en/e9/23c80d66d08c4c8c044a3ea11ca90f/content.htm.

2. **Define an SAP Connection**

   Select **Tools > Options > API Testing** tab > **SAP Connections** node. In the SAP Connections pane, define one or more SAP connections. For details, see "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 275.

3. **Open the Import from SAP dialog box**

   Select **Tools > Import from SAP**. For user interface details, see "Import from SAP/Update Dialog Box" on page 502.

4. **Select a connection**

   Select one of the connections that you defined in the SAP Connections pane of the Options dialog box.

5. **Provide credentials - optional**

   Accept the default credentials (as entered in the SAP Connections pane in the Options dialog box) or click **Override connection** to provide different information.

6. **Search for an IDoc or RFC**

   Select **IDoc** or **RFC** and specify a search string using an asterisk (*) as a wildcard. Click **Search**.

7. **Add the IDoc or RFC to the Toolbox pane**

   Check the returned items and click **Import Selected** to add them to the **SAP** category in the Toolbox pane.

8. **Create a test/action step**

Expand the **Local Activities > SAP > <Connection Name> > IDOCs/ RFCs** node in the Toolbox pane, and drag an SAP activity to the canvas. Provide values for the **General** and **Input/Checkpoints** properties.

How to Perform Activity Sharing

This task describes how to save activities to a repository, update them, and view their properties.

This task includes the following steps:

- "Connect to ALM" below
- "Set up the repository paths" below
- "Import a WSDL or create a REST service" below
- "Move the activity to a repository" below
- "View the service properties - optional" below
- "Refresh the activity - optional" on next page
- "Update the activity - optional" on next page
- "Add the activity to a new test - optional" on next page

1. ### Connect to ALM

   If you want to work with a repository on ALM, connect to the desired ALM server. For details, see the "ALM Connection Dialog Box" on page 335.

2. ### Set up the repository paths

   a. Select **Tools > Options > API Testing** tab **> General** node.

   b. In the Activity Repositories section, click the **Browse** button and navigate to the location in the file system or in ALM.

   c. Click **OK**.

3. ### Import a WSDL or create a REST service

   Import a WSDL file or create a REST service method. The Toolbox pane shows the services in the Local Activities node. By default, the test stores these activities in its `EmbeddedJavaResources` subfolder.

4. ### Move the activity to a repository

   Right-click the service node, and select **Move to > File System Activities** or **ALM Activities**. This moves the service from Local Activities to **File System Activities** or **ALM Activities** in the Toolbox pane. The service is also removed from the test's `EmbeddedJavaResources` subfolder and placed in the repository folder. The next time you create a test, these activities will be accessible from the **File System Activities** or **ALM Activities** nodes.

5. ### View the service properties - optional

   Right-click the service's parent node, and select **Properties**. The Properties window shows the service's significant properties. For details see "Web Services Properties Dialog Box" on

page 491.

6. **Refresh the activity - optional**

   To reload the WSDL from its stored location, right-click the service node, and select **Refresh**  . This is useful in cases when the WSDL is stored in a shared location and may have been updated by another user.

7. **Update the activity - optional**

   To reimport the WSDL from its original location, for example to revert back to the original version, right-click the service node and select **Update WSDL**, or click the **Update WSDL** button in the Toolbox pane  .

8. **Add the activity to a new test - optional**

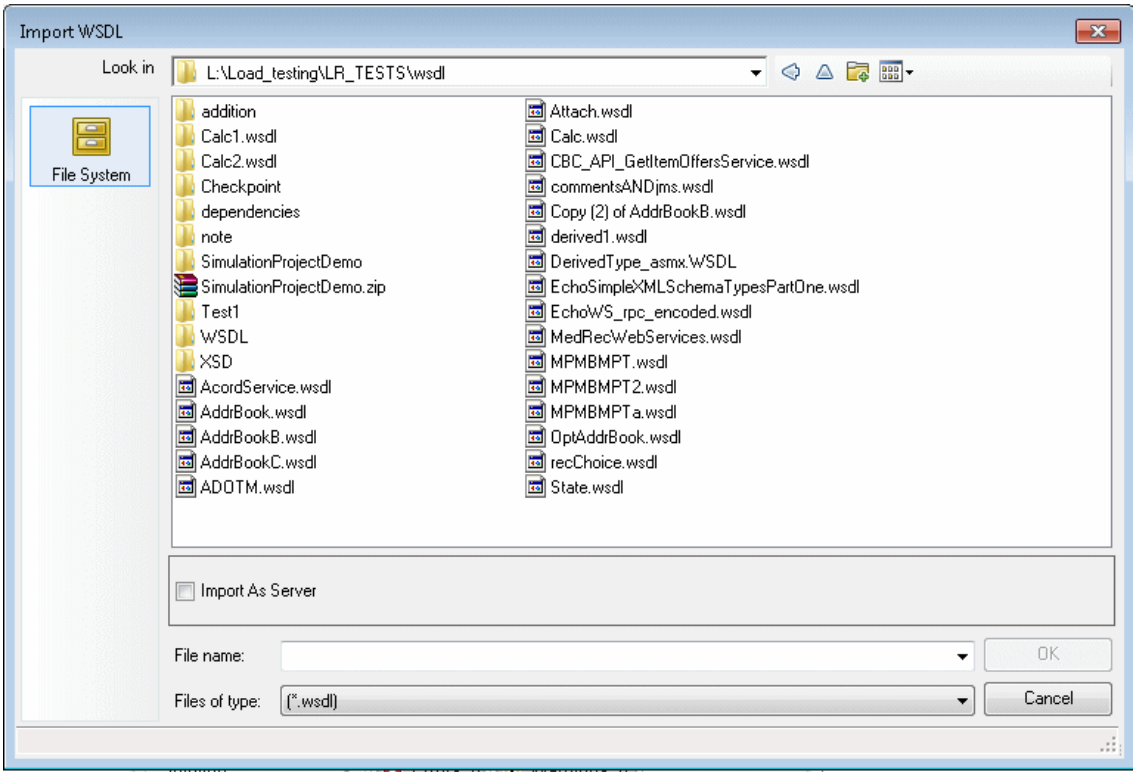
   Open a new test. Expand the **File System Activities** or **ALM Activities** node, select the desired service or method, and drag it onto the canvas.

   You can also remove shared activities from the toolbox and re-add them. For details, see "Toolbox Pane User Interface" on page 256.

References

Select WSDL Dialog Box

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



To access	1. Create or open a test or component. 2. Select **Import WSDL > Import WSDL from File or ALM Application Component**.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 472

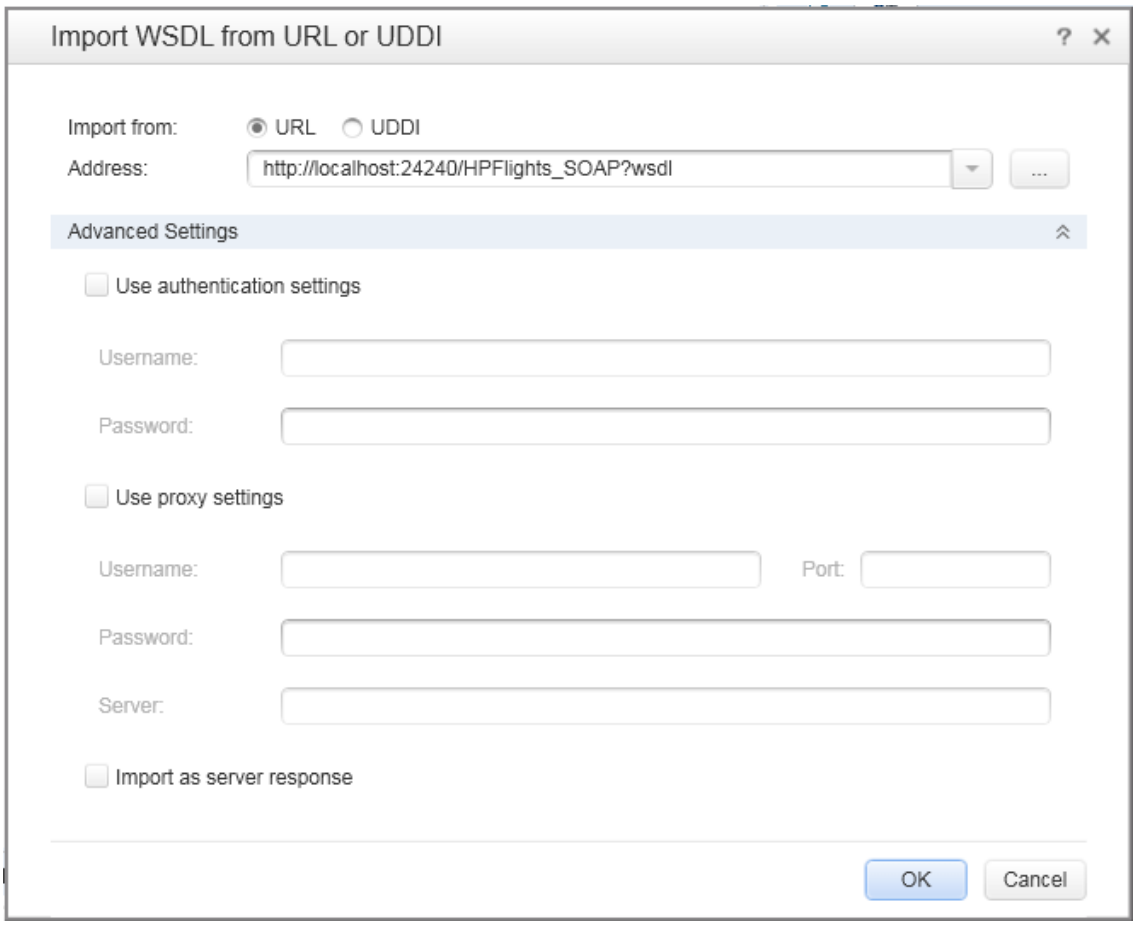
The following elements are included:

UI Elements	Description
File System	Displays WSDLs in the file system.
ALM Application Components	Displays WSDLs in the ALM repository. **Note:** Available only if there is an open connection to ALM with the Service Test Management extension.

UI Elements	Description
Import as Server Response	Imports the WSDL as a server response. Its methods will be server responses—not requests. This is useful for asynchronous type Web services.
File name	The contract name: • **File System.** The file name of the WSDL. • **ALM Application Components.** The name of the application component or service.
Files of type	The document type: **WSDL** extension or **All files**.

Import/Update WSDL from URL or UDDI Dialog Box

This dialog box enables you to import WSDLs using a URL or UDDI or update an existing WSDL from a new location.



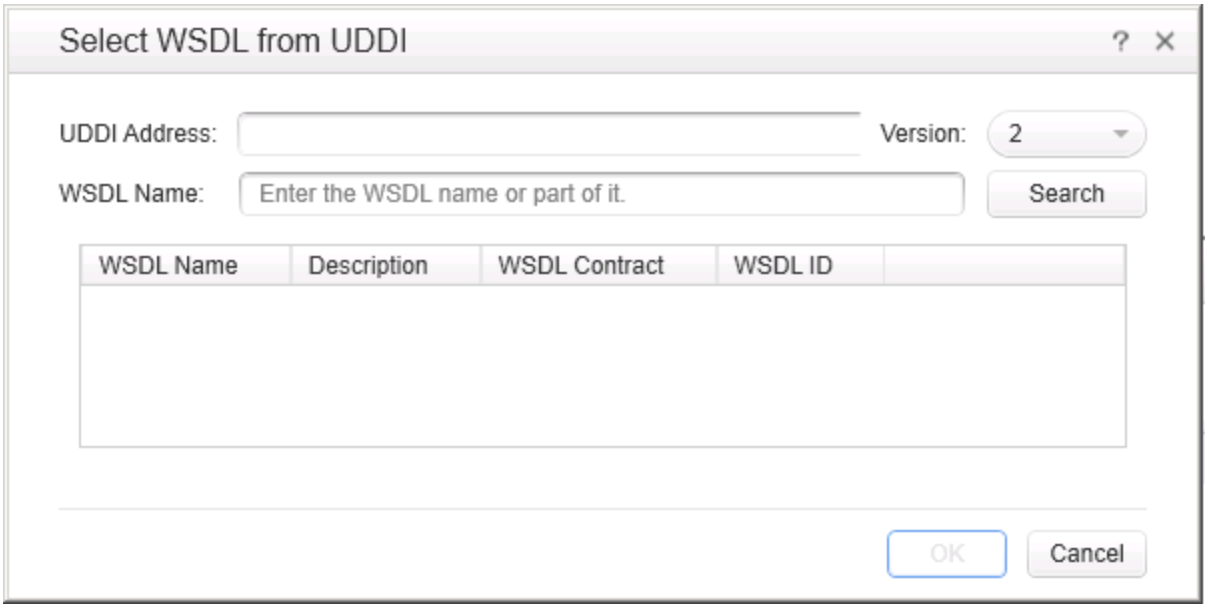
To access	1. Create or open a test or component. 2. Select **Import WSDL > Import WSDL from URL or UDDI.**
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 472

The following elements are included:

UI Elements	Description
...	**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".
Address	The URL or UDDI path of the WSDL.
Advanced Settings	Expands the dialog box to show the authentication and proxy settings.
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**.
Import as Server Response	Imports the WSDL as a server response. Its methods will be server responses—not requests. This is useful for asynchronous type Web services.

Select Service from UDDI Dialog Box

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



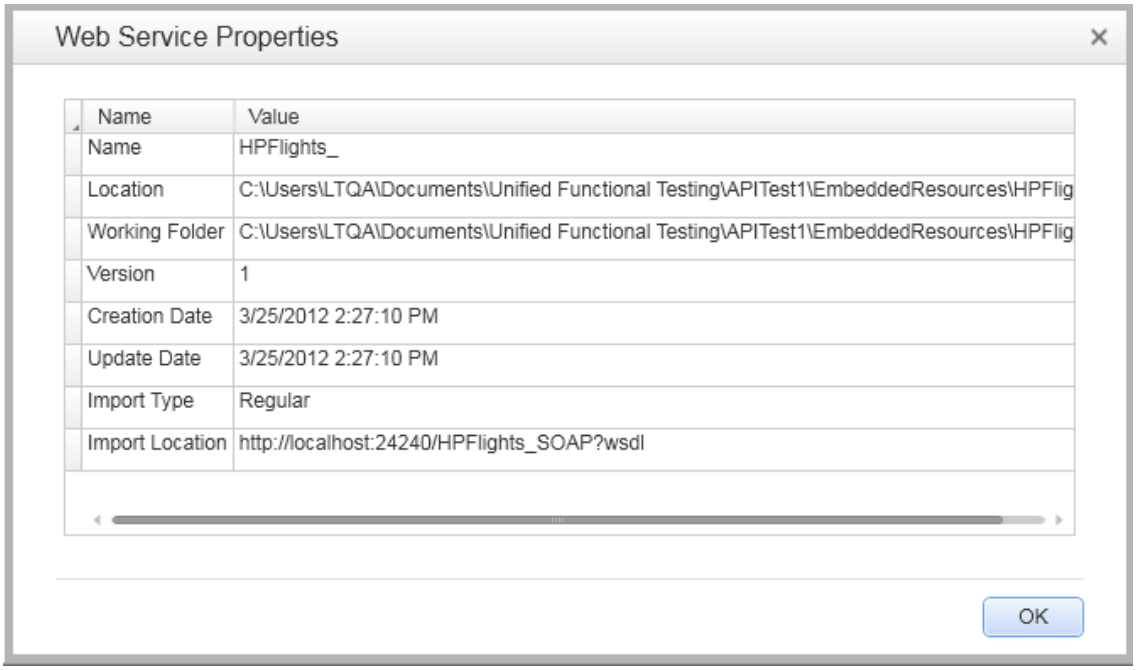
To access	1. Create or open a test or component. 2. Select **Import WSDL > Import WSDL from URL or UDDI.** 3. Select **Import from: UDDI**. 4. Click the  button.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 472
See also	"Import/Update WSDL from URL or UDDI Dialog Box" on page 489

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

UI Elements	Description
UDDI Address	The name or IP address of the UDDI server inquiry API. **Example:** `http://mysite.com:8080/uddi/inquiry`
Version	The UDDI version.
<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**
Service Name	A string in the WSDL name by which to filter the list.
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.

Web Services Properties Dialog Box

This dialog box displays the properties of imported Web services.



To access	1. Create or open a test or component. 2. Import a WSDL. Select **Import WSDL >Import WSDL from URL or UDDI**. 3. Select the parent node of the WSDL in the Toolbox pane. 4. Select **Properties** from the context menu.
Relevant tasks	"How to Import a WSDL-Based Web Service" on page 472

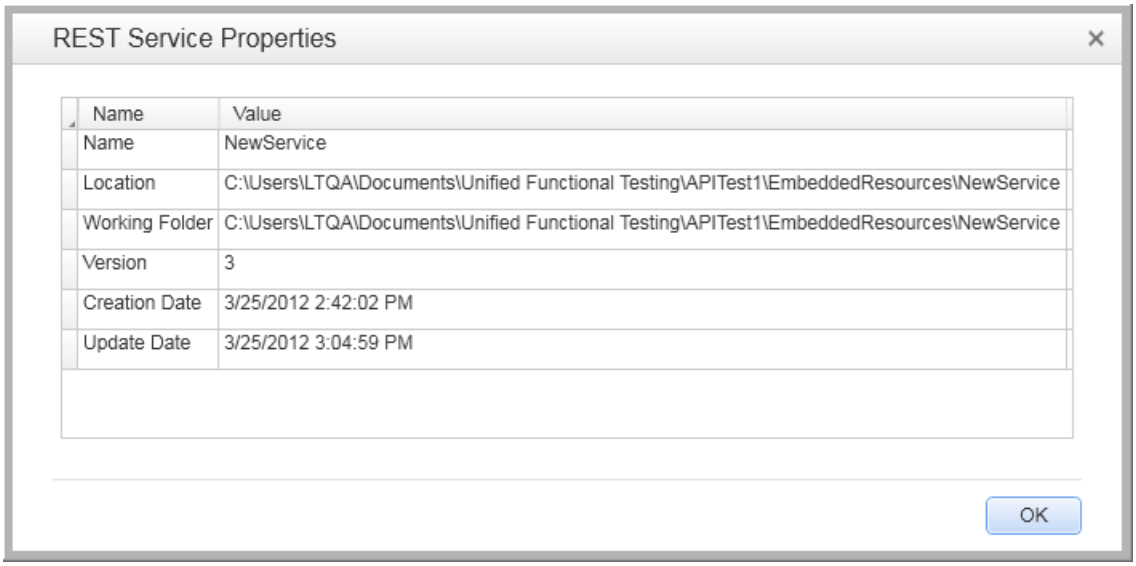
The following elements are included:

UI Elements	Description
Name	The name of the Web service as defined in the WSDL file.
Location	The current location of the WSDL. If it was updated from a location other than the original, it will indicate the new location.
Working Folder	A folder containing a local, working copy of the WSDL. For Web services moved to the shared repository, this is a temporary folder.
Version	The WSDL version.
Creation Date	The date and time in which the WSDL was first imported.
Update Date	The date and time of the latest update. If no updates were done, this value is the same as the **Creation Date**.

UI Elements	Description
Import Type	The way in which the WSDL was imported: **Regular** or **As Server**. For details about importing the WSDL as a server response, see the "Select WSDL Dialog Box" on page 488.

REST Services Properties Dialog Box

This dialog box displays the properties of REST services.



To access	1. Create or open a test or component. 2. Create a REST Service or import a Web Application service. For details, see "Add/Edit REST Service Dialog Box" on page 495. 3. Select the parent node of the service in the Toolbox pane. 4. Select **Properties** from the context menu.
Relevant tasks	"How to Create a REST Method" on page 475

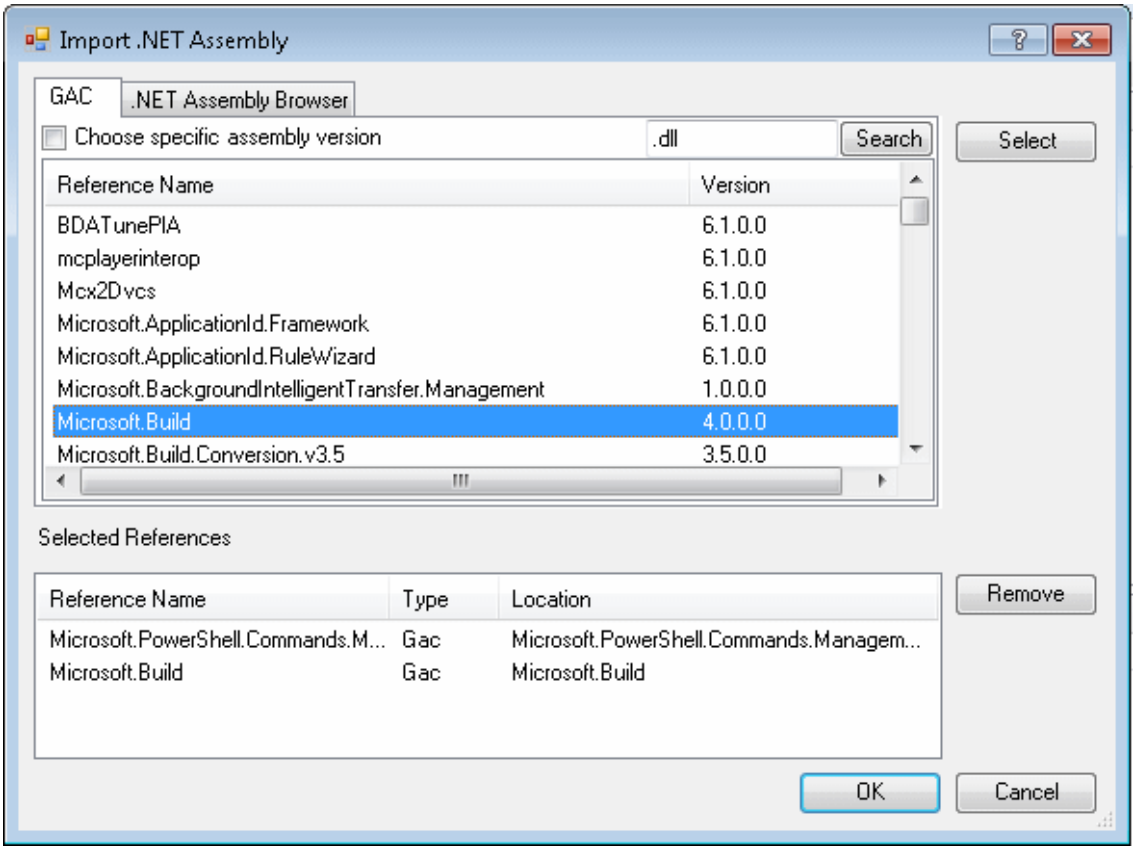
The following elements are included:

UI Elements	Description
Name	The name of the REST service as defined in the REST Service Designer.
Location	The current location of the REST service files.
Working Folder	The folder containing a local, working copy of the test. By default, this is `<Test_Folder>\EmbeddedResources\NewService`.

UI Elements	Description
Version	The current version of the REST service.
Creation Date	The date and time in which the service was created.
Update Date	The date and time of the latest change and refresh. If no updates were done, this value is the same as the **Creation Date**.

Import .NET Assembly Dialog Box

This dialog box enables you to import .NET assemblies in order to use them as activities from the Toolbox pane.



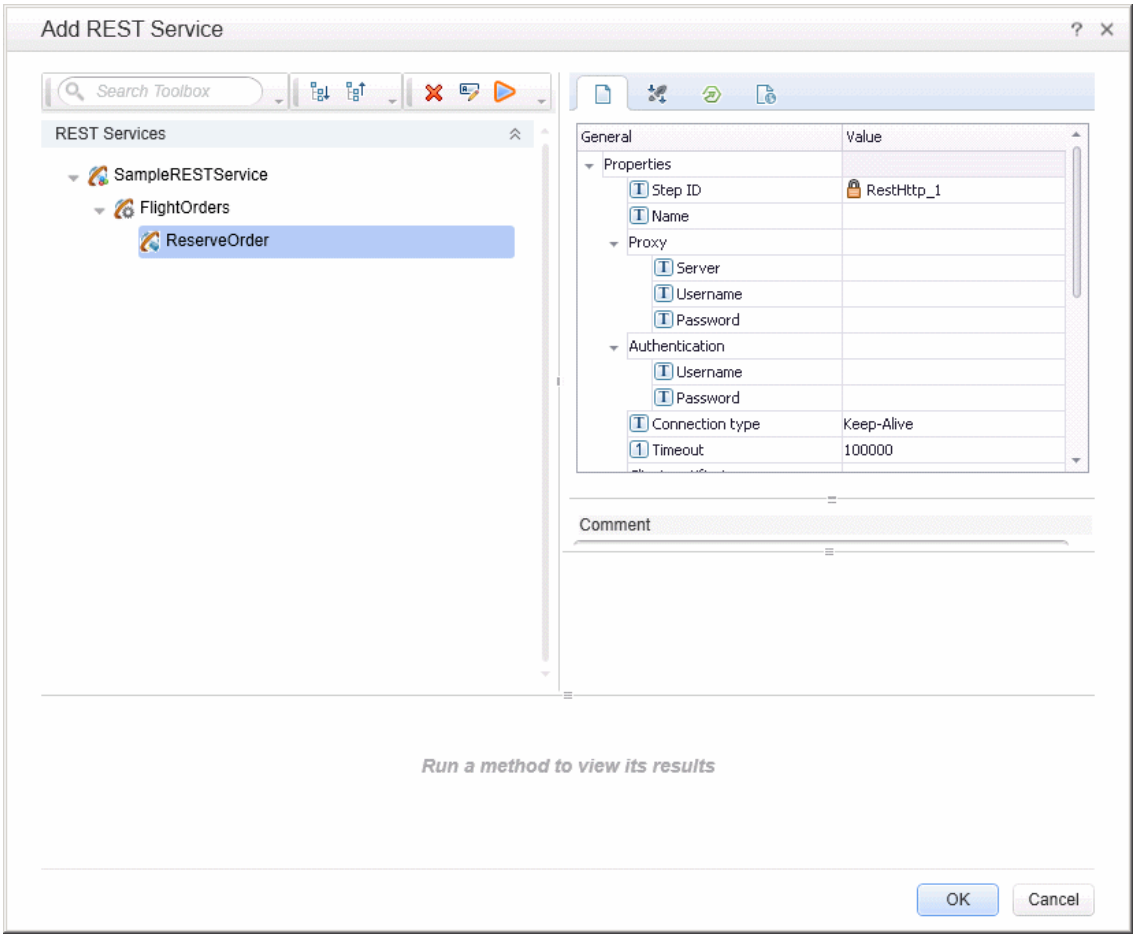
To access	1. Create or open a test or component. 2. Click the **Import .NET Assembly** button on the toolbar.
Relevant tasks	"How to Create a .NET Assembly API Test Step" on page 473

The following elements are included:

UI Elements	Description
GAC tab	Displays a list of the GAC (Global Assembly Cache) assemblies. **Choose specific assembly version.** Shows all versions of the GAC assemblies as separate entries. This enables you to add a specific version of a GAC assembly.
.NET Assembly Browser tab	Enables you to browse for an assembly to import.
Select	Adds the selected assemblies to the **Selected References** list.
Selected References	A list of the GAC assemblies that you selected. The grid shows the following information: • **Reference Name.** The name of the reference as it will appear in the Toolbox pane. • **Type.** GAC or Assembly • **Location.** For `GAC` types, the class; For `Assembly` types, the full path of the DLL.
Remove	Removes the selected assemblies from the **Selected References** list.

Add/Edit REST Service Dialog Box

This dialog box enables you to define a new REST service.



To access	1. Create or open a test or component. 2. Do one of the following: ▪ Click the **Add REST Service** button on the toolbar and select **REST Service Editor** or **Import WADL**. ▪ Select a REST Service or Web Application service node in the Toolbox pane, and choose **Edit Service** from the right-click menu.\
Relevant tasks	"How to Create a REST Method" on page 475

Some user interface elements are available only when you select a specific node. The user elements are (unlabeled elements are shown in angle brackets):

UI Elements	Description
	Expand All. Expands all of the nodes of the parent REST service.
	Collapse All. Collapses all of the children nodes of the parent REST service.

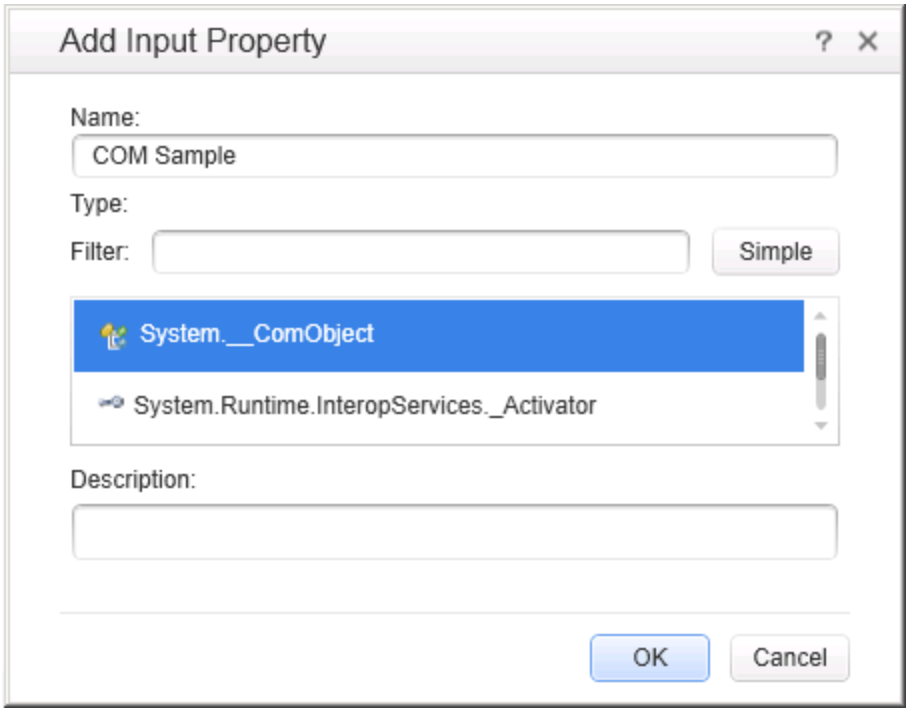
UI Elements	Description
	**Add Method.** Adds a method to the selected REST service node.
	**Add Resource.** Adds a resource to the selected REST service or resource.
	**Delete.** Deletes the selected REST service, resource, or method.
	**Rename.** Renames the selected REST service, resource, or method.
	**Run Method.** Runs the selected method and shows the results in the lower part of the pane. **Note:** Available only when you select a method.
<properties pane>	The properties of the REST service methods. For details, see the "Properties Pane" on page 184.
<REST Services tree>	Displays a list of the REST services that match the filter condition.
<results pane>	The results of the `Run Method` command. This section also shows the property values and the response.
Filter box	Filters the display by the entered text.

Add Input/Output Property/Parameter Dialog Box

This dialog box enables you to define custom input or output properties for the current step or input or output parameters for a test.

- For adding a property for a Custom Code or .NET Assembly step, see "Add Property for Custom Code or .NET Assembly Activities" on next page.
- For adding a test property, see "Add Input/Output Parameter for Test Settings" on page 499.
- For adding a property for a REST activity step, see "Add Input/Output Property for REST Service Steps" on page 501

Add Property for Custom Code or .NET Assembly Activities



To access	1. Create or open a test or component. 2. Add a Custom Code or .NET Assembly activity to the canvas. 3. Select the Properties pane's **Input/Checkpoints** tab. 4. Click the **Add** button and select one of the following: ▪ **Add Input Property** ▪ **Add Output Property**
Relevant tasks	• "How to Create a .NET Assembly API Test Step" on page 473 • "Create a Custom Code activity - optional" on page 384 • "How to Define Test Properties or User/System Variables" on page 59

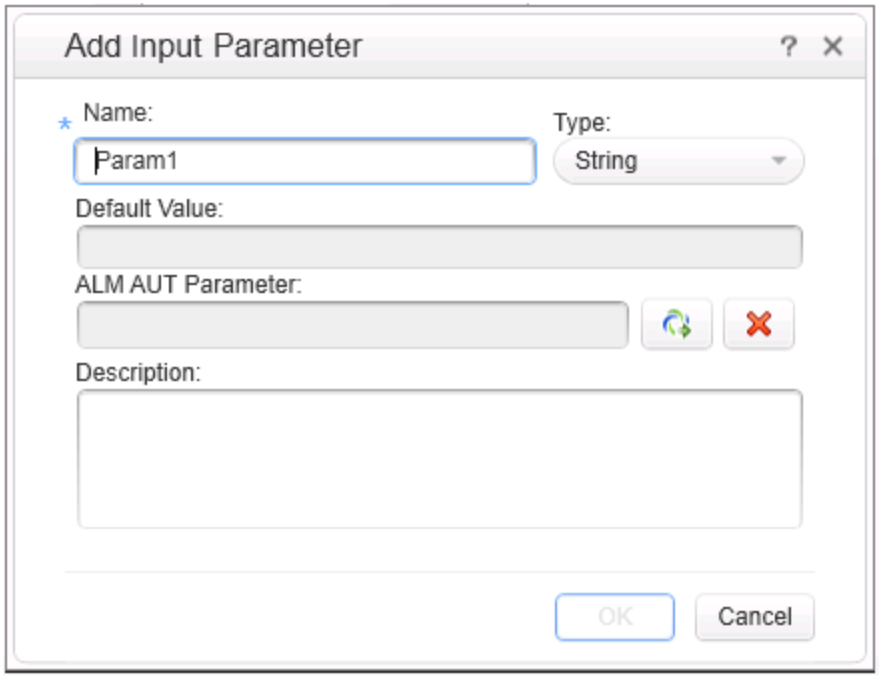
The following elements are included (unlabeled elements are shown in angle brackets):

UI Elements	Description
Name	A name for the property.
Type	For **Simple** types, a drop down list of simple data types, such as `String`, `Integer`, and so forth.
Filter	For **Advanced** types, a textual expression by which to filter the displayed assemblies. For example, to show only those assemblies that begin with the **system** prefix, enter `system.`

UI Elements	Description
Simple	Shows the simple data types, such as `String`, `Integer`, and so forth.
Advanced	Shows an extensive list of types, including all .NET types and DLLs referenced by the test, and all types from the imported assemblies.
<list of properties>	A list of the types, including .NET types, types from DLLs referenced by the test, and all types of the imported assemblies.
Description	A description of the property.

Add Input/Output Parameter for Test Settings

The image below shows the dialog used to add an input parameter. This example also shows the options available when adding a test parameter to a test stored on an ALM server with Lab Management.



To access	1. Create or open a test or action. 2. Select the **Start** or **End** steps in the canvas. 3. Click the Properties pane's **Test Input/Output Properties** tab. 4. Click the **Add** button and select one of the following: ▪ **Add Input Parameter** ▪ **Add Output Parameter** (not relevant for ALM AUT Parameters)
Relevant tasks	"How to Perform Server-Side Execution" on page 290.

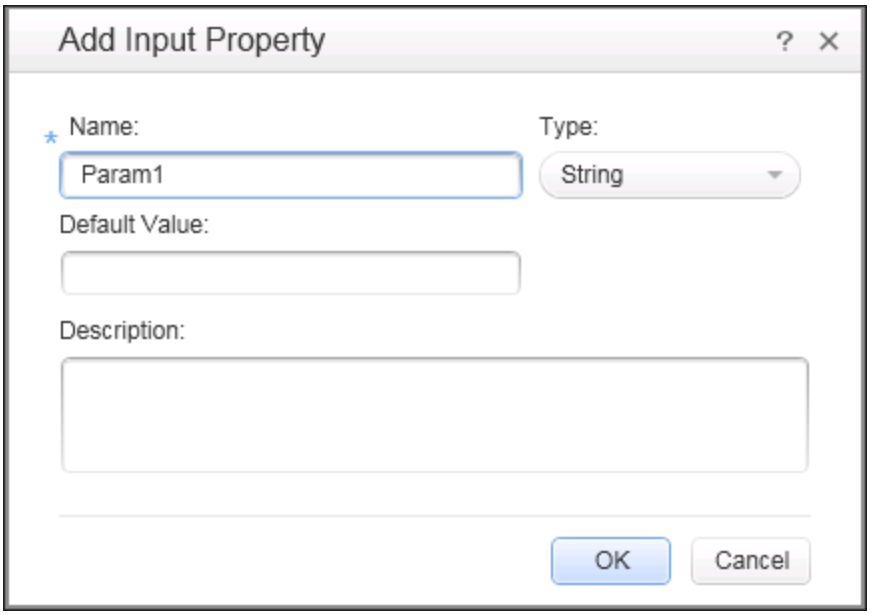
See also	• "Server-Side Execution " on page 285 • "Select AUT Parameter Dialog Box" on page 299

The following elements are included:

UI Elements	Description
Name	A name for the parameter. A default name for the parameter is provided.
Type	A drop down list of simple data types, such as `String`, `Integer`, and so forth. **Note:** For parameters linked to ALM AUT parameters, the type will always be `String`.
Default Value	A default value for the parameter, used during the run session if no other value is provided for the parameter. **Note:** For ALM AUT parameters, you can set this value by selecting **Copy default value (override)** in the "Select AUT Parameter Dialog Box" (described on page 299).
ALM AUT Parameter	The path to an ALM AUT Parameter from the **Lab Resources > AUT Environment** module in ALM Lab Management. You cannot edit the text in this box. Use the **Select ALM application parameters** to enter an AUT parameter. **Note:** Available only if Service Test is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
	**Select ALM application parameters.** Opens the Select AUT Parameter dialog box for selecting a parameter from an AUT environment defined in the **Lab Resources > AUT Environment** module. For details, see "Select AUT Parameter Dialog Box" on page 299. **Note:** Available only if Service Test is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.

UI Elements	Description
	**Remove link to ALM application parameters.** Removes the linking of the selected parameter from ALM. This parameter will now be an ordinary test parameter. **Note:** Available only if Service Test is connected to an ALM server that has the Lab Management module, and a test stored on that server is selected in the solution explorer.
Description	A description of the parameter.

Add Input/Output Property for REST Service Steps



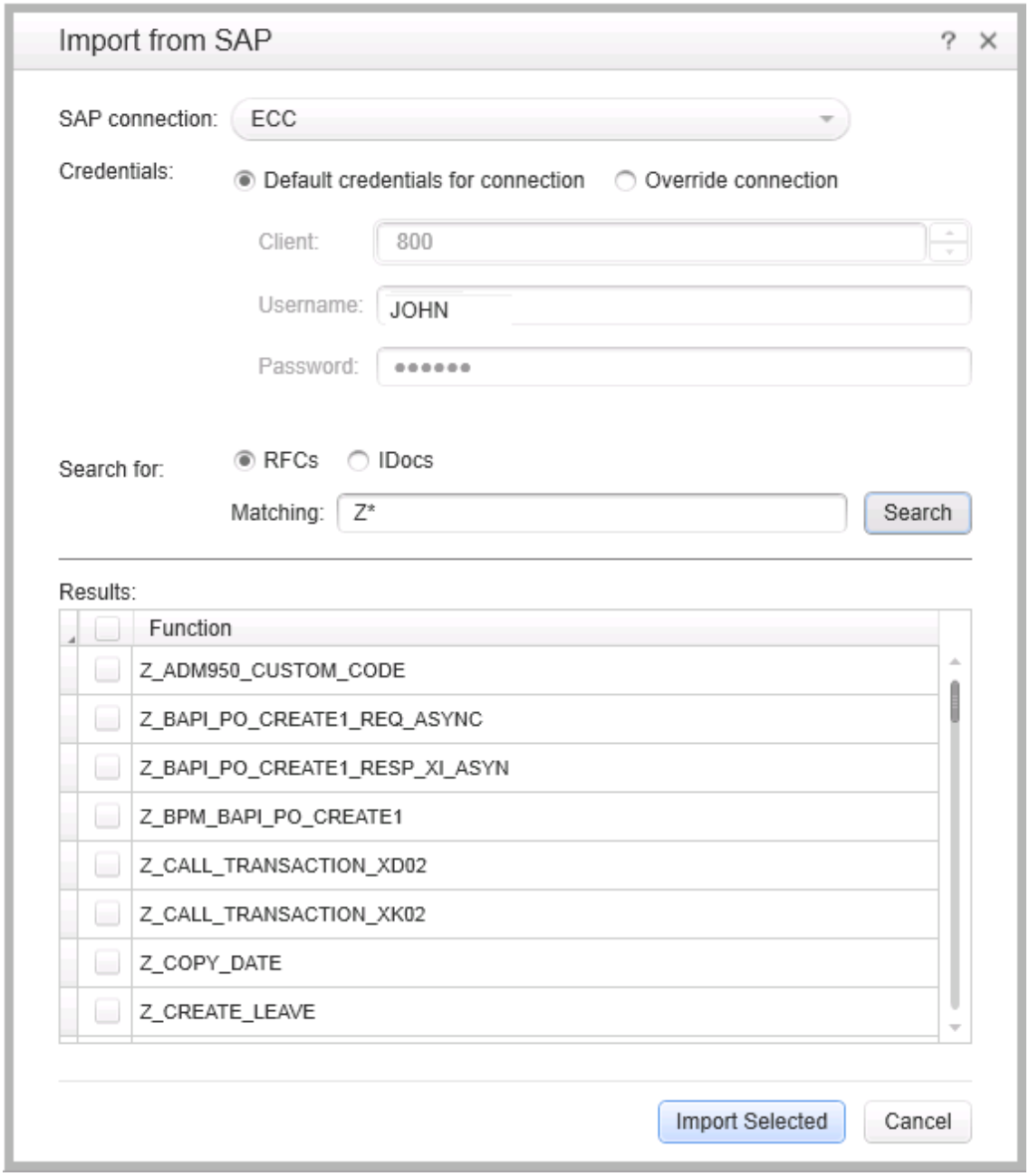
To access	1. Create or open a test or component. 2. Open the Add REST Service Dialog box . 3. Create a REST method, service, or resource. 4. Click the **Custom Input/Checkpoints** tab in the right pane . 5. Click the **Add** button and select one of the following: ▪ **Add Input Property** ▪ **Add Output Property** (for REST methods only)
Relevant tasks	"How to Create a REST Method" on page 475.
See also	"Add/Edit REST Service Dialog Box" on page 495

The following elements are included:

UI Elements	Description
Name	A name for the property.
Type	A drop down list of simple data types, such as `String`, `Integer`, and so forth.
Description	A description of the property.

Import from SAP/Update Dialog Box

This dialog box enables you to import an SAP IDoc or RFC into the Toolbox pane or update an existing one from a new location.



To access	**To import a new RFC or IDoc:** 1. Define an SAP Connection. For details, see "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 275. 2. Select **Tools > Import from SAP**. **To update an RFC or IDoc from a different location:** 1. Select the RFC or IDoc in the Toolbox pane. 2. Select **Update from** from the right-click menu.

Relevant tasks	"How to Create an SAP Test Step" on page 485

Some user interface elements are only available when you select a specific node or if you import a new RFC or IDoc. The user elements are:

UI Elements	Description
SAP Connection	A drop down containing the names of the SAP connections defined in the "SAP Connections Pane (Options Dialog Box > API Testing Tab)" on page 275 (read-only).
Credentials	The credentials to use for accessing the server: • **Default credentials for connection.** Use the client and credentials defined for the connection. • **Override connection.** Manually provide client and credential information for this import.
Search for	The type of document to search for: **IDOC** or **RFC**.
Matching	The matching condition. Use an asterisk, `*`, as a wildcard.
Search	Begins the search based on the matching conditions.
Results	The IDOCs or RFCs that matched the condition.
Import Selected	Imports the RFCs or IDocs indicated with a check mark into the **Local Activities > SAP** section of the toolbox.
Update	Updates the selected RFC or IDoc.

Troubleshooting and Limitations - Test Activities

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

This section includes the following:

Web Services

- You cannot import WSDL files with names that are restricted by the Windows operating system. This list includes: `CON`, `PRN`, `AUX`, `NUL`, `COM1`, `COM2`, `COM3`, `COM4`, `COM5`, `COM6`, `COM7`, `COM8`, `COM9`, `LPT1`, `LPT2`, `LPT3`, `LPT4`, `LPT5`, `LPT6`, `LPT7`, `LPT8`, and `LPT9`.

  Workaround: Rename the WSDL file before the import.

- The following WSDLs are not supported:
  - WSDL version 2.0
  - WSDLs containing a `<appInfo>` element.
  - RPC Encoded WSDLs configured as Asynchronous Web services.
  - WSDLs authenticated by HTTP digest on Apache servers.
- When opening many tests in the same solution using one or more of the same Web services, Service Test may develop a memory leak.

  Workaround: Move the Web service to the file system or ALM repository. To do this, import a Web service and then, in the **Toolbox** pane, right-click the Web Service activity and select **Move to > File System Activities** or **ALM/QC Activities**.

- For a Web service imported as a server response:
  - RPC type WSDLs cannot be imported as a server response. If you attempt to update a service from an RPC encoded WSDL, it will create a duplicate entry.
  - If you end a `ServiceTest.exe` process during the listening stage, after the binding was added, the binding will not be removed from the system.

    Workaround: Remove the binding manually using a utility such as `httpcfg.exe` or `netsh.exe`.
  - When working with SSL, certificates from a file are not supported. If you move the test to another machine, the certificate will not be saved with the test.

    Workaround: Add the certificate to the local machine store before the listener starts, and remove it at the end of the listening process.
  - The error messages displayed when importing a WSDL are not localized.

REST Services

- Importing a schema to a REST, HTTP, or SOAP checkpoint may remove the links of the input properties.
- Copy and Paste in the Toolbox pane is not supported for REST services.
- XPath checkpoints are not supported for HTTP and REST activities.
- Links that are defined in the Toolbox pane are deleted on the canvas.
- A link defined in the Toolbox pane cannot be deleted.
- Links between input and output properties of the same REST method in which the source property is deleted without removing the link, the link expression still remains in the destination property. If you save the test in this state, you may be unable to reopen it.

  Workaround: Delete the link explicitly either before or immediately after deleting its source property.
- When defining a REST method prototype in the Toolbox pane - in order to use the `Trim`, `Ignore`, or `Stop test` options, select the check box in the **Validate** column, in the Checkpoints pane.
- When running a REST method using the **Run Step** command, checkpoints of dynamic property values in the method linked to other property values (input or output) are ignored.

Web Applications (WADL)

The following elements are not supported when importing your WADL. UFT does not import these elements into the WADL hierarchy inside your test:

- `Grammars` element
- `Resource_type` element
- `Link` element

Note: Any child elements of these elements are also ignored by UFT and not added to your WADL inside the test.

In addition, if you use the href attribute to link to other elements, you must refer to an element in the same WADL file. Linking between WADL files is not supported.

For more details on WADL elements, see http://www.w3.org/Submission/wadl/

.NET Assemblies

- .NET 4 assemblies are not supported. They cannot be imported or referenced.
- Importing or adding references to 64-bit .NET assemblies is not supported.

Chapter 25: Reusable Actions in Service Test

This chapter includes:

Concepts

Actions Overview

Actions are sequences of operations that you perform within the context of a test, consisting of either one or multiple steps. Calls to actions placed in your test enable you to test a specific area or function of your application. In this way, your test can check the user interface and the application functionality, such as opening a flight reservation system using based on a Web Service.

Actions also allow you call a repeated activity, such as logging in to a Web site multiple times. Each time the action is called, the test calls the action steps, its properties, and its data. Instead of recreating the process and entering its data each time you call the action, the call to an action refers to the original action instead of having to define a new action.

An action contains its own test coding, including all of the steps in that action and all necessary objects for its execution.

When you insert a call to an action into your test, the test adds the action to the Solution Explorer. As a result, you can reuse the action multiple times within a test or call it from another test.

Actions can also be saved as business components to be used for Business Process Testing.

When to Use Actions

Actions are useful when you need to repeat a specific activity multiple times. For example, if you are testing your Web service which requires a login, you can insert a call to a log-in action multiple times instead of inserting a new log-in step each time it is needed.

Actions also enable you to parameterize repeated steps within your test. For example, if you want to test your Web service for multiple users, you can call an action connected to a data source with user information for multiple users. The call to an action gives you the ability to provide data for a specific step or feature to ensure it works for multiple scenarios or users. Thus, you can not only test the functionality of the processes of your application, but you can also test that the individual elements within that process work correctly.

Actions also enable you to modify a specific element within an application without affecting the test. When the call to action is inserted in the test of a larger process, if the application element changes, the entire test does not. The action properties are updated to reflect the change in the application element. However, the application process test steps do not need to be modified.

Calling Actions from Other Tests

You can call both actions defined within your current test or defined in other tests. When you call an action from another test, by default its data is read-only. Using the Property View, you can enable the action's data for editing. For details, see "How to Use Actions in a Test" on page 513.

Nesting and Repeating Actions

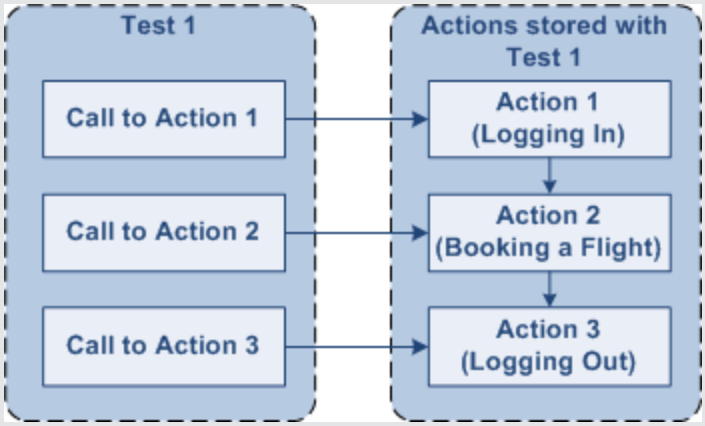
You can nest additional actions within existing ones. This allows you to create specialized actions and call them at the desired point within your test.

You can also set the loop properties for individual actions, independent of the loop properties in the main Test Flow. You click in the action's flow, and specify the loop properties in the Property View.

Example

Suppose you want to test a Web service which enables customers to book a flight and purchase tickets. This requires you to test several processes and features: request a Web service, log in, choose flights, process customer information, output a complete reservation, and log out. Testing this service also requires you to test both application processes and user interface elements. Furthermore, you might want to test the capability of a user, such as a travel agent to use the service to make multiple reservations.

Your test might look similar to this model:



Action Placement

Service Test provides reusable actions that can be called multiple times within a test or by other tests. These actions make it easier to maintain your tests, because when an object or procedure in your application changes, the modified element is updated in all calls to the action.

Note: This is different from GUI Test (formerly QuickTest Professional) actions, which allow you to indicate whether the action should be reusable or not.

Calls to Existing Actions

When you plan a test or multiple tests, you may realize that each test requires identical, repeated activities, such as logging in. Instead of inserting these steps three times in separate places or tests (and adding checkpoints, parameters, and data) separately, you can create an action one time and store it with the test. You then populate the action with activities from the Toolbox. After you are satisfied with the action you have created, you can also call it from other tests.

If you call an action from an external test that has data assigned to its properties, you can indicate whether you want the data from action's data sheet to be imported as a local, editable copy, or whether you want to use the (read-only) data from the original action.

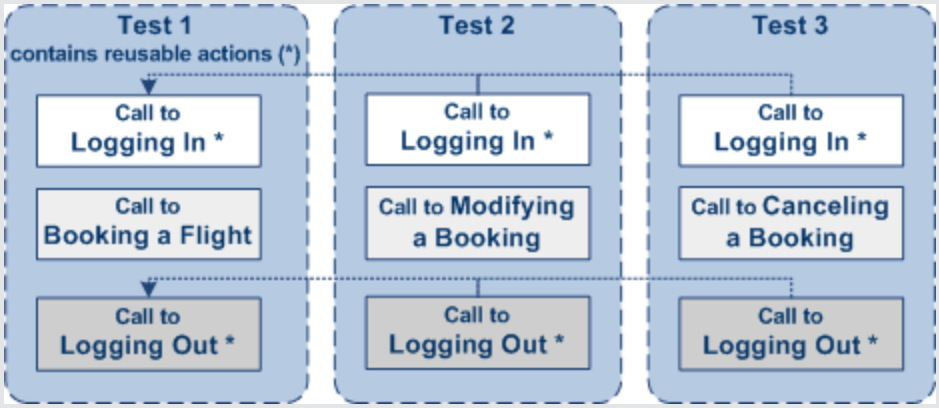
To modify the steps of an action from an external test, you must open the test in which the action is stored and make your modifications there. The modifications apply to all tests that call that action.

For more details, see "How to Use Actions in a Test" on page 513.

Example

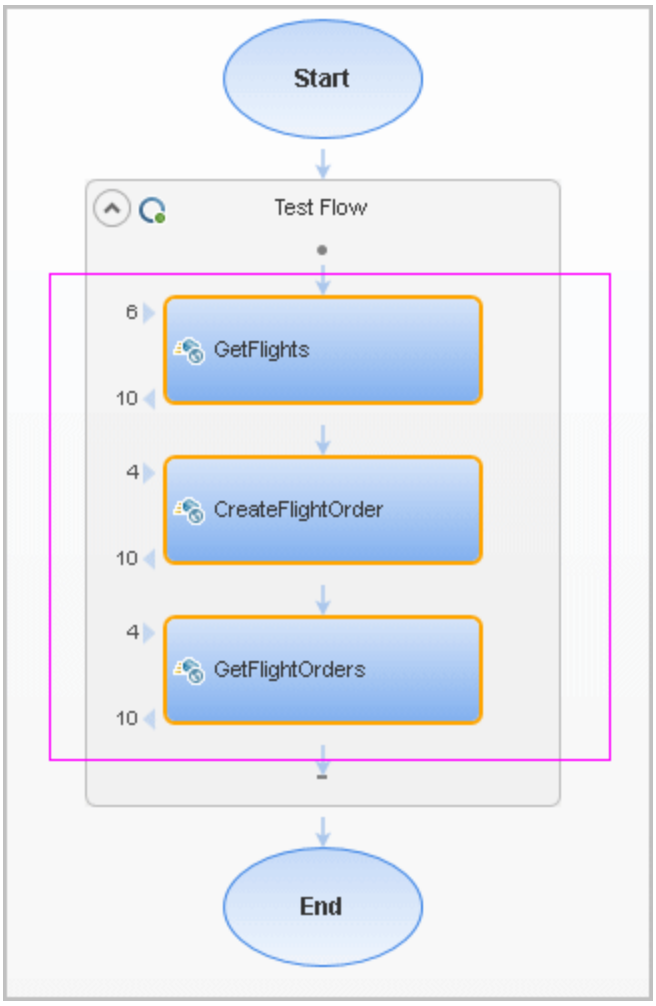
Suppose you want to create the following three tests for flight reservation--booking a flight, modifying a reservation, and deleting a reservation. While planning your tests, you realize that for each test, you need to log in and log out of the site.

Test 1 would contain three actions (Logging In, Booking a Flight, and Logging Out). Test 2 would contain three actions (Logging In, Modifying a Booking, and Logging Out). Test 1 would contain three actions (Logging In, Cancelling a Booking, and Logging Out). The Logging In and Logging Out actions are defined in the same test, and called as external actions by all three tests.



Combining Steps into Actions

Multiple steps can be combined together to constitute an action. For example, when designing a test to check your Web service for booking flights, you may find that certain actions and processes are repeated multiple times. You can select these steps on the canvas and group them in an action. You can also select multiple steps using the standard Windows multiple selection methods, with the **SHIFT** and **CTRL** keys.



After selecting the steps you copy/cut and paste them into the action's tab. Your test can then call the action, and apply loop behavior to the action as a single unit.

For more details, see "How to Use Actions in a Test" on page 513.

Actions Using the Data Table

You can use the Data pane tables to provide property values for your actions. Each action uses its own data sources—not the test's data source.

> **Tip:** If you want to use the same data for the test and in a specific action, you must define the data source path twice, once for the test and once for the action. See "Defining Data " on page 609.

By default, data from actions called from other tests, are not visible in your current test. You can expose the data by enabling the **Allow other tools to override the data** option in the actions' test.

As a result, you can view the data in the Data Pane. It is marked with an icon  indicating that this is data from an action.

An **Update** option allows you to reload the action's data in your test when it changes in its original location. This only applies to data that you did not make editable. For details, see "How to Use Actions in a Test" on next page.

Note: You can switch data from read-only to editable or vice versa. However, when changing from editable to read-only, any changes made to the data are lost.

If an action is called repeatedly in the test, only one set of the action's data is displayed in the Data Pane. If you make changes to the data at the action's original location and it is not marked as editable, then the changes are applied to all calls to this action.

For details on how Service Test handles data in a test, see "Data Handling Overview" on page 609.

Considerations for Working with Actions

Inserting Actions

You should consider using actions if:

- You intend to use an identical or virtually identical procedure or action in more than one test.
- You want to edit the data in a specific call to an action that you call multiple times within your test.

Naming Actions

You may want to rename the actions in your test with descriptive names to help you identify them. It is also a good idea to add detailed action descriptions. This facilitates inserting actions from one test to another. You can rename an action by right-clicking on the action in the Solution Explorer and selecting **Rename**.

When renaming an action, make sure to use the following naming conventions:

- Cannot begin or end with a space
- Cannot exceed 1,023 characters
- Cannot contain the following characters: **\ / : * ? " < > | % ' ! { }**

Tasks

How to Use Actions in a Test

This task describes the different operations you can perform to use actions in your test.

This task includes the following steps:

Insert a call to a new action

1. Select **Design > Call to New Action**.
2. Specify the action name and description. Click **OK** to add the call to the action to the canvas.
3. In the Solution Explorer, double-click the action to open its canvas.
4. Add steps to the action. Drag activities from the Toolbox onto the canvas.

> **Note:** You can insert a call to another action, from an existing action call.

Call an existing action or a test

1. Drag an **HP Automated Testing Tools > Call API Action** or Test or **Call GUI Action or Test** activity onto the canvas.
2. Open the Properties pane (**CTRL+ALT+P**).
3. In the Input/Checkpoints tab, click the **Select Action or Test** button.
4. Select the desired test and action. Click **OK**.

Set action properties

To set an action call's properties, you must set the values for each of the steps separately.

1. Select an activity within the action whose properties you want to set.
2. In the Properties tab, provide values or data drive the properties. For details, see the "Properties Pane Tabs" on page 186.
3. The property values that you set will be the default values used when you add a call.

> **Note:** If you reload the service using the **Refresh** button, the new version may add or remove properties. Properties that remain unchanged, will retain the default values.

Copy or move steps

Copying or moving steps is ideal if you want to copy or move existing steps into another location in the Test Flow or to an action.

1. Make sure the actions into which you want to copy or move the steps, are open. To open an action, double-click on it in the Solution Explorer or select **Open** from the action's context menu in the test.
2. Select one or more steps that you want to copy or move. To select multiple steps, do one of the following:
   - Drag the mouse pointer around the steps. A transparent rectangle is be drawn over the selected area.
   - Press **SHIFT** while selecting the beginning and ending steps to be grouped. All steps between the endpoints are grouped.
3. Select **Copy** or **Cut** from the context menu.
4. Place the cursor in the desired location. Select **Paste** from the right-click menu.

Remove a call to an action

In the test canvas, select the call to the action and press **DELETE** or right-click and select **Delete**

Delete an action

In the Solution Explorer, right-click the action and select **Delete**. The action is removed from the Solution Explorer and all calls to the action are removed. Edited data from the removed step is moved to the main node of the Data Pane as described in "Actions Using the Data Table" on page 511.

Enable an action's data for editing when called by another test

To view or override an external action's data, you must expose it in its original location. This only applies to Excel type data sources, and only available for tests—not business components.

1. Define data for the action that you will be calling. Assign the data manually or use data driving. For details, see "How to Assign Data to Test Steps" on page 621.
2. In the Data Pane, select the data source's node.
3. Open the Properties pane (**CTRL+ALT + P**) and select the **Allow other tools to override the data** option. The Data pane creates a local copy of the data which can be edited.

Override data from an action called by another test

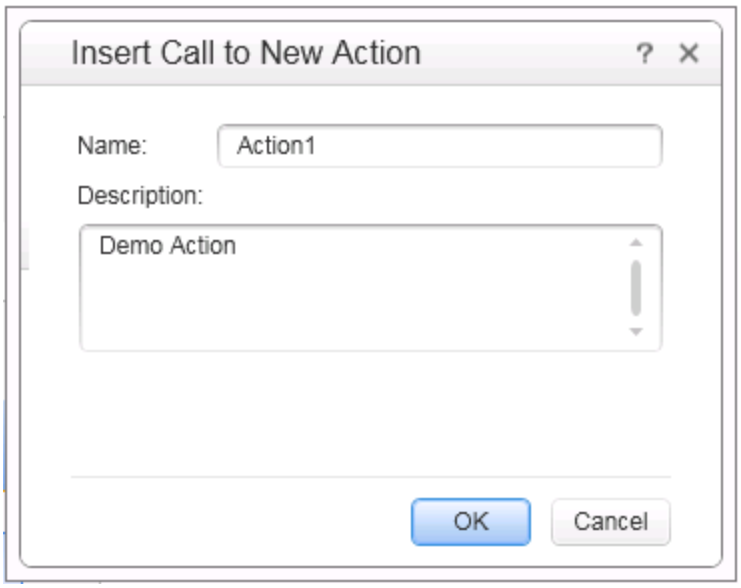
1. Make sure the action that you want to call has its data exposed. For details, see the above step.
2. Open the test in which you want to add the call to the action.
3. In the **HP Automated Testing Tools** node, drag a **Call API Action or Test** or a **Call GUI Action or Test** activity onto the canvas.

4. In the Properties pane Input/Checkpoints tab, click the **Select Action or Test** button.
5. In the Select Action or Test dialog box, browse for the test and select the desired action. Click **OK**.
6. Open the Properties pane and select the **Use a local, editable copy** option.
7. Edit the data tables in the Data pane.

Reference

Insert Call to New Action Dialog Box

This dialog box enables you to create a new action for your test.



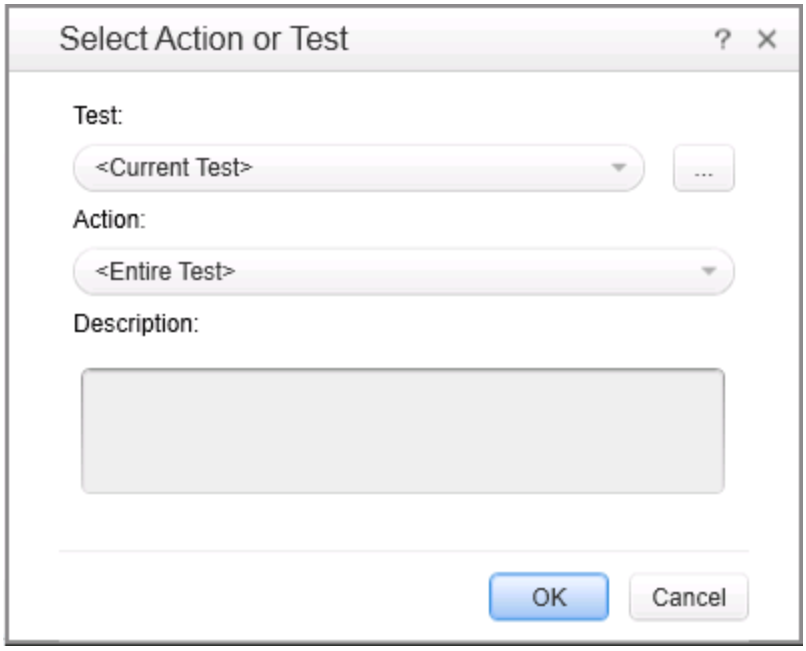
To access	1. Create or open a test or component. 2. Do one of the following: ▪ **Design > Call to New Action** ▪ Click the **Insert Call to New Action** button
Relevant tasks	"Insert a call to a new action" on page 513

User interface elements are described below:

UI Element (A-Z)	Description
Name	A name for the action as it will appear in the Solution Explorer and canvas.
Description	A detailed description of the action.

Select Action or Test Dialog Box

This dialog box enables you to add a call to a copy of an action, or a call to an existing action within your test.



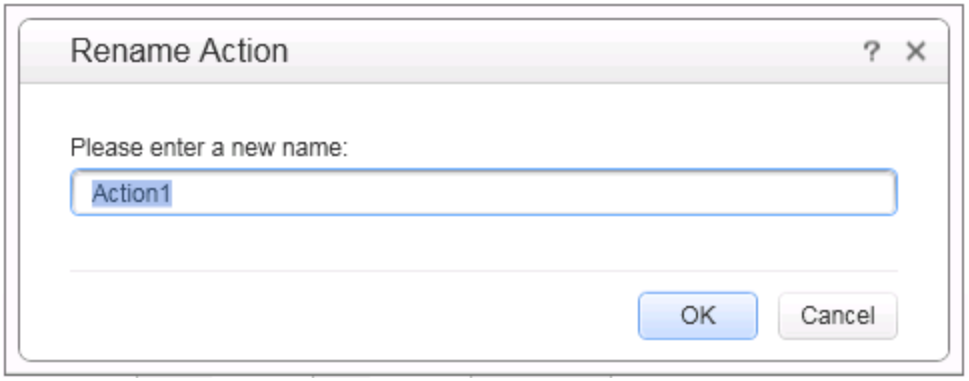
To access	1. Create or open a test or component. 2. Drag an **HP Automated Testing Tools > Call API Action or Test**activity onto the canvas. 3. Open the Properties pane. 4. Click the **Select Action or Test** button.
Relevant tasks	"Call an existing action or a test" on page 513

User interface elements are described below:

UI Element (A-Z)	Description
Test	The name of the test from which you are calling the action.
...	**Browse.** This allows you to browse your test library to find a test.
Action	A drop down list of the actions available for the selected test. You can view a list of the available actions in the Solution Explorer.
Description	The original description given to the action (read-only).

Rename Action Dialog Box

This dialog box enables you to rename an action.



To access	Right-click on an action name in the Solution Explorer.
Relevant tasks	"How to Use Actions in a Test" on page 513
See also	"Considerations for Working with Actions" on page 512

User interface elements are described below:

UI Element (A-Z)	Description
New Name	A new name for the action.

Actions Context Menu

The context menu for action steps is described below:

UI Element (A-Z)	Description
Cut	Cuts the action from the current flow, and places it on the clipboard.
Copy	Copies the selected action to the clipboard.
Delete	Removes the selected action from the current flow.
Open	Opens a tab for the selected action. This allows you to add steps and set properties for the action. **Note:** For external actions, stored in other tests, a message window asks if you want to add the external test to your solution. This creates a local copy of the action allowing you to modify its data.

Chapter 26: Updating Services and Assemblies

This chapter includes:

Concepts

Updating Web Services

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

When you update a service, Service Test first 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 pane, as an additional port for the service.

If the port paths (`<service name:port name>` combination) do not match, and security was set for on the port level, Service Test opens the **Update Port Security Wizard**. For details, see the "Update Port Security Wizard" on page 528.

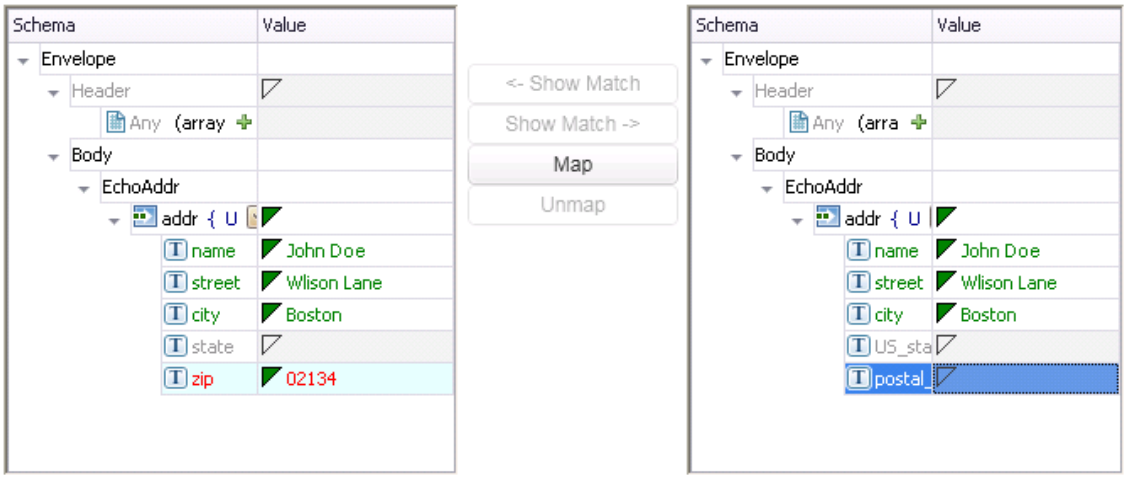
If HTTP transport information such as the endpoint was changed, Service Test will update this automatically, as long as the port path is the same.

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 enables 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 enables 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 through Service Test Management, you must update the service in ALM—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 next page.

You can also update the data assigned to properties. When working with an Excel data source, you can update the data in the Data pane. For details, see "New/Change Excel Data Source Dialog Box" on page 98.

REST Service Conflicts

Using REST services that you added to the Toolbox pane, you can create tests with minimal configurations. For details, see "How to Create a REST Method" on page 475.

If you modify a REST step's properties after incorporating it into your test, it will no longer match the method in the Toolbox. You may want to keep the change, or you may want to remove the conflicts so that your step will match the method defined in the toolbox.

You may encounter conflicts in the following areas:

- Adding or removing an input or output property
- Renaming an input or output property
- A change in the REST service URL
- A change in the HTTP request/response body type or schema
- Internal links between a REST property and its inner HTTP elements (if you are working with a test created in Service Test 11.51 or earlier
- HTTP methods

The Resolve REST Method Steps wizard enables you to view the conflicts and decide what to do with the conflicts in your test step—keep, remove, or map them. The wizard lets you resolve property-related conflicts. Other conflicts, such as discrepancies in the URL values, the HTTP body, and so forth, are resolved automatically.

For wizard details, see the "Resolve REST Method Steps Wizard" on page 535.

For task details, see "How to Resolve Conflicts in a REST Steps" on page 525.

Updating SAP RFCs or IDocs

You can update SAP RFCs and IDocs from their original location or from another connection or location.

The **Update** option automatically updates the item from its original location. Service Test loads the information for the RFC/IDoc with the same names, from the original connection.

The **Update from** option lets you specify different connection information, or a name of a different RFC/IDoc on the same server.

If Service Test detects an inconsistency between the original and updated RFC/IDoc names or their properties, it enables the Update Step wizard. This wizard lets you map the original and new items. For details, see "How to Update an SAP RFC or IDoc" on page 526.

Tasks

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" below
- "Update the service " below
- "Run the Update Port Security wizard - optional" below
- "Run the Update Step wizard" on next page

1. **Prerequisites**

   If you want to update the WSDL from ALM, make sure you have an open connection to the ALM server. For details, see the "ALM Connection Dialog Box" on page 335.

2. **Update the service**

   - To update a service from its original location, select its main node in the Toolbox pane. Choose **Update WSDL** from the context menu.
   - To update the service from a different location, or if the **Update WSD**L option is not available:

     i. Select the service's main node in the Toolbox pane.

     ii. Choose **Update WSDL from > URL or UDDI** or **Update WSDL from > File or ALM Application Component** from the right-click menu.

     iii. Navigate to the WSDL and click **OK**.

     iv. Accept any warning or informational pop-ups.

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

3. **Run the Update Port Security wizard - optional**

   If Service Test detects a change in the port path (<`service name:port name`> combination) the **Update Port Security** Wizard opens. The wizard only opens if you configured security settings for the original service. Follow the steps of the wizard to resolve all of the Service/Port conflicts. For details, see "Update Port Security Wizard" on page 528.

   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 pane, delete them again manually, after the update.

- If you manually remove a service with conflicts from the Toolbox pane, you will no longer be able to resolve the conflicts using the wizard.

4. **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.

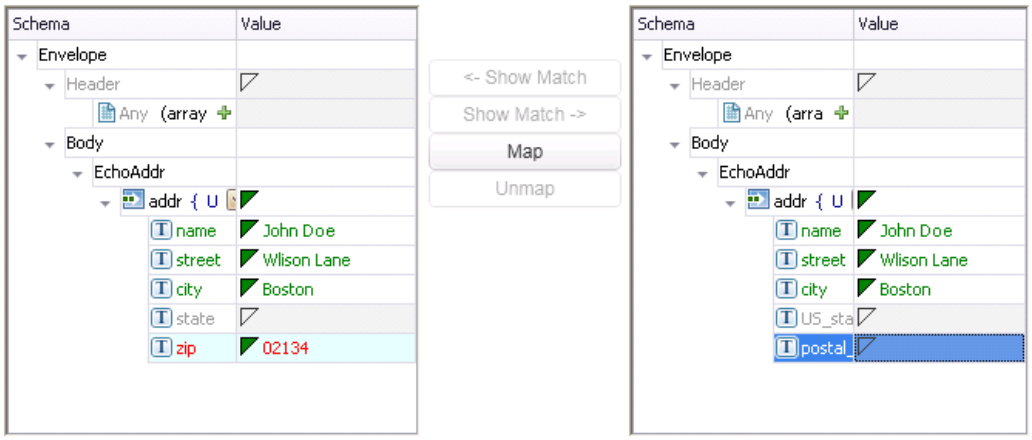
   For details about the wizard, see the "Update Step/Activity Wizard" on page 530.

   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** window 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 one in the **Original Operation** pane. Click **Next**. Properties for which conflicts were detected 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 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 **New Properties** pane 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.

How to Update a .NET Assembly

This task describes how to update a .NET assembly with a newer version. The updating of .NET assembly is similar to the importing of an assembly described in "How to Create a .NET Assembly API Test Step" on page 473.

This task includes the following steps:

- "Select the assembly to update" below
- "Open the Import .NET Assembly dialog box" below
- "Import a .NET assembly" below
- "Handle warnings - optional" below
- "Modify custom code - optional" below

1. ### Select the assembly to update

   In the Toolbox pane, expand the **.NET Assemblies** node and select the assembly you want to update.

2. ### Open the Import .NET Assembly dialog box

   Click the **Import .NET Assembly** button on the toolbar. For user interface details, see "Import .NET Assembly Dialog Box" on page 494.

3. ### Import a .NET assembly

   Click the **.NET Assembly Browser** tab. Click **Browse** and locate the `.dll` or `.exe` file. Click **Open** in the Open dialog box to add it to the **Selected References** list. Click **OK** to begin the update.

4. ### Handle warnings - optional

   If the updated .NET assembly is missing types that are in use by existing activities, you will need to modify the step properties. The canvas displays warning icons adjacent to the steps whose properties use the missing type.

5. ### Modify custom code - optional

   If you wrote custom code in an event handler, you may need to modify the step properties. If the updated .NET assembly is missing types that are used by the custom code, make sure to modify the code to use only the types that are available.

How to Resolve Conflicts in a REST Steps

This task describes how to update a REST method step that was changed.

This task includes the following steps:

- "Prerequisites" below
- "Modify the step as required" below
- "Open the wizard" on next page
- "Run the wizard" on next page

1. ### Prerequisites

   Create one or more prototype methods for REST services. Drag them onto the canvas to create REST method steps. For details, see "How to Create a REST Method" on page 475.

2. ### Modify the step as required

   Modify the REST service step as required: add or remove properties, change property values,

and so forth. If there are conflicts, the canvas will display warning icons next to the relevant steps.

3. **Open the wizard**
   - To resolve conflicts for the selected step, click the warning icon in the top right corner of the REST method step in the canvas. Select **This step should be resolved. Resolve step.** The Resolve REST Method Steps wizard opens.
   - To resolve conflicts for all steps created with the prototype, right-click the method in the Toolbox pane and select **Apply Changes to all Steps**.
4. **Run the wizard**
   a. Select the steps that you want to resolve (only relevant when opening the wizard through the Toolbox pane).

      The left pane, **Before Changes**, shows the step's properties before they were resolved. The right pane, **After Changes**, shows the step's properties after they were resolved.

   b. Proceed with the wizard, resolving or keeping the detected differences.

   For details about the wizard, see the "Resolve REST Method Steps Wizard" on page 535.

How to Update an SAP RFC or IDoc

This task describes how to update an SAP RFC or IDoc from its original or from a different location.

This task includes the following steps:

- "Update the RFC/IDoc from its original location" below
- "Update the RFC/IDoc from a different location" below
- "Run the Update Step wizard" on next page

Update the RFC/IDoc from its original location

To update an item from its original location, drill down to the actual RFC or IDoc in the Toolbox pane. This assumes that the location and name of the RFC or IDoc did not change. Choose **Update** from the context menu.

Update the RFC/IDoc from a different location

To update an RFC or IDoc from a different server, or from the same server but a different RFC or IDoc:

1. Select the RFC or IDoc in the Toolbox pane and choose **Update from** from the context menu. The Update <Item_Name> dialog box opens.
2. To change the connection or server information, select **Override connection** and specify the details. Alternatively, you can change the default connection information in the "SAP Connections Pane (Options Dialog Box > API Testing Tab)" described on page 275.
3. To select a different RFC or IDoc, search for it and select it in the bottom pane. For details, see the "Import from SAP/Update Dialog Box" on page 502.
4. Click **Update**. Accept any warning or informational pop-ups.

Run the Update Step wizard

If a test step contains conflicts because an RFC or IDoc name or property changed, then the canvas marks the step with a warning marker.

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

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

2. In the wizard's **Select Activity** page, click in the **New item** area and select the operation that corresponds to the one in the **Original item** pane. Click **Next**. Properties for which conflicts were detected are highlighted in red.

3. If there are conflicted input properties, the **Update Input Properties** page opens. 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 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**.

4. If there are conflicted output properties, the **Update Output Properties** page opens. Select a property for which there is a conflict, highlighted in red. In the **New Properties** pane 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**.

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

For details about the wizard, see the "Update Step/Activity Wizard" on page 530.

Reference

Update Port Security Wizard

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

To access	1. Create or open a test or component. 2. Import a WSDL using the **Import WSDL** toolbar button. 3. In the Toolbox pane, click on one of the service's ports and select **Security Settings** from the right-click menu. 4. In the Security Settings for Port dialog box, configure the port's security. 5. Select the service's parent node in the Toolbox pane and select **Update** (or **Update from)** from the right-click menu. **Note:** This wizard only opens if there is a discrepancy between the service or port names, and if the port's security settings were configured.
Relevant tasks	"How to Update a Web Service" on page 523
Wizard map	This wizard contains: "Map Services and Ports Page" > "Finish Page"
See also	"Updating Web Services" on page 521

Map Services and Ports Page

This wizard page enables you to map a port or service from the new WSDL, with the original WSDL'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/Activity Wizard" on page 530.
Wizard map	This wizard contains: **Map Services and Ports Page** > "Finish Page"

User interface elements are described below (unlabeled 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.
<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.
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.
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.

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 Port Security Wizard" on page 528.
Wizard map	This wizard contains: "Map Services and Ports Page" > Finish Page

User interface elements are described below:

UI Elements	Description
Ignores unresolved services/ports	Ignores unresolved conflicts. If you intend to remove the unresolved services, you can enable this option. **Tip:** To return to the wizard screens to resolve all of the services or ports, click **Back**.
Finish	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.

Update Step/Activity Wizard

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

To access	1. Create or open a test or component. 2. Use the **Tools** menu to import a Web service or an SAP RFC or IDoc. 3. Drag an operation/activity from the Toolbox pane onto the canvas and set input values. 4. Select the parent node and select **Update** or **Update from** in the shortcut menu. Accept any warning popup messages. 5. Click on the alert adjacent to the step  . 6. Click on the text **The step must be resolved. Resolve step**. **Note:** The wizard will open only when there is a change in property names that were assigned values.
Relevant tasks	"How to Update a Web Service" on page 523

Wizard map	This wizard contains: "Select Operation/Activity Page" > "Update Input Properties Page" > "Update Output Properties Page" > "Finish Page"
See also	"Updating Web Services" on page 521

Select Operation/Activity Page

This wizard page enables you to map an operation/activity from the new service/RFC or IDoc, with the original one. The **Next** button proceeds to the next conflict.

Important information	• General information about this wizard is available here: "Update Step/Activity Wizard" on previous page. • The wizard may indicate that it could not find any compatible operations in the updated contract. This may be a result of different SOAP versions.
Wizard map	This wizard contains: **Select Operation/Activity Page** > "Update Input Properties Page" > "Update Output Properties Page" > "Finish Page"

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

UI Elements	Description
<search box>	**For Web services:** Enables you to filter the operations list. For example to display only those operations containing the term `Get`, type `Get` into the text field.
New operation/item pane	**For Web services:** A tree hierarchy of the new service, its ports, and its operations. **For SAP IDocs and RFCs:** An expansion of the new item in the toolbox showing the SAP Connection name, item type (`RFC` or `IDoc`), and item name.
Original operation/item	**For Web services:** A tree hierarchy of the operation in the selected test step. **For SAP RFCs and IDocs:** The original RFC or IDoc.
Apply to all steps of this type	Converts all test steps using the original operation/item, to steps using the new operation/item name. If you do not select this option, you will need to resolve the same conflict in other test steps, one-by-one. **Note:** This option is only enabled when you have multiple steps in the canvas that use the original operation/activity.

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/Activity Wizard" on page 530.
Wizard map	This wizard contains: "Select Operation/Activity Page" > **Update Input Properties Page** > "Update Output Properties Page" > "Finish Page"

User interface elements are described below (unlabeled 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. **Tip:** 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. **Note:** When mapping to an operation from another WSDL, the new WSDL must have the same SOAP version as the original.
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/Activity Wizard" on page 530.
Wizard map	This wizard contains: "Select Operation/Activity Page" > "Update Input Properties Page" > **Update Output Properties Page** > "Finish Page"

User interface elements are described below (unlabeled 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.
<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.
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.

UI Elements	Description
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.
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.
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.
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.

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/Activity Wizard" on page 530.
Wizard map	This wizard contains: "Select Operation/Activity Page" > "Update Input Properties Page" > "Update Output Properties Page" > **Finish Page**

User interface elements are described below:

UI Elements	Description
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**.

UI Elements	Description
Finish	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.

Resolve REST Method Steps Wizard

This wizard enables you to resolve differences between REST service steps and the prototype method upon which they were based. The wizard detects changes such as added, removed, or renamed properties and helps you resolve them. For details about the conflict types, see "REST Service Conflicts" on page 522.

To access	Create or open a test or component. As a prerequisite: 1. Create a prototype for a REST service method. For details, see "How to Create a REST Method" on page 475. 2. Drag one or more REST service methods from the Toolbox pane to the canvas. 3. Select a REST method in the Toolbox pane and choose **Edit Service** from the right-click menu. In the Edit REST Service dialog box, modify the method's properties and/or values. To start the wizard: • **For the selected step only:** Click on the alert in the bottom right corner of the REST method step  and click the text **The step should be resolved. Resolve step**. • **For all steps using the prototype method:** In the Toolbox pane, right-click the method and select **Apply Changes to all Steps**.
Relevant tasks	"How to Resolve Conflicts in a REST Steps" on page 525
Wizard map	This wizard contains: "Select Steps Page" > "Resolve Conflicts Page" > "Finish Page"

The wizard's global interface elements are described below:

UI Elements	Description
Cancel	Closes the wizard, discarding all resolutions that you made until this point.

UI Elements	Description
Finish	Does the following: • Performs the automatic resolutions for remaining test steps whose conflicts were not resolved manually. • Ignores all remaining conflicting properties, by applying the **Keep** action to all of them. • Advances to the wizard's **Finish** page.
Next	Proceeds to the conflicts of the next step. The wizard displays a separate **Resolve Conflicts** page for each step.

Select Steps Page

When you start the wizard from the Toolbox pane, it enables you to resolve conflicts for all steps that use the selected method. This wizard page enables you to include or exclude specific steps in which to apply the resolutions.

Important information	• General information about this wizard is available here: "Resolve REST Method Steps Wizard" on previous page. • The ability to select the steps to resolve, is only available when you open the wizard from the Toolbox pane—not from the alert on the canvas. • The wizard only checks those REST steps that are based on the prototype selected in the Toolbox pane.
Wizard map	This wizard contains: **Select Steps Page >** "Resolve Conflicts Page" > "Finish Page"

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

UI Elements	Description
<test tree>	A tree hierarchy of the test steps. The following steps of the wizard will only affect the selected steps.
Check All	Selects all test steps in which a conflict was detected.
Uncheck All	Clears the check box for all test steps in which a conflict was detected, excluding them from the wizard.

Resolve Conflicts Page

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

Important information	• General information about this wizard is available here: "Resolve REST Method Steps Wizard" on page 535. • The wizard repeats this page for each of the methods selected in the previous step.
Wizard map	This wizard contains: "Select Steps Page" > **Resolve Conflicts Page** > "Finish Page"

User interface elements are described below (unlabeled elements are shown in angle brackets). The **Keep**, **Remove**, and **Map** buttons relate to both the Input and Output properties.

UI Elements	Description
<conflict resolutions>	A list of all the resolved conflicts—the type of conflict, and the resolution. The text color indicates the conflict status: • **Green.** Conflicts that were resolved automatically or resolved by a **Keep**, **Map**, or **Remove** action. • **Red.** Conflicts that were not yet resolved or removed.
Input Properties / Output Properties	A list of all of the step's input/output properties. • **Before changes.** A list of the step's original properties • **After changes.** A list of the step's properties after the conflicts were resolved. The text color indicates the property status: • **Green.** Properties that were resolved automatically or resolved by a **Keep**, **Map**, or **Remove** action. • **Red.** Properties for which a conflict was found, and not yet resolved. • **Black.** Properties for which no conflict was found.
Keep	Accepts the conflict for use within the test. The resulting test step's properties do not correspond to the prototype.

UI Elements	Description
Map	Assigns the value of the selected property in the **Before changes** pane, to the property in the **After changes** pane After you map a property, the grid displays it in green. **Note:** This button is enabled only if you select a red or green property in the **Before changes** pane, and a black or green property in the **After changes** pane.
Remove	Removes the selected conflicting property from the current test step.

Finish Page

This wizard page indicates whether you successfully resolved the conflicts between the prototype and the test steps.

Important information	General information about this wizard is available here: "Resolve REST Method Steps Wizard" on page 535.
Wizard map	This wizard contains: "Select Steps Page" > "Resolve Conflicts Page" > **Finish Page**

User interface elements are described below:

UI Elements	Description
Finish	Closes the wizard and applies the resolutions to the test step. **Note:** If you click the **Finish** button before resolving the conflicts, the wizard applies a **Keep** action to all conflicts, and performs an automatic resolution for non-property related conflicts.

Troubleshooting and Limitations - Updating

This section describes troubleshooting and limitations for updating services and assemblies.

REST Services

- When comparing a REST step to its prototype - a difference in the request/response body contents is not considered a conflict. Therefore, when resolving a REST step whose contents do not match the body contents of the prototype, the body contents will not be affected.

Chapter 27: Web Service Security

This chapter includes:

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?

Security Levels

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

For task details, see "How to Set Security for a Specific Step" on page 547.

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 enables 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 Scenario Overview" on next page.

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

For details on the built-in security scenarios see one of following sections:

- "Web Service Scenario Overview" on next page
- "WCF Service (CustomBinding) Scenario Overview" on page 544
- "WCF Service (Federation) Scenario Overview " on page 544
- "WCF Services (WSHttpBinding) Scenario Overview " on page 545

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.

This section also includes:

Web Service Scenario Overview

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 "Result Tab (Properties Pane)" on page 207.

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 "Result Tab (Properties Pane)" on page 207.

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 enables 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 Step" on page 547.

WCF Scenario Settings

This section describes the WCF security scenarios. It includes:

WCF Service (CustomBinding) Scenario Overview

The **WCF Service (CustomBinding)** scenario enables the highest degree of customization. Since it is based upon the `WCF CustomBinding` standard, it enables 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 `NamedPipe` 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, select **HTTPS transport**.

For task details, see "How to Set Security for a WCF Service" on page 550.

For user interface details, see "WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 568.

Note: For WSE3 security configurations, use the WCF Service(CustomBinding) Scenario. For details, see "How do I test a WCF service?" on page 554.

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:

- 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.

- Define an STS binding in the **Referenced binding** field.

For task details, see "How to Set Security for a WCF Service" on page 550.

For user interface details, see "WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 568.

WCF Services (WSHttpBinding) Scenario Overview

Note: The **WCFService (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 **WCFService (CustomBinding)** scenario.

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

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.

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

For user interface details, see "WCF Service (WSHttpBinding) Scenario (Security Settings for Port <Port_Name>) Dialog Box)" on page 570.

Advanced Security Settings

This scenario's settings let you customize a **WCFService (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 573.

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 Step" below.

This task includes the following steps:

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

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 pane and select **Security Settings** from the context 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 559.

   c. For Web Service type scenarios, add tokens, signatures, and encryption. For details, see "Web Service Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 560.

5. **Save the settings - optional**

   Click **Save** to save the settings to an `.stss` file.

How to Set Security for a Specific Step

This task describes how to set security properties for an individual test step.

This task includes the following steps:

- "Add a step to which security can be applied" below
- "Open the Security Setting view" below
- "Enable the Service Settings details" below
- "Specify a scenario type" below
- "Configure the security settings" below

1. **Add a step to which security can be applied**

   Drag a Web Service operation or SOAP Request activity onto the canvas.

2. **Open the Security Setting view**

   Open the Security Settings tab  in the Properties pane.

3. **Enable the Service Settings details**

   Clear the **Use the port's security settings** option.

4. **Specify a scenario type**

   Select a scenario from the **Service Details** list: `Web Service`, and so forth. For user interface information, see the "Security Settings for Port <Port_Name> Dialog Box" on page 559.

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" below
   - "How to Set Security for a WCF Service" on page 550

   To see the request and response before and after WSE2 (non-WCF) security is applied, view the Output pane after the test run.

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 (Security Settings for Port <Port_Name> Dialog Box)" on page 560.

This task includes the following steps:

- "Open the Security Settings dialog box or tab" on next page
- "Create a Web Service scenario" on next page
- "Configure the HTTP settings" on next page
- "Define security elements (optional)" on next page
- "Configure the WS-Addressing (optional)" on next page

1. ## Open the Security Settings dialog box or tab

   To set security on a port level, select a Web Service port in the Toolbox and choose **Security Settings** from the context menu.

   To set security for a specific Web Service step already on the canvas, select the step and open the **Security Settings** tab in the Properties pane. Clear the **use the port's security settings** option.

   For a SOAP Request step, click the **Security Settings** tab in the Properties pane.

   For details, see the "Security Settings for Port <Port_Name> Dialog Box" on page 559.

2. ## Create a Web Service scenario

   Select **Web Service** in the **Service Details** list (default).

3. ## Configure the HTTP settings

   In the Properties pane, select the HTTP tab, and set the transport and proxy information. For details, see the "HTTP tab" on page 561.

4. ## 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 542.

     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 562.

   - To add a message encryption (you must first add at least one token), click  and select the token that will do the encryption from 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.

   - Indicate whether or not to include a timestamp when sending the token, signature, or encryption to the server. To exclude it select **Exclude Timestamp**.

5. ## Configure the WS-Addressing (optional)

   Click the **WS-Addressing** tab. Select the relevant version or `None` if WS-Addressing is not used. Provide an alternate destination in the **Reply to** box.

For details about the security elements. see the "Web Service Scenario Overview" on page 542.

For user interface details see the "Security Settings for Port <Port_Name> Dialog Box" on page 559.

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 543.

This task includes the following steps:

- "Create a WCF scenario" below
- "Configure the security settings" below
- "Configure advanced settings - optional" below
- "Save the scenario - optional" below

1. ### Create a WCF scenario

   - For port level security, click on the service's port in the Toolbox pane and select **Security Settings** from the context menu.
   - For step level security, open the **Security Settings** tab in the Properties pane. 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 559.

3. ### Configure advanced settings - optional

   Click the **Advanced** button to configure advanced security settings. For details, see the "Advanced Settings Dialog Box" on page 573.

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 **WCFService (Custom Binding)** scenario is recommended. For additional examples, see "How to Customize Security for WCF Type Web Services" on page 553.

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 Step" on page 547.

This section includes the following:

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

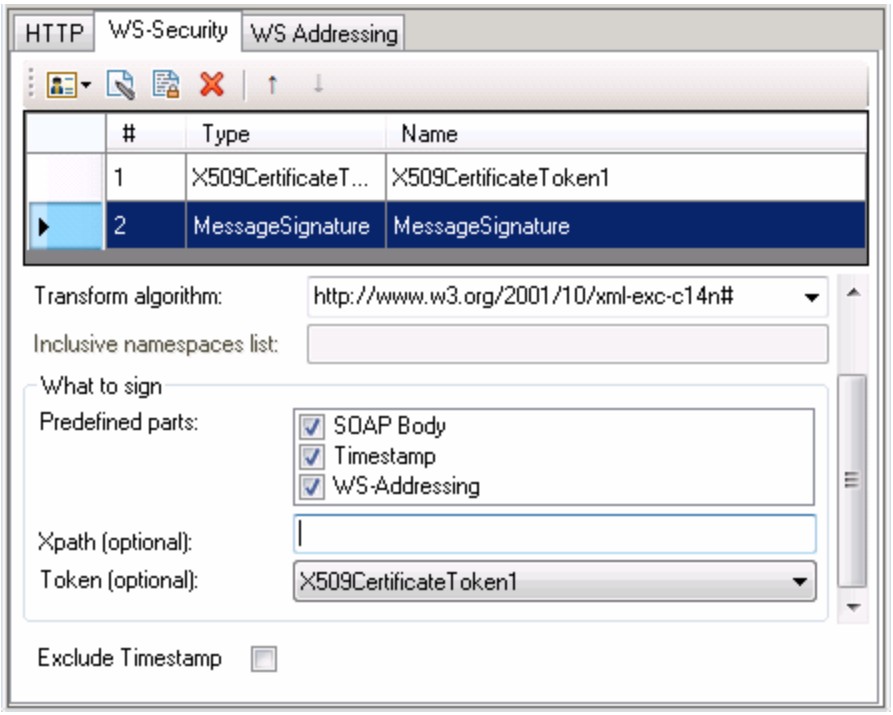
To send a message using a X.509 certificate for a digital signature.

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. Browse for the certificate file. The certificate needs to be installed in the Windows certificate store.
5. Select a **Reference type**. Since this token is used for a signature, the most common type is `BinarySecurityToken`.
6. Click the **Add Message Signature** button . In the **Signing Token** drop down, select the token you created in the previous steps.
7. To sign a specific element with the certificate, scroll down to the **XPath** field and provide an XPath expression.

   You cannot use an XPath expression to sign a timestamp or token that is under the security element of a SOAP request.

   - To sign a **SOAP Body**, **Timestamp,** or **WS-Addressing,** select the check box in the **Predefined parts** area.
   - To sign tokens within the security element, select a token in the **Token (optional)** field, in

the **What to sign** area.



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.

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?

In the **Service Details** list, select a WCF scenario.

If the Federation or WSHttpBinding scenarios are not appropriate, select the **WCFService (Custom 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 **WCFService (WSHttpBinding)** scenario from the **Service Details** list.

Choose a client credential type from the **Client authentication type** list, to use in your binding—`Windows`, `Certificate`, or `Username`. This value corresponds to the **MessageClientCredentialType** property of the WCF's WSHttpBinding parameter.

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 545.

How do I test a WCF service that uses CustomBinding?

Open the **Security View** in the Properties pane and select the **WCFService (Custom 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 550.

How do I test a WCF service that uses netTcp or namedPipe transport?

Select the **WCFService (Custom Binding)** scenario from the **Service Details** list.

Configure the transport to **TCP** or **NamedPipe**.

For details, see "Configure the security settings" on page 550.

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 **WCFService (Federation)** scenario from the **Service 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 (Security Settings for Port <Port_Name> Dialog Box)" on page 568.

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 W3E3 service, and drag the operation onto the canvas.
2. Open the **Security View** in the Properties pane or from the port's shortcut menu. Select the **WCFService (Custom 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—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:** `WSSe-curity11WSTrustFebruary2005WSSecureConversationFebruary2005` (first entry)
      - **Require Derived keys:** Enabled

        For all other fields, use the default settings.

        For details, see "Configure the security settings" on page 550.

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 **WCFService (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 579.
5. Provide the **Expected DNS**.
6. Click the Advanced button and configure the following settings:
   a. **Encoding** tab—**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 550.

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 **WCFService (Custom Binding)** scenario from the **Service Details** list.
2. Set the Transport to **TCP** and the Net Security to **SSL stream security**.
3. Open the Properties pane'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. 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.
6. 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 includes:

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 for 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 **WCFService (Custom Binding)** scenario, or the **WCFService (WSHttpBinding)** scenario with the transport credentials.

How do I test a Web Service that requires Windows authentication at the HTTP level?

For details, see "How to Set Security for a Standard Web Service" on page 548.

1. Select the **Web Service** scenario from the **Scenario Details** list.
2. In the HTTP tab, specify the credentials.

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 573.

How to Set up Advanced Standards Testing

This section provides guidelines for using Service Test in advanced standards testing.

This section includes:

How do I test a Web Service that uses MTOM?

1. Select the **WCFService (Custom Binding)** scenario from the **Service 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 573.

For more details about the scenario, see "Configure the security settings" on page 550.

How do I change the WS-Addressing version of a service?

1. Select the **Web Service** scenario from the **Service 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 547.

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 573.

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:

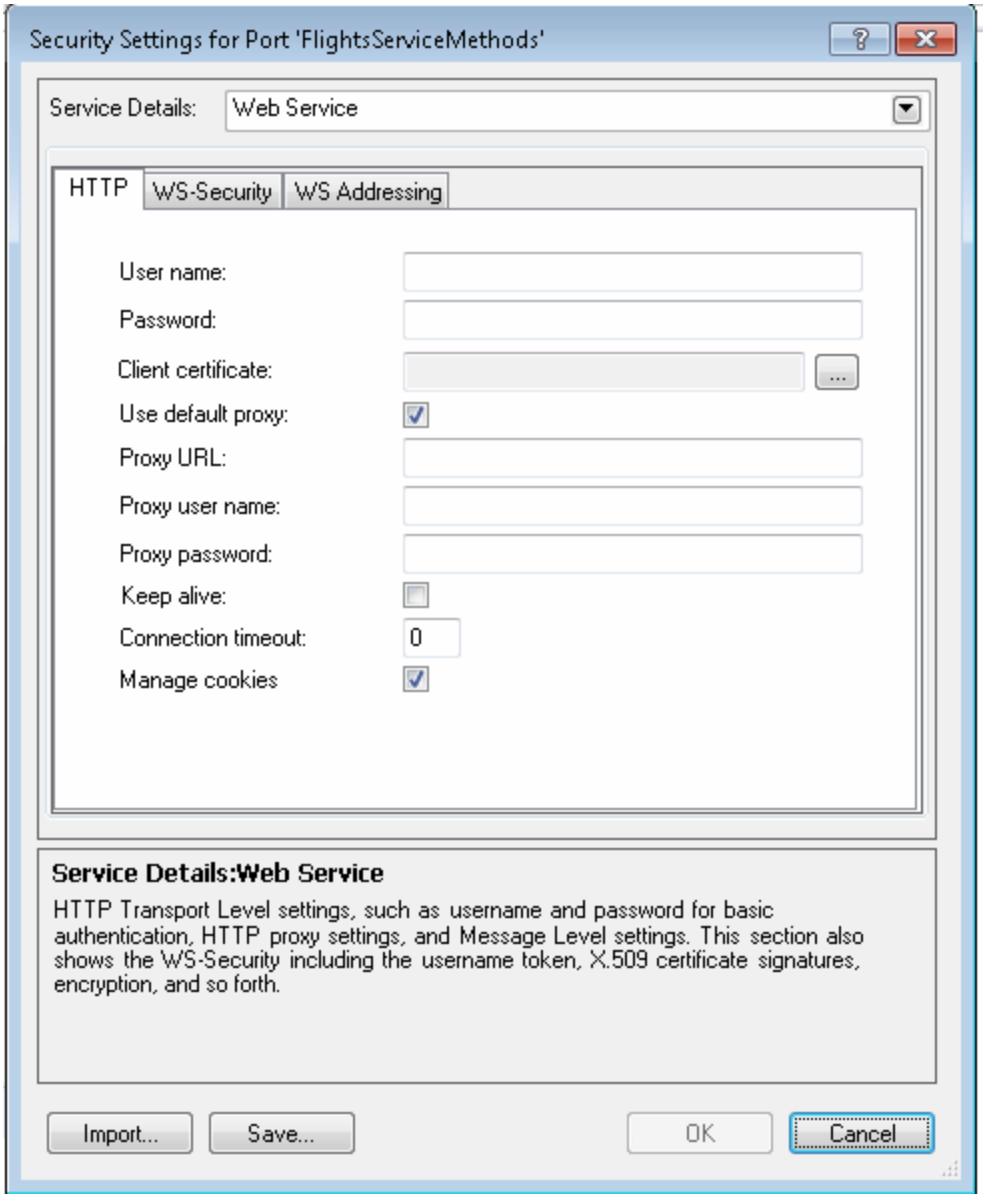
1. Type **`gpedit.msc`** at a command prompt to open your group policy editor.
2. Choose **Computer Configuration > Administrative Templates > Network > SSL Configuration Settings**.
3. Open the only item—**SSL Cipher Suite Order**.
4. Select **Enabled**.
5. The first item in the list is `TLS_RSA_WITH_AES_128_CBC_SHA`

   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 Settings for Port <Port_Name> Dialog Box

Using the Security Settings dialog box, you can configure security settings for all operations in a Web service port. To set the security for a specific step within your test, use the **Security Settings** tab in the Properties pane. For details, see "Result Tab (Properties Pane)" on page 207.



To access	1. Create or open a test or component. 2. Import a Web Service. 3. Select a Web Services port node in the Toolbox pane and select **Security Settings** from the context menu.
Important information	• For details about choosing a security scenario type, see "Security Scenarios Overview" on page 541. • For examples, see "How to Set up Common Web Services Security Scenarios" on page 550.
Relevant tasks	"How to Set Security for a Specific Step" on page 547

User interface elements are described below:

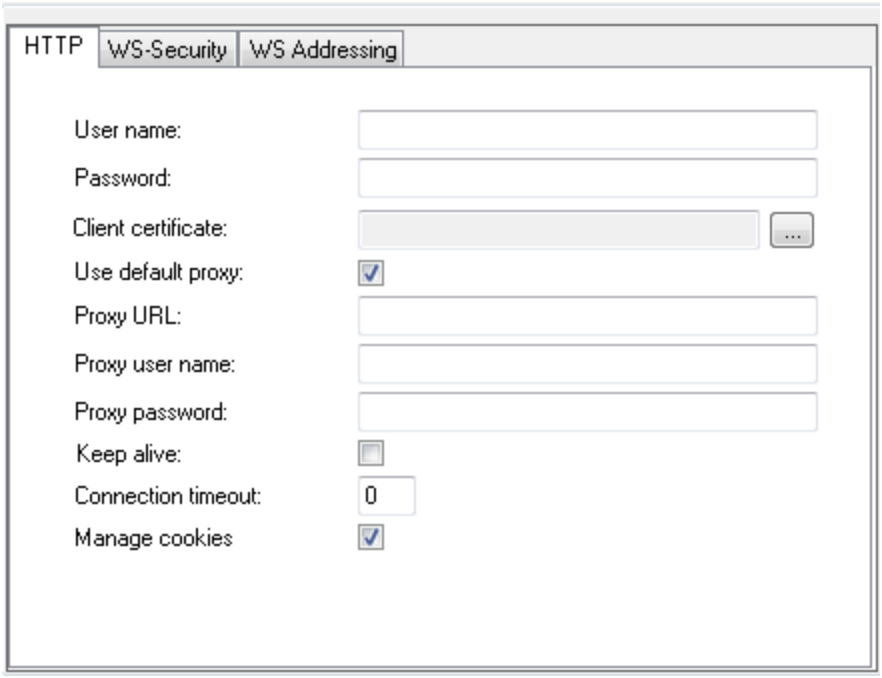
UI Elements	Description
Advanced	Opens the Advanced Settings dialog box. For details, see the "Advanced Settings Dialog Box" on page 573. **Note:** Available only for WCF type Web services.
Import	Loads security settings from a previously saved `.stss` file.
Save	Saves the security scenario settings to an `.stss` (Service Test Security Scenario) file, for use in other tests. If you are connected to ALM, it saves the file together with the test.
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 (Security Settings for Port <Port_Name> Dialog Box)" below • "WCF Service (Custom Binding) Scenario (Security Settings for Port <Port_Name> Dialog Box" on page 566 • "WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)" on page 568 • "WCF Service (WSHttpBinding) Scenario (Security Settings for Port <Port_Name>) Dialog Box)" on page 570

Web Service Scenario (Security Settings for Port <Port_Name> Dialog Box)

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.



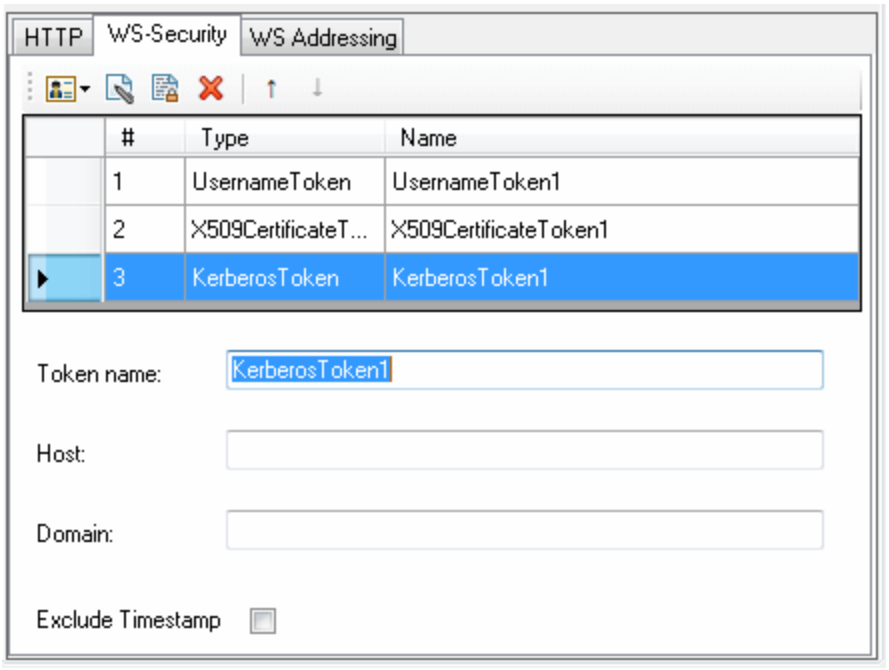
The user interface elements are described below.

UI Elements	Description
User name, Password	The credentials for HTTP authentication such as basic authentication, digest, or NTLM. **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 579.

UI Elements	Description
Proxy URL	The URL and port of the proxy server through which the message must pass. **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).

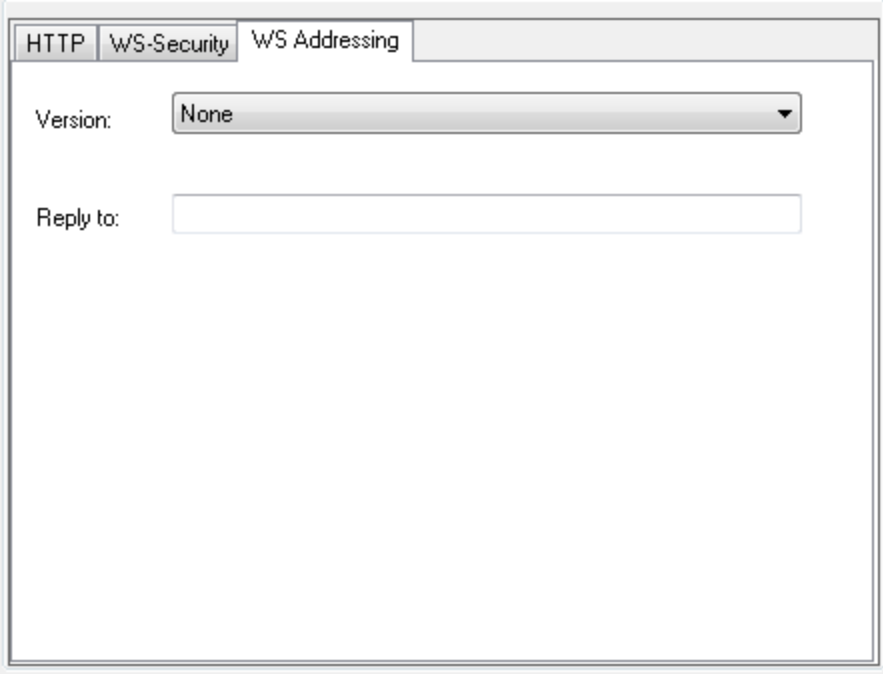
UI Elements	Description
	**Security Tokens.** Enables you to add one of the following tokens: • **User Name**, • **X509** • **Kerberos**, • **Kerberos2** • **SAML**
	**Add Message Signature.** Adds a signature to the message. This requires a token.
	**Add Message Encryption.** Adds encryption to the 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. **Note:** Make sure the security elements are positioned in order of their priority.
<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. • **What to encrypt** ▪ **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.
<security element list>	A list of the tokens, message signatures, and encryptions.

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.
<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> - 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" on page 579. • **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) select `BinarySecurity Token`.
<Token details pane> - SAML tokens	Token details for **SAML** tokens: • **<SAML token assertions>.** ▪ **Grid** view**,** an expandable node listing the assertions in the SAML schema. If you expand the list, you can set the attribute values. A drop down list lets you select a specific assertion. ▪ **Text** view, an XML reference to the token. ▪ **Revert.** Discards all changes made in Text view. • **Load from file.** Enables you to browse to a SAML certificate. • **Certificate.** The path of the certificate file. The **Browse** button opens the "Select Certificate Dialog Box" on page 579. • **Certificate reference type.** How the certificate should be referenced— by `X509 Data` or `RSA`.
Exclude Timestamp	Removes the timestamp from the SOAP header before sending the security element to the server.

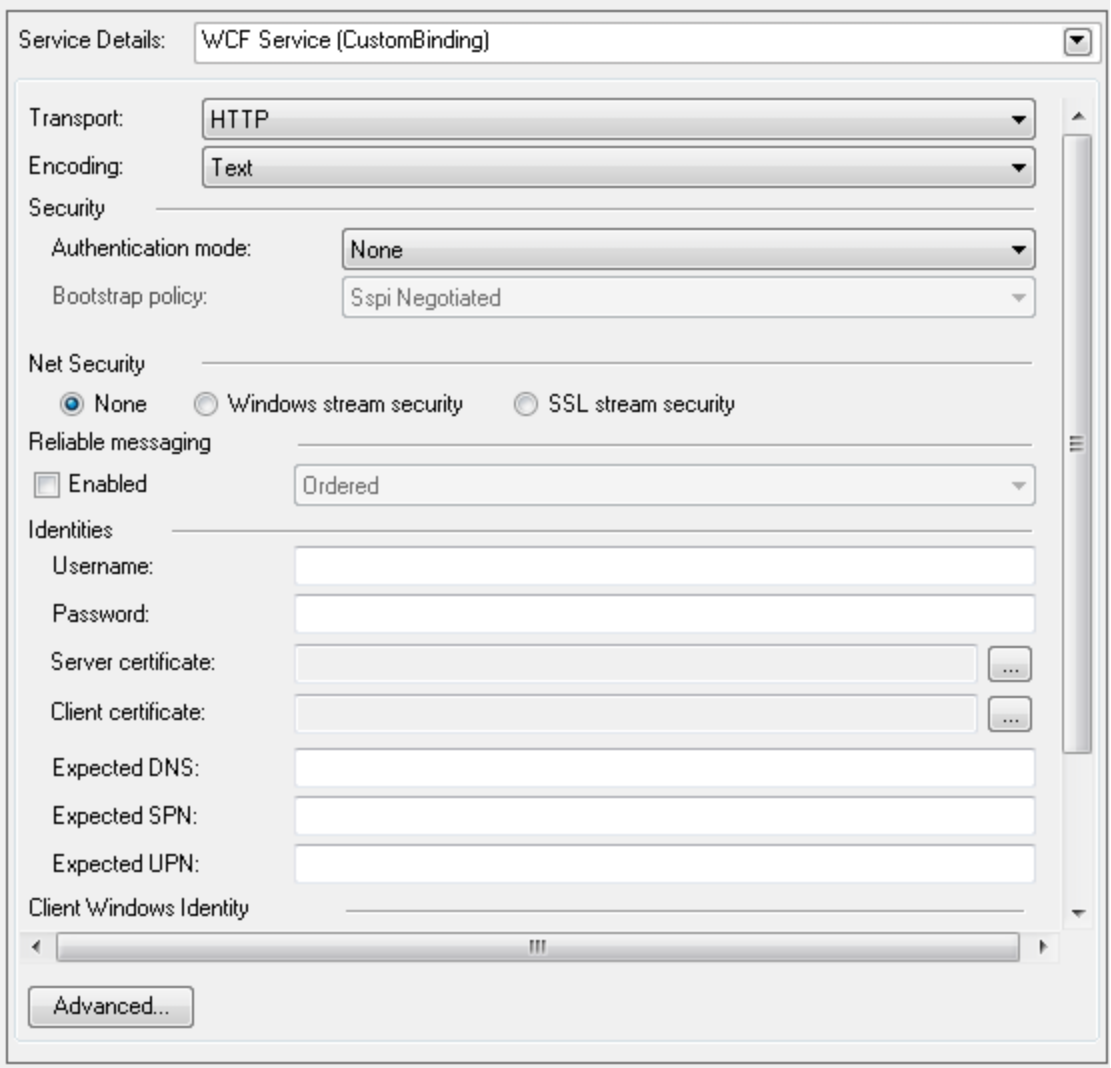
WS-Addressing tab

The **WS-Addressing** tab indicates whether WS-Addressing is used by the service, and if so, its version number. You can also specify the IP address of the server to which you want the response to be sent.



WCF Service (Custom Binding) Scenario (Security Settings for Port <Port_Name> Dialog Box

Use this scenario to test WCF services which require security or transport configurations. For general details, see "WCF Service (CustomBinding) Scenario Overview" on page 544.



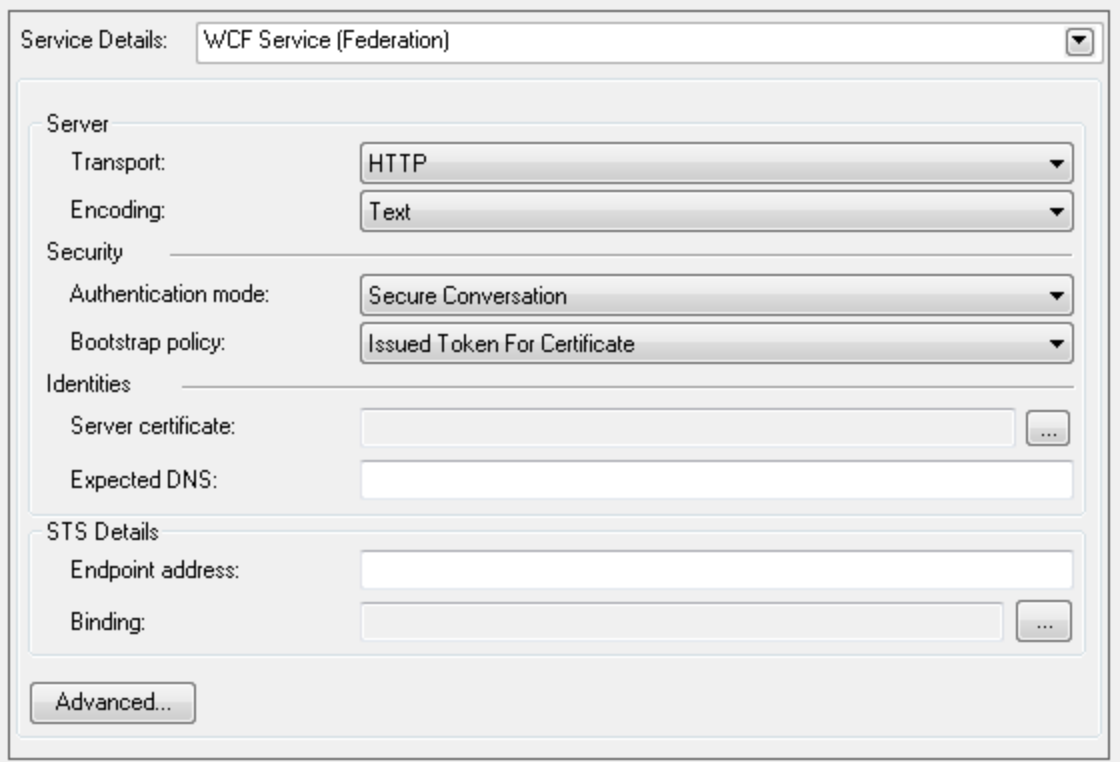
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Transport	**Transport type.** The transport types include: • `HTTP` • `HTTPS` • `TCP` • `NamedPipe` • `AutoSecuredHTTP`

UI Elements	Description
Encoding	**Encoding.** You can choose from the following encoding types: • `Text` • `MTOM` • `WCF Binary`
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 573.

WCF Service (Federation) Scenario (Security Settings for Port <Port_Name> Dialog Box)

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.



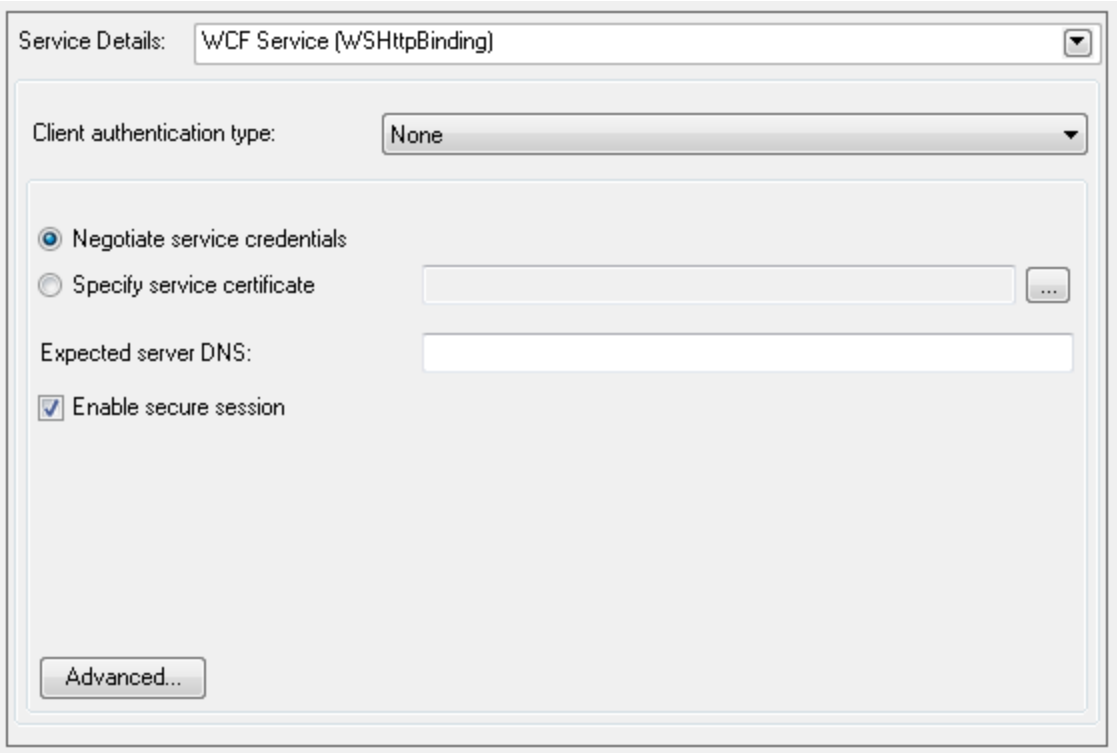
User interface elements are described below (unlabeled elements are shown in angle brackets). For details, see "WCF Service (Federation) Scenario Overview " on page 544.

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.

UI Elements	Description
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 573.

WCF Service (WSHttpBinding) Scenario (Security Settings for Port <Port_Name>) Dialog Box)

In the **WCFService (WSHttpBinding)** scenario, you can select from several types of authentication: None, Windows, Certificate, or Username (message protection). For details, see "WCF Services (WSHttpBinding) Scenario Overview " on page 545.



User interface elements are described below (unlabeled elements are shown in angle brackets) by the client authentication types:

UI Elements	Description
Client authentication type	Authentication type: • **None**, • **Windows** • **Certificate** • **Username (message protection)** For details, see below.
Advanced	Opens the Advanced Settings dialog box. For details, see "Advanced Settings Dialog Box" on page 573.

Client Authentication Types:

Client Authentication Type — None

UI Elements	Description
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.
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 details, see the "Select Certificate Dialog Box" on page 579.
Enable secure session	Enables a secure session using no client authentication.

Client Authentication Type — Windows

UI Elements	Description
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	Enables a secure session using `Windows` type authentication.
Expected server identity	The expected server identity method: `SPN` or `UPN`.

Client Authentication Type — Certificate

UI Elements	Description
Client certificate	The location of the client certificate. The **Browse** button opens the "Select Certificate Dialog Box".
Enable secure session	Enables a secure session using `Certificate` type authentication.
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.
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 details, see the "Select Certificate Dialog Box" on page 579.

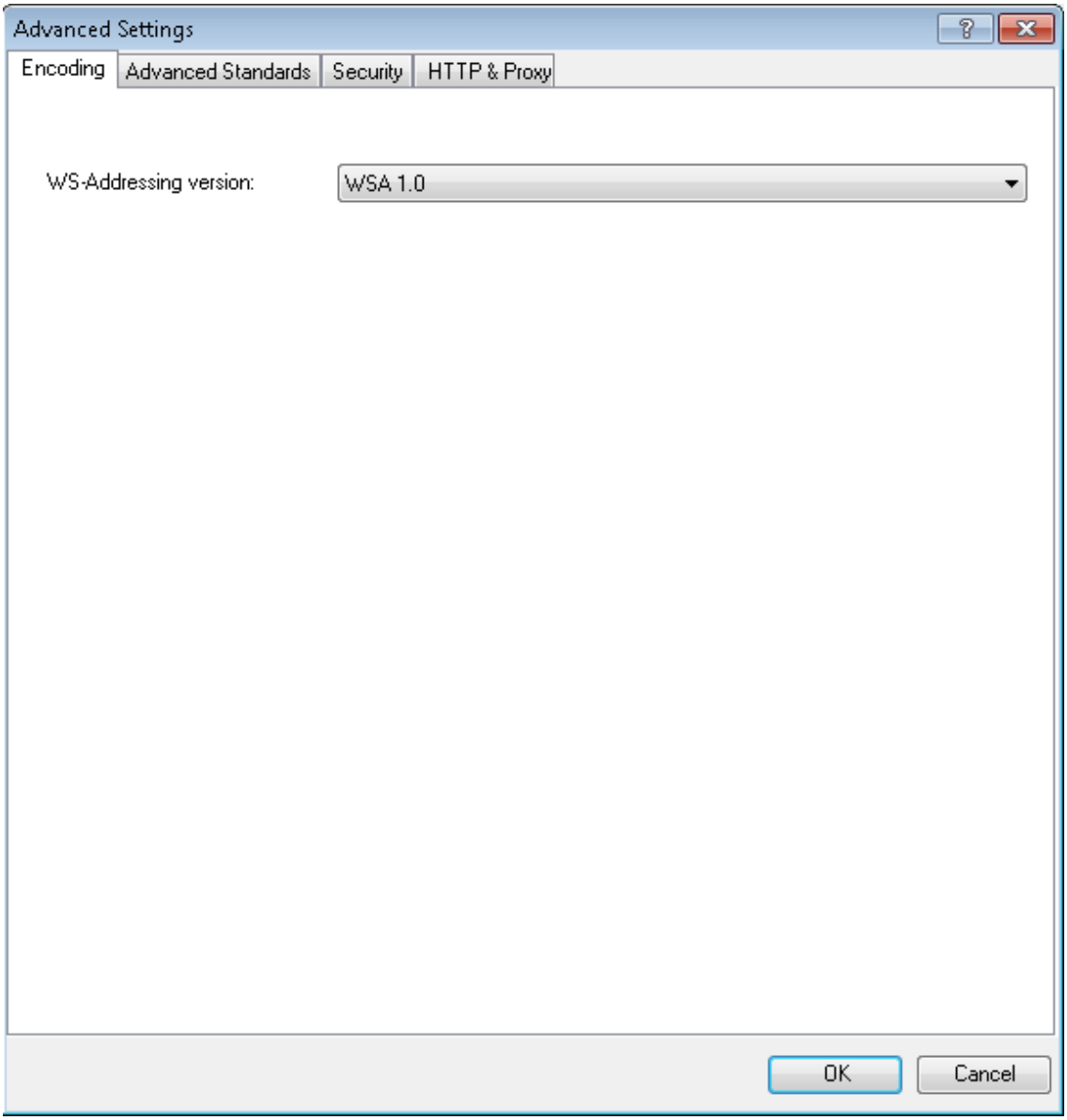
Client Authentication Type — Username (message protection)

UI Elements	Description
Enable secure session	Enables a secure session using `Username` type authentication.

UI Elements	Description
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.
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 details, see the "Select Certificate Dialog Box" on page 579.
Username, Password	The authentication credentials of the client.

Advanced Settings Dialog Box

This dialog box enables 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 546.
Relevant tasks	• "How to Set Security for a Web Service on the Port Level" on page 547 • "How to Set Security for a Specific Step" on page 547 • "How to Set Security for a WCF Service" on page 550

This section includes:

Encoding Tab (Advanced Settings Dialog Box)

The Encoding tab lets you indicate the type of encoding to use for the messages: **Text**, **MTOM**, or **WCFBinary**. The default is **Text** encoding.

The user interface elements are described below:

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 (Advanced Settings Dialog Box)

This tab lets you configure advanced WS- standards, such as Reliable Messaging and the Via address option.

The user interface elements are described below:

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.
Reliable messaging version	The version to apply to the messages: `WSReliableMessagingFebruary2005` or `WSReliableMessaging11`.

UI Elements	Description
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 (Advanced Settings Dialog Box)

The Advanced security settings correspond to the **WS-Security** specifications.

WCF Service (WSHttpBinding) Scenarios

UI Element (A-Z)	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 (A-Z)	Description
Allow serialized signing token on reply	Enables the reply to send a serialized signing token.
Default algorithm suite	The algorithm to use for symmetric/asymmetric encryption. The algorithm drop down list gets its values from the **SecurityAlgorithmSuite** configuration in WCF. **Default:** `Basic256`
Include timestamp	Includes a timestamp in the header.

UI Element (A-Z)	Description
Key entropy mode	The entropy mode for the security key. The possible values are: `Client Entropy`, `Security Entropy`, and `Combined Entropy`. **Default:** `Combined Entropy`
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.
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`
Require derived keys	Indicates whether to require derived keys.
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.
Require signature confirmation	Instructs the server to send a signature confirmation in the response.
Security header layout	The layout for the message header: `Strict`, `Lax`, `Lax Timestamp First`, or `Lax Timestamp Last`.
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.

UI Element (A-Z)	Description
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`
X509 reference style	How to reference the certificate: • `Internal` • `External`
X509 require derived keys	Indicates whether X.509 certificates should require derived keys.

HTTP and Proxy Tab (Advanced Settings Dialog Box)

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 (A-Z)	Description
Allow cookies	Indicates whether to enable or disable cookies.
Authentication scheme	The HTTP authentication method: `None`, `Digest`, `Negotiate`, `NTLM`, `Integrated Windows Authentication`, `Basic`, or `Anonymous`.
Bypass proxy on local	Indicates whether to ignore the proxy when the service is on the local machine.
Keep-Alive enabled	Indicates whether to enable or disable keep-alive connections.
Max response size (KB)	The maximum size of the response before being concatenated. **Default:** 65 KB
Proxy address	The URL of the proxy server.

UI Element (A-Z)	Description
Proxy authentication scheme	HTTP authentication method on Proxy: `Digest`, `Negotiate`, `NTLM`, `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.
Transfer mode	The transfer method for requests/responses. The possible values are `Buffered`, `Streamed`, `Streamed Request`, and `Streamed Response`.
Use default web proxy	Indicates whether to use machine's default proxy settings.

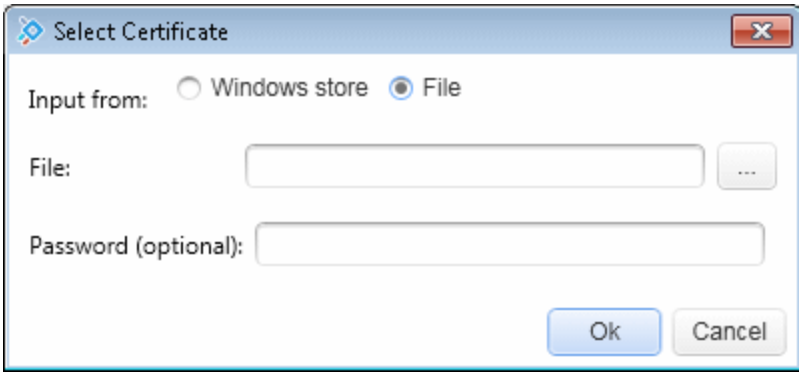
Select Certificate Dialog Box

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 550

Select Certificate from File

When you select **Import from: File**, the dialog box shows the relevant user elements.

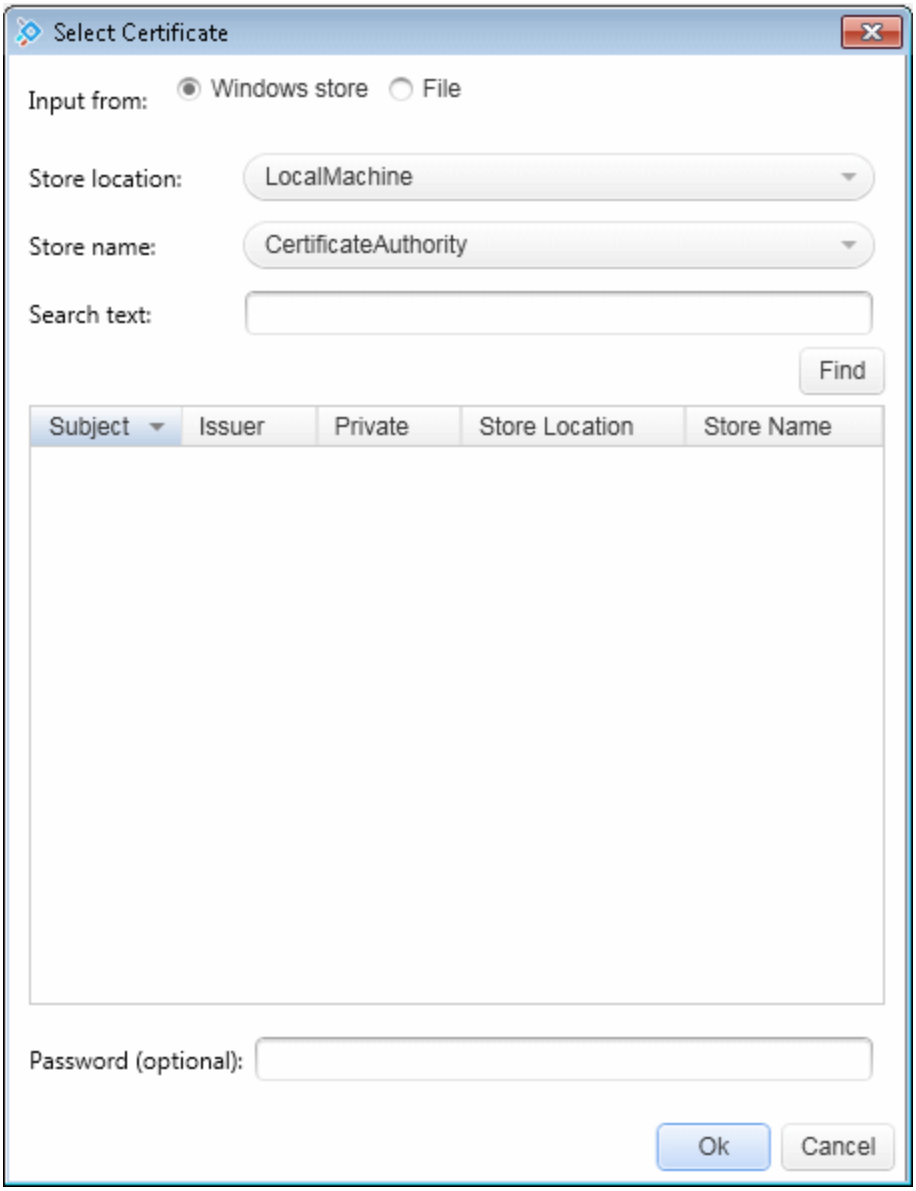


The user interface elements are described below:

UI Elements	Description
Import from	The source of the certificate file: **Windows store** or **File**.
...	**Browse.** Enables 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 select **Import from > Windows store**, the dialog box shows the Windows Store related user elements:



The user interface elements are described below:

UI Elements	Description
Import from	The source of the certificate file: **Windows store** or **File**.
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**.
Find	Begins the search for the certificate based on the Search text.
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.
- For Web Service configured with WCF settings: Configuring different security settings for operations residing on the same port is not supported.
- For Web Service configured with WCF settings, some of the user event handlers (such as the **AfterProcessRequestSecurity**, **BeforeProcessResponseSecurity,OnSendRequest,** and **OnReceiveResponse** events) will not be invoked.
- 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.
- When working with Federation type scenarios that use STS (Security Token Service), you cannot change the SOAP version.
- If you are using SOAP version 1.2:
  - You can choose only UserName or X509 tokens when configuring the message level security.
  - When configuring the canonicalization algorithm and the transform algorithm for the message signature cannot be http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform.

Chapter 28: Testing Techniques

This chapter includes:

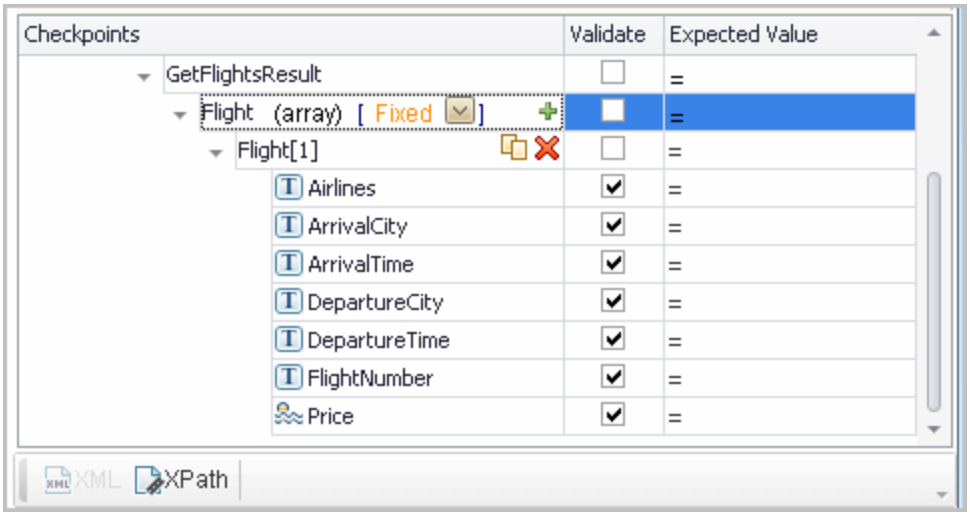
For general information about running a test, see "Running Tests and Components" on page 284.

Concepts

Checkpoint Validation

In functional testing, it is necessary to check the results of test steps to confirm that the activity performed its expected functionality. The response can contain several properties, each containing several data items. The **Checkpoints** section is a central point for defining the expected values of the properties.

Before replaying a test, you set the expected output property values. You can enter the values manually, link them to a data source, or load values that were captured during a prior replay. This is useful when you have many argument values—instead of manually entering values, you automatically load them.



To include a checkpoint in the test, you select its check box. If you loaded replay values, you can select only those properties that you loaded.

For WSDL-based Web Services and SOAP Requests, Service Test includes two built-in checkpoints for the purpose of validation. One checkpoint validates the XML structure and the other checks its compliance with WS-I.

Additional checkpoint settings let you trim the string, ignore case inconsistencies, and indicate whether to stop on failed checkpoints.

For details, see the "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.

After the test run, the Run Results Viewer displays the checkpoint status in its own sub-node. For details, see the *HP Run Results Viewer User Guide*.

This section also includes

XPath Checkpoints

For steps with XML output properties, such as **Web Service** and **SOAP Request**, **String to XML**, and so forth, you can validate the test results against XPath expressions.

You can specify a fully qualified XPath expression, or you can instruct Service Test to ignore the namespaces and prefixes during the test run. By ignoring the namespaces, you can use a simpler expression.

In the following example, to retrieve the contents of the second node, `B`, you would need to write an expression that also indicates the namespace, such as `//*[local-name(.)='Node' and namespace-uri(.)='ns2']`.

```
<Root>
   <Body>
        <Node xmlns="ns1">A </Node>
        <Node xmlns="ns2">B </Node>
   </Body>
</Root>
```

When working with simple XPath expressions, you can further simplify the XPath expression by selecting **Ignore namespaces**. In the above example, the expression `//Node[2]` is sufficient to evaluate the value `B` in the second node.

You can type in the XPath expressions manually, or use the shortcut menu to retrieve the simplified or fully qualified XPath.

Service Test also enables you to evaluate XML with namespace prefixes. For example, if the XML contains the prefix definition `xmlns:T="ns1"`, you can specify the prefix in the XPath expression: `//T:`*NodeName*. To evaluate namespace prefixes, disable the **Ignore namespaces** option.

XPath checkpoints can only be used when the XPath query returns a scalar value—not XML.

For task details, see "How to Set XPath Checkpoints" on page 592.

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.

The Properties pane 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 Tab (Properties Pane)" on page 209.

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 your machine, Service Test provides a custom template, **API Load Test**.

For tests created with the standard **API 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 **EndTransaction** 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 633.

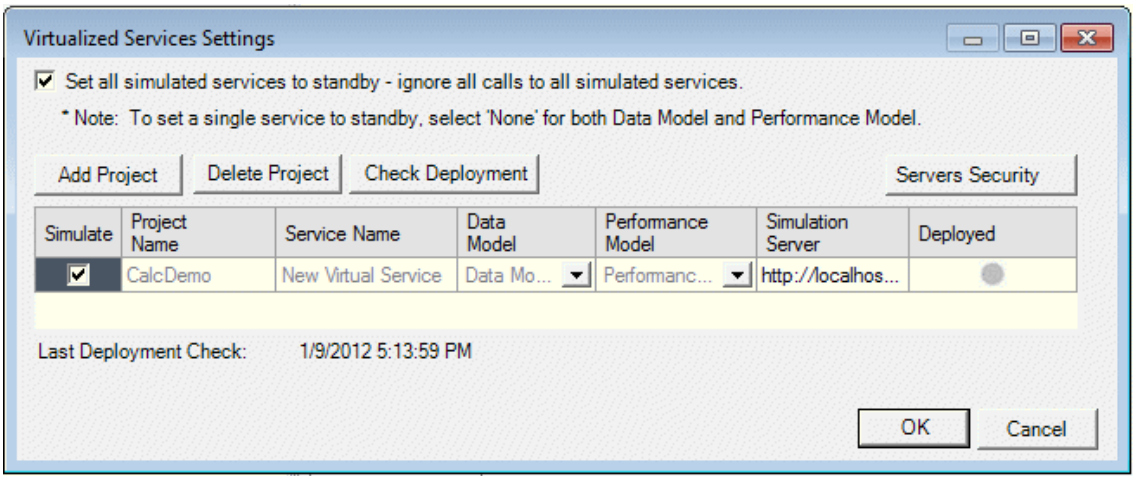
For task details, see "How to Prepare and Run a Load Test" on page 594.

Using Virtualized Services

Service Test integrates with HP Service Virtualization. This integration enables you to run virtual services that simulate actual services. This is especially useful where running a service incurs a cost, such as credit card processing. In addition, it is useful when development is incomplete for the service you need to test.

In HP Service Virtualization, you create simulation projects. These projects contain information about the virtualized services that simulate actual services.

In Service Test, you can load one or more simulation projects. A dialog box indicates if and when the service was deployed, and information about the virtualization server. For details, see the "Virtualized Services Settings Dialog Box" on page 598.



Once you deploy the project, you can import its services the same way you would import any other WSDL. For details, see "Import/Update WSDL from URL or UDDI Dialog Box" on page 489.

Data and Performance Models

You can select the Data and/or Performance models to use with the virtualized service in test execution.

The Data Model enables you to record actual requests and responses for the real service and then use this data for simulation using the virtual service. You can edit the data, add service calls, and model the behavior.

The Performance Model enables you to record the performance for the real service and then use this as performance criteria for the service.

You define these models when you create the Simulation project in HP Service Virtualization. You should select at least one Data or Performance model. If you select `None` for both models, it is equivalent to working in Pass Through mode.

For details, refer to the *HP Service Virtualization User Guide*.

Tasks

How to Set Array Checkpoints

This task describes how to manually set checkpoints for elements of an array. For details about setting regular checkpoints, see "How to Run an API Test" on page 288.

For details about data driving array checkpoints, see "How to Data Drive Array Checkpoints" on page 591.

This task includes the following steps:

1. ### Enable active content on your computer

   The Run Results Viewer shows array checkpoint results in an expandable tree. To enable this view, modify your browser settings as follows:

   a. In Internet Explorer, select **Tools > Options**.

   b. Select the **Advanced** tab.

   c. Enable the option **Allow active content to run in files on My Computer** in the **Security** section.

   d. Click **OK** and close the browser.

2. ### Add a step with an array output

   Add a test step with output properties in the form of an array.

3. ### Select an array validation method

   Expand the drop down adjacent to the name of the parent array node. Select one of the following:

   - **Fixed.** Checks that each of the returned array elements matches its corresponding array element in the **Checkpoints** pane. Each array is marked by an index number, as it checks the arrays by their index.
   - **All.** Checks that all of the returned array elements match the array element in the **Checkpoints** pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked **>= 2** and the same property in another array element is

set to **<=10**, the test run will check that all returned values are between 2 and 10.

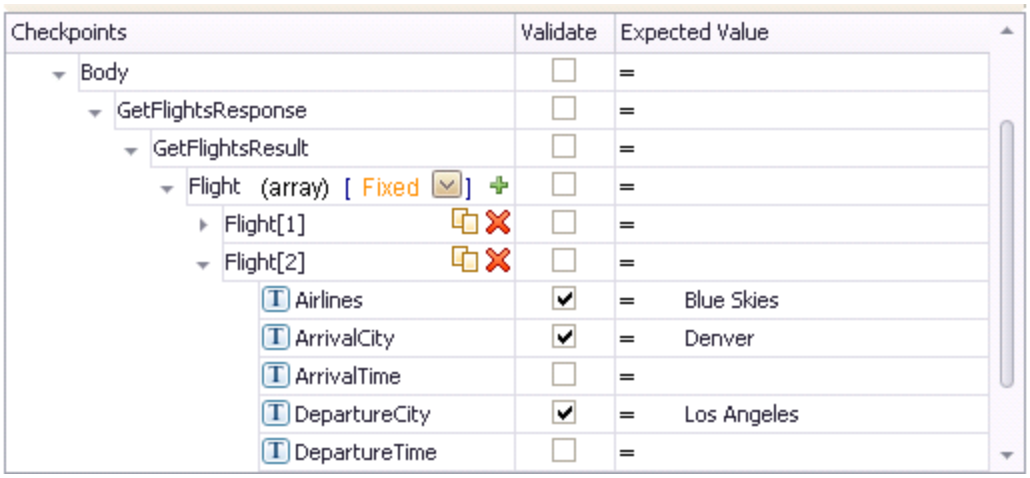
- **Contains.** Checks that at least one of the returned array elements matches the value of the property in the **Checkpoints** pane. In this mode, arrays are not marked by an index number.

4. ## Add array elements

   Use the plus button ✚ in the row of the parent node to add the desired number of array elements.

5. ## Provide values

   Select the **Validate** box for each value that you want to validate and provide values for those properties.



6. ## Validate the element count - optional

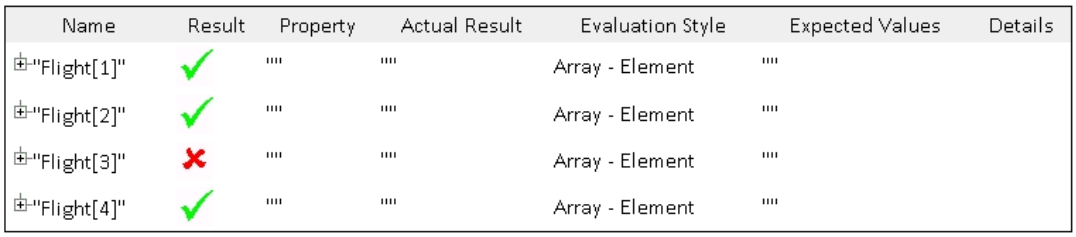
   Click in the parent row of the array and enter a desired count number in the **Expected Value** column and the evaluation expression, such as **=**, **>**, and so forth. During the test run, the validation will check the number of returned array elements against this value.

7. ## Set the number of iterations and run the test

   Click in the **Test Flow** or **Loop** box and open the Properties pane. Set the test flow properties, such as the number of iterations or loop type. Click the **Run** button and provide a results location. For details, see "How to Run an API Test" on page 288.

8. ## Open the Checkpoint report

   In the Run Result Viewer, expand the test results tree in the left pane. Select a Checkpoint node. In the **Captured Data** pane, click **View Report** in the Details column. In the report that opens, expand the checkpoint node for each of the iterations to view its actual and expected value.



How to Data Drive Array Checkpoints

This task describes how to set checkpoints for elements of an array through data driving. For details about setting regular checkpoints, see "How to Run an API Test" on page 288.

This task includes the following steps:

1. ### Enable active content on your computer

   The Run Results Viewer shows array checkpoint results in an expandable tree. To enable this view, configure your browser to allow active content as described in "How to Set Array Checkpoints" on page 589.

2. ### Add a step with an array output

   Add a test step with output properties in the form of an array.

3. ### Add an array element

   a. Open the Properties pane and click in the **Checkpoints** area.

   b. Use the plus button . in the row of the parent node, to add an array element.

   c. Provide property values. These values are transferred to the Data pane during data driving. If you do not provide data, include the array element by selecting the triangle icon in the parent node.

4. ### Data Drive the array

   a. Select the array's parent node and click the **Data Drive** button .

   b. In the "Data Driving Dialog Box" (described on page 625), select **Only Input, Checkpoints** or **Both Input and Checkpoints** as a **Data Driven Section** option.

   c. Click **OK**. The data driving mechanism informs you that it succeeded.

5. ### Set the evaluation expression

   In the Properties pane, adjacent to the data driving expression, select an evaluation operator,

such as **=**, **<**, and so forth.

6. ### Provide data for the array

   a. Once you data drive an array, you do not set property values from the **Checkpoints** pane. Instead, you edit the table in the Data pane. In the Data pane, which opens automatically, click the node in the left pane, corresponding to the array element that is to be validated.

   b. Enter the iteration number to which to checkpoint should be applied in the `MainDetails` column of the **<array_name>** and `MainDetails` tables. A "1" indicates the first iteration. If you have several columns with "1" as the iteration number, the checkpoints will be validated against all of those values.

7. ### Select an array validation method

   Select the step in the canvas and view the Properties pane. Expand the drop down adjacent to the name of the parent array node. Select one of the following:

   - **Fixed.** Checks that each of the returned array elements matches the corresponding array element in the data table in the Data pane. Each array is marked by an index number, as it checks the arrays by their index.
   - **All.** Checks that all of the returned array elements match the array element in the Data pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked >= 2 and the same property in another array element is set to <=10, the test run will check that all returned values are between 2 and 10.
   - **Contains.** Checks that at least one of the returned array elements matches the value of the property in the **Checkpoints** pane. In this mode, arrays are not marked by an index number.

8. ### Set the number of iterations - optional

   If you selected the **Configure Test Flow as a ForEach loop using the new data source option** in the "Data Driving Dialog Box" (described on page 625), you can skip this step. If not, click in the **Test Flow** or **Loop** box and open the Properties pane. Set the test flow properties, such as the number of iterations or loop type.

9. ### Run the test

   Click the **Run** button and provide a results location. For details, see "How to Run an API Test" on page 288.

10. ### Open the Checkpoint report

    In the Run Results Viewer, expand the test results tree in the left pane. Select a Checkpoint node. In the **Captured Data** pane, click **View Report** in the Details column. In the report that opens, expand the checkpoint node for each of the iterations to view its actual and expected value.

How to Set XPath Checkpoints

This task describes how to validate your test results against XPath expressions. For conceptual information, see "XPath Checkpoints" on page 586.

This task includes the following steps:

1. ## Add a step with XML output

   Add a test step with XML output, such as **String to XML**. Open the **Input/Checkpoints** tab . Paste a source string into the **Value** column or use the **Link to data source** button to link to a value.

2. ## Set the namespace setting

   Click the **XPath** tab, and indicate whether or not to ignore namespaces. By default, the **Ignore namespaces** option is enabled. If you plan to validate against a fully qualified expression, or if you need to specify a namespace prefix, disable the option. For details, see "Properties Pane Tabs" on page 186.

3. ## Copy the XPath

   Copy an XPath expression to the clipboard. If you intend to enter the XPath expression manually, skip this step.

   - To retrieve a simple XPath, click in the **Value** column, and select **Copy XPath** from the shortcut menu.
   - To retrieve a complete qualified XPath, click in the **Value** column, and select **Copy Fully Qualified XPath** from the shortcut menu.

4. ## Add an XPath checkpoint

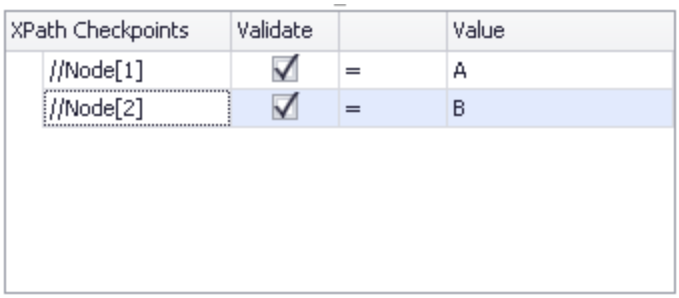
   Click the **XPath** tab in the Checkpoints section. Click the **Add** button to add a new XPath checkpoint.

5. ## Provide the XPath expression

   Type or paste the expression into the **XPath Checkpoints** column. You can also use the **Link to data source** button to link to a value.

6. ## Set up the validation

   Select the check box in the **Validate** column, select a comparison operator, and provide an expected value.



7. **Run the test**

   Run the test and check the 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 587.

This task includes the following steps:

- "Prerequisite" below
- "Create a load test" below
- "Add test steps" below
- "Prepare for load testing - optional" below
- "Assign data to the test - optional" on next page
- "Set the data retrieval properties - optional" on next page
- "Set the run configuration to Release" on next page
- "Run in Load Testing mode to validate the test" on next page
- "Incorporate the test into LoadRunner" on page 596

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**

   In the "New <Document> Dialog Box" on page 67 (described on page 67), in the **Select type** section, choose **API Load Test**.

   If you have a test that was created with the standard Service Test template, select **Design > Operation > Enable Test for Load Testing** or click the **Enable Test for Load Testing** button .

3. **Add test steps**

   Drag activities from the Toolbox pane onto 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 pane's **Load Testing** category, onto the canvas. Place it before the first step of the group of steps that you want to measure.
- Click the Input/Checkpoints tab in the Properties pane. 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 Properties pane's Input/Checkpoints tab, 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 Properties pane's Input/Checkpoints tab, 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 621.

6. ## Set the data retrieval properties - optional

If you have data in the Data 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 633.

7. ## Set the run configuration to Release

In the General pane of the API Testing tab in the Options dialog (**Tools > Options > API Testing** tab **> General** node), in the **Run Sessions** options, select **Release**. The **Release** mode conserves resources, thus enhancing the load testing capabilities.

8. ## Run in Load Testing mode to validate the test

Expand the toolbar **Run** button and select **Run Test in 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 pane does not contain data and the Run Results Viewer does not open. To view the results, select **Run** 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.

Reference

Command Line Syntax

The Service Test executable, `ServiceTestExecuter.exe`, is located in the product's `bin` directory.

The following table describes the command line options for `ServiceTestExecuter.exe`:

UI Elements (A-Z)	Description
-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 59.
-report	The directory in which to store the report.
-test	The full path of the test (required). Specify the test directory—not the solution directory.

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`:

```
%ProgramFiles%\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>
```

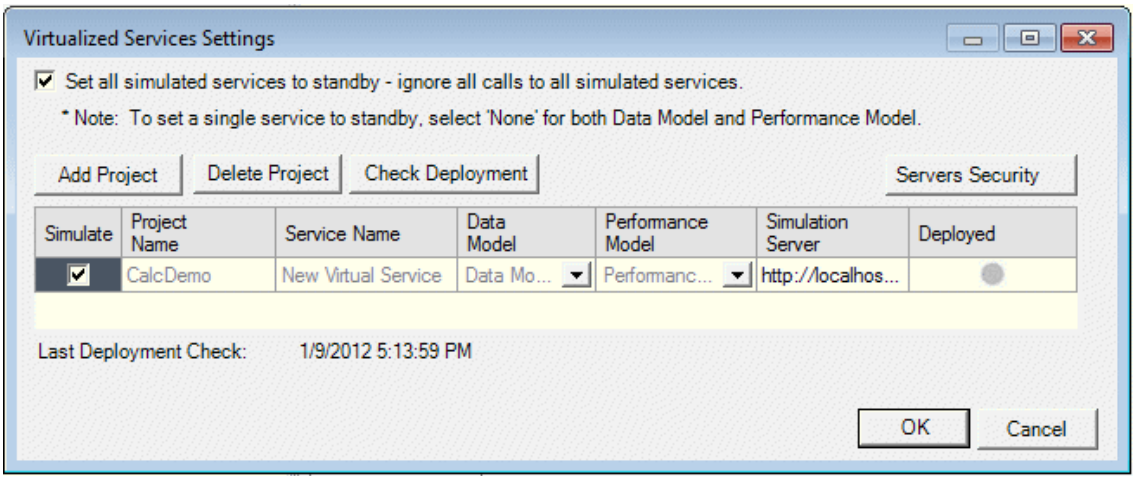
Note: To run a test from the command line, you must save and run the test at least once.

Virtualized Services Settings Dialog Box

The Virtualized Service user interface provides integration with HP Service Virtualization. This integration enables you to test services with virtualized servers instead of actual servers.

For details, see "Using Virtualized Services" on page 587.

This dialog box enables you load HP Service Virtualization projects.



To access	1. Create or open a test or component. 2. Click the **Virtualized Services Settings** toolbar button .
Important information	You must stop the service simulation and close the Service Virtualization Designer window, before running the test that uses the virtualized service. Failure to do so will result in the service being locked for all other users.
Relevant tasks	"How to Run an API Test" on page 288

The user elements are described below (unlabeled UI elements are shown in angle brackets).

UI Element (A-Z)	Description
<emulated services>	A list of the virtualized services showing the following information: • **Simulate.** Includes the virtual service when executing the test. • **Project Name.** The name of the HP Service Virtualization project containing the virtual service. • **Service Name.** The name of the virtual service to use during test execution. • **Data Model.** The data model to associate with the virtual service. • **Performance Model.** The performance model to associate with the virtual service. • **Simulation Server.** The machine used for emulating the virtual service. If the value was configured on the server machine as `localhost`, be sure to change it to the actual server name. • **Deployed.** An indicator showing the deployment status: ▪ **Unknown.** Click the **Check Deployment** button to check the status. ▪ **Deployed.** Successfully deployed. ▪ **Not Deployed.** The service is not deployed on the specified emulation server.
Add Project	Opens the Add Project dialog box, allowing you to specify or browse for an HP Service Virtualization project. Project files have a `.vproj` extension.
Check Deployment	Checks if the virtual service was deployed on the virtualization server.
Delete Project	Removes the selected project(s) from the list of virtualized services.
Servers Security	Opens the Servers Security dialog box showing the credentials for each of the servers.
Set all simulated services to standby - ignore all calls to all simulated services.	Allows you to temporarily ignore all calls to simulated services. To set a specific virtualized service to standby, select `None` for its Performance and Data models.

Troubleshooting and Limitations - Load Testing

This section describes troubleshooting and limitations for working with load testing.

- Related data mapping in a load-enabled test is not supported for the LoadRunner parameter advance policy of **Each Occurrence**.
- 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.
- You cannot run tests containing actions or calls to other tests on a remote load generator. This limitation does not apply when running the test on a local load generator.
- The data assignment method, **Use a unique value for each Virtual User when load testing** is not supported in all environments.

Chapter 29: Asynchronous Service Calls

This chapter includes:

Concepts

Asynchronous Services

You can use Service Test to emulate asynchronous services. Asynchronous services can be Web Services, REST services, HTTP requests, JMS/MQ-based services, and so forth.

In synchronous messaging, the replay engine blocks step execution until the server responds. The client sends a request and receives a response immediately, using the same connection. During the waiting time, the replay engine is blocked and does not perform any other activity. If the timeout was reached without a response from the server, the client returns an error.

In asynchronous mode, the replay engine executes the step without waiting for server's response from previous requests.

Wait Steps

When sending asynchronous calls with HTTP or HTTPS, you use a **Wait** step to instruct the test to wait for the response of earlier asynchronous requests before continuing with its execution.

You do not have to place the **Wait** step directly after the Receive step. The test can proceed with other steps, but the **Wait** step will instruct the test to wait for a response before ending the test, or before continuing to execute any steps that follow the **Wait** step.

Checkpoints

When sending a request to a server, you use checkpoints to verify the **Output** property values.

When getting a request from a server, as in most of the asynchronous patterns, you use checkpoints to verify the **Input** properties.

Service Test provides a solution for the following asynchronous patterns:

- "WS-Addressing" below
- "HTTP Receiver" on next page
- "Web Service Publish Subscribe" on next page
- "Web Service Solicit Response" on next page
- "Dual WSDL Files" on page 604

WS-Addressing

WS-Addressing is a specification that enables Web services to communicate addressing information. You can instruct the server to respond to any location, and not necessarily to the machine that issued the request. To do this, you use the WS-Addressing **replyTo** attribute.

In this implementation, Service Test pauses the test and uses a listener mechanism to verify that the response arrived at the specified address. After the listener acknowledges that the server responded to the address or if it reaches the timeout, the test resumes. Upon the completion of the test, you can validate the response with the standard Service Test checkpoints.

For user interface details, see the "Asynchronous Tab (Properties Pane)" on page 186.

HTTP Receiver

In the **HTTP Receiver** pattern, the **server** sends an HTTP request to the client, reversing the typical roles of the client and server.

This is useful, for example, if you want to test a service which publishes information over HTTP to a client. You define a receiver, which waits for a request from the server, sent over HTTP.

After a trigger, the receiver captures the request. The trigger can be an HTTP client request, a call to a Web service, an email, or any other event that will trigger the server. If there are inner steps, the receiver waits for them to finish and only then is the receiver activity considered complete.

Using the Service Test interface, you can insert the necessary logic and validate the checkpoints in the captured request.

The response from the receiver should wait for the inner steps to complete and link to them.

The completion event name fired for the receiver should only be fired AFTER the inner steps are done.

For details about the properties, see the "Network Activities" on page 426.

Web Service Publish Subscribe

In the **Web Service Publish Subscribe** pattern, the **server** sends an HTTP request to the client, reversing the typical roles of the client and server. it is similar to the HTTP Receiver, except that the request is sent to the client through a Web Service call instead of exclusively via HTTP.

Using Service Test, you test the publishing of messages to the client. You set up a receiver, which waits for a server request, sent from the server as a Web service call.

After a trigger, the receiver captures the request. The trigger can be an HTTP client request, a call to a Web service, an email, or any other event that will trigger the server.

Using the Service Test interface, you can validate the response with standard Service Test checkpoints.

Web Service Solicit Response

The **Web Service Solicit Response** pattern is a variation of the **Web Service Publish Subscribe** pattern. It enables you test a a service which publishes information through a Web Service to a client.

In this pattern, however, the client is expected to send a response to the server request. The response can be a simple acknowledgement or a full SOAP message.

You set up a receiver activity, which waits for a server request. This server request is sent from the server as a Web service call. The receiver then sends a client response back to the server.

After the trigger, the receiver captures the request. The trigger can be an HTTP client request, a call to a Web service, an email, or any other event that will trigger the server.

Using the Service Test interface, you can validate the response with standard Service Test checkpoints.

Dual WSDL Files

The Dual WSDL technique is a standard request-response pattern. In this pattern, however, the client request is defined by one WSDL, and the server response is defined by another WSDL.

You implement this scenario in two stages:

- Import the Request WSDL file, using the **Import WSDL from** import command.
- Import the Response WSDL file, using the **Import WSDL from** import command, enabling the **Import as Server Response** option. For details, see the "Import/Update WSDL from URL or UDDI Dialog Box" on page 489 or "Select WSDL Dialog Box" on page 488.

For task details, see "How to Create a Test for an Asynchronous Web Service" on next page.

Tasks

How to Create a Test for an Asynchronous Web Service

This task describes how to create a test for testing an asynchronous Web service.

For an overview of the asynchronous patterns supported by Service Test, see "Asynchronous Services" on page 602.

The following section describes the following tasks:

- "Create a test for WS-Addressing" below
- "Create a test for HTTP Receiver" below
- "Create a test for a Web service publish subscribe pattern" on next page
- "Create a test for Dual WSDL Files" on next page

Create a test for WS-Addressing

To use WS-Addressing for a Web Service call:

1. Import a Web Service using **Import WSDL > Import WSDL from URL or UDDI** and drag an operation onto the canvas. For details, see "How to Import a WSDL-Based Web Service" on page 472.
2. Open the Asynchronous tab  in the Properties pane and select **This is an asynchronous call** box.
3. Specify a value for the **Listen for response on** property. This is the port to which you expect the server to respond.
4. Open the Security tab  in the Properties pane. Select the **WS Addressing** tab.
5. Select a WS-Addressing **Version** and provide and a URL and port (same port as defined for the **Listen for response on** property) in the **Reply to** box, to indicate the destination of the server response.
6. Run the test and perform checkpoint validations as you would with any step.

For more details, see the "Asynchronous Tab (Properties Pane)" on page 186.

Create a test for HTTP Receiver

To create a test for an HTTP Receiver in which the client receives the response:

1. Drag a **Network > HTTP Receiver** step onto the canvas.
   a. Make sure you are logged in as an administrator. Administrator privileges are required to run **HTTP Receiver** steps.
   b. Specify the **General** properties  . For details, see the "Network Activities" on page 426.
   c. Set the **HTTP Receiver** properties  . For details, see the "HTTP Receiver Tab (Properties Pane)" on page 201.

d. Set a filter for the HTTP message . For details, see "Filter Settings Tab (Properties Pane)" on page 195.

2. Drag a **Flow Control > Wait** step onto the canvas. Specify timeout information and one or more completion events. You can link to the completion event from a prior **HTTP Receiver** step.
3. If required, drag additional activities from the Toolbox pane into the **HTTP Receiver** flow.
4. Run the test and apply the trigger for the server. For details, see the "HTTP Receiver" on page 603.

> **Tip:** If your test needs to listen to more than one message, receiver steps (such as **HTTP Receiver** or Web Service calls set up as receivers) can be data driven and placed inside a loop. The placement of the **Wait** step—inside or outside of the loop—depends on whether the send order matters:
>
> - If the messages to be sent to the receiver are expected in a specific order, you must place the **Wait** step inside the receiver step's frame. All steps that are contained within the receiver can be data driven using this loop.
> - If however, the messages are expected in a random order, place the **Wait** step outside the receiver step. Steps that are contained within the receiver should not be data driven using the same loop as the receiver step and should not link to other steps outside the receiver.

Create a test for a Web service publish subscribe pattern

To create a test to check that messages are properly published to the client:

1. Import a WSDL as a server by enabling the **Import as server** option. For details, see "How to Import a WSDL-Based Web Service" on page 472. This Web Service should have a receiver type pattern. Drag an operation onto the canvas.
2. Set the input or output properties . For details, see the "Input/Checkpoints/Output Properties Tab (Properties Pane)" on page 201.
3. Drag a **Flow Control > Wait** step onto the canvas. Specify timeout information and one or more completion events. You can link to the completion event from a prior **HTTP Receiver** step.
4. Run the test and activate the trigger for the server. For details, see "Web Service Publish Subscribe" on page 603.

Create a test for Dual WSDL Files

To create a test for a dual WSDL pattern, using one WSDL for the request and another for the response:

1. Import the request WSDL file using the standard import command, **Import WSDL from**. The imported operations can send client requests and receive server responses.
2. Import the response WSDL file as a server by enabling the **Import as Server Response** option in the "Select WSDL Dialog Box" or the "Import/Update WSDL from URL or UDDI Dialog Box" (for URL and UDDI imports, click **Advanced Settings** to show the option). The imported operations first receive server requests and then send client responses.

3. Drag a Web service operation with the request onto the canvas.
4. Drag a Web service operation with the response onto the canvas, after the request step.

For details, see "Dual WSDL Files" on page 604.

Troubleshooting and Limitations - Asynchronous Testing

This section describes troubleshooting and limitations for creating asynchronous tests.

For a Web service imported as a server response:

- When enabling the SSL option, Service Test temporarily binds the SSL certificate to the specified port on a system, **http.sys**, level. If you end the `Service Test.exe` process from the Task Manager during the listening stage after the binding was added, the binding will not be automatically removed from the system.

  Workaround: Remove the binding manually using a utility such as `httpcfg.exe` or `netsh.exe`.

- When working with SSL, certificates from a file are not supported. If you move the test to another machine, the certificate will not be available.

  Workaround: Add the certificate to the local machine store before running the test. If desired, remove it after you finish working with the test.

Chapter 30: Data Handling

This chapter includes:

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 pane lets you view and maintain data for your test steps. You can import data from Excel files, XML files, or databases. You can also create local data tables, which you populate manually, as a source for your data. For details, see "How to Assign Data to Test Steps" on page 621. For details on the Data pane, see "Data Pane Overview" on page 89.

You can data-drive the properties of a step or the SOAP or HTTP request and response bodies. 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 625.

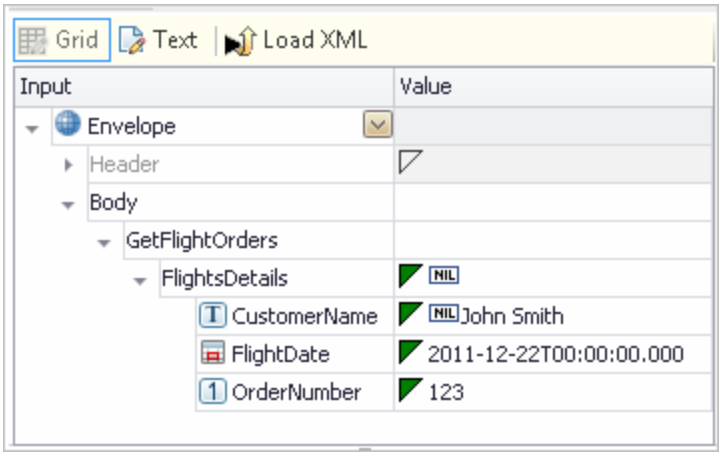
Defining Data

Service Test enables you to assign data to your steps in several ways. You can enter values manually, link to existing data, or data drive the values.

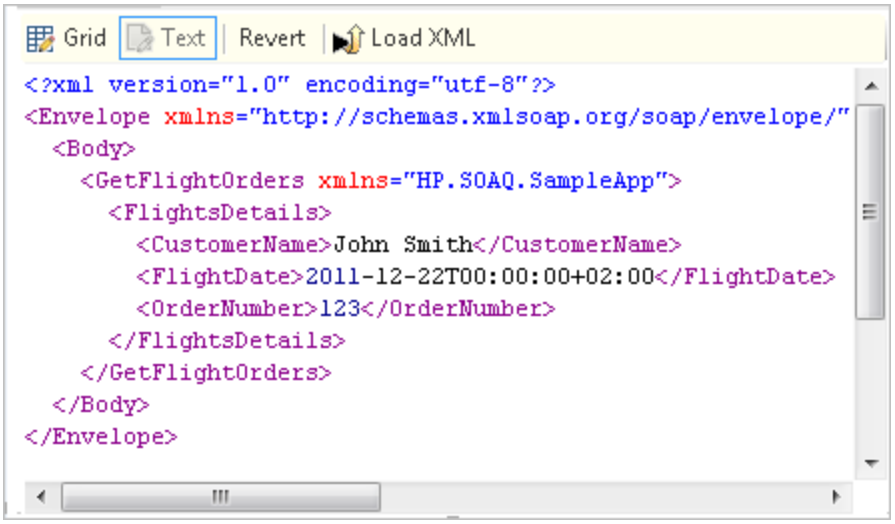
This section also includes:

Populating Data Tables Manually

The "Properties Pane Tabs" grid enables 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 enables you to discard changes and return to the original XML using the **Revert** button.



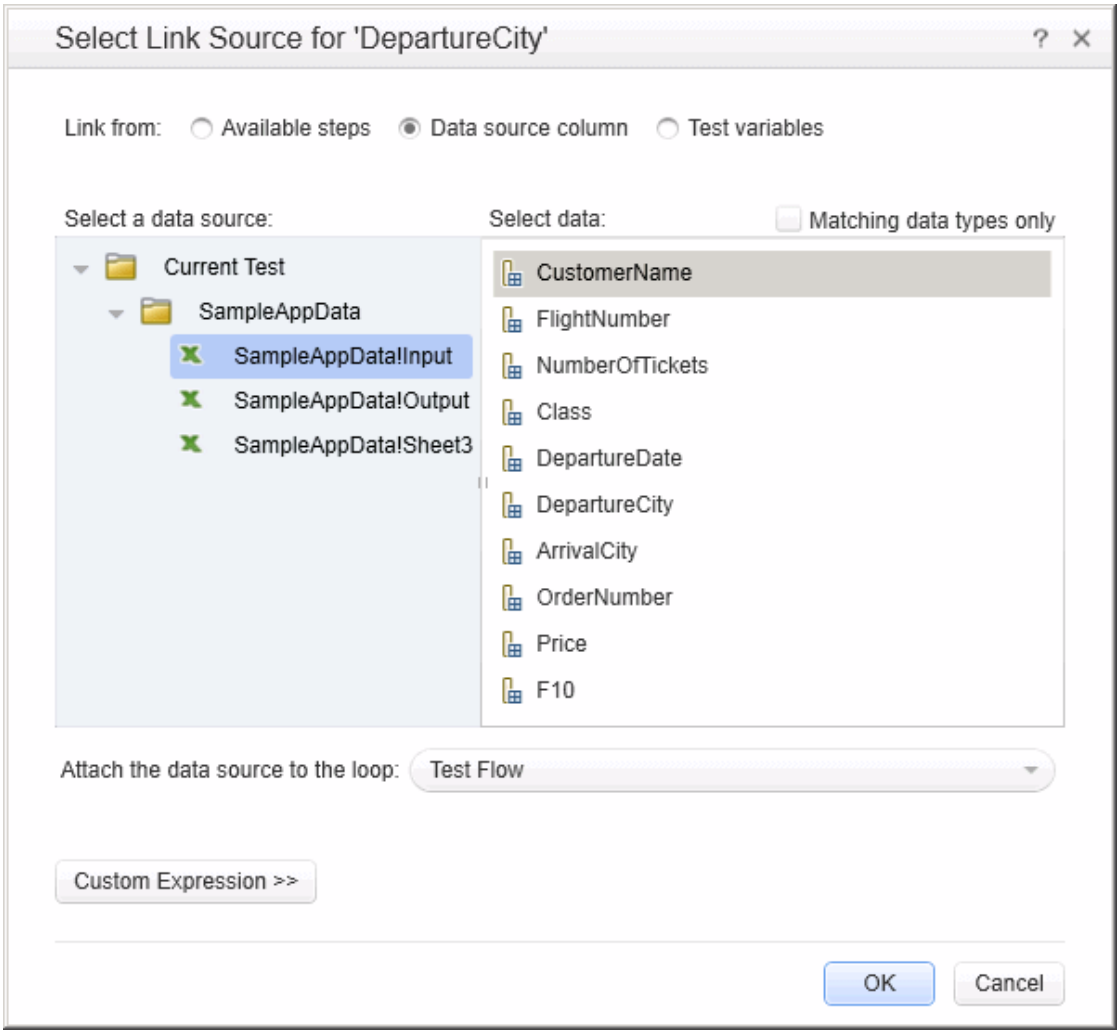
You can manually populate the checkpoint values or load values that were obtained from an earlier run using the **Load from Replay** button. The **Load from Replay** button is enabled after running the step test at least once—either by running the entire test or executing the individual step using the **Run Step** command. For details see "Run Step Results Pane Overview" on page 220.

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" (described on page 630) enables you to link to data in several ways:

- Reference a property of a another step
- Use a user or system test variable

- Use data from a data source, such as an Excel file or database table



You can also use a combination of these types: for example, a combination of a **Data source column** and **Test variable** sources to make a single expression.

To use data source columns, you add one or more data sources to the test through the Data pane. For details, see the "Data Pane" on page 96.

When preparing scripts for load testing, the "Select Link Source Dialog Box" (described on page 630) 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 633.

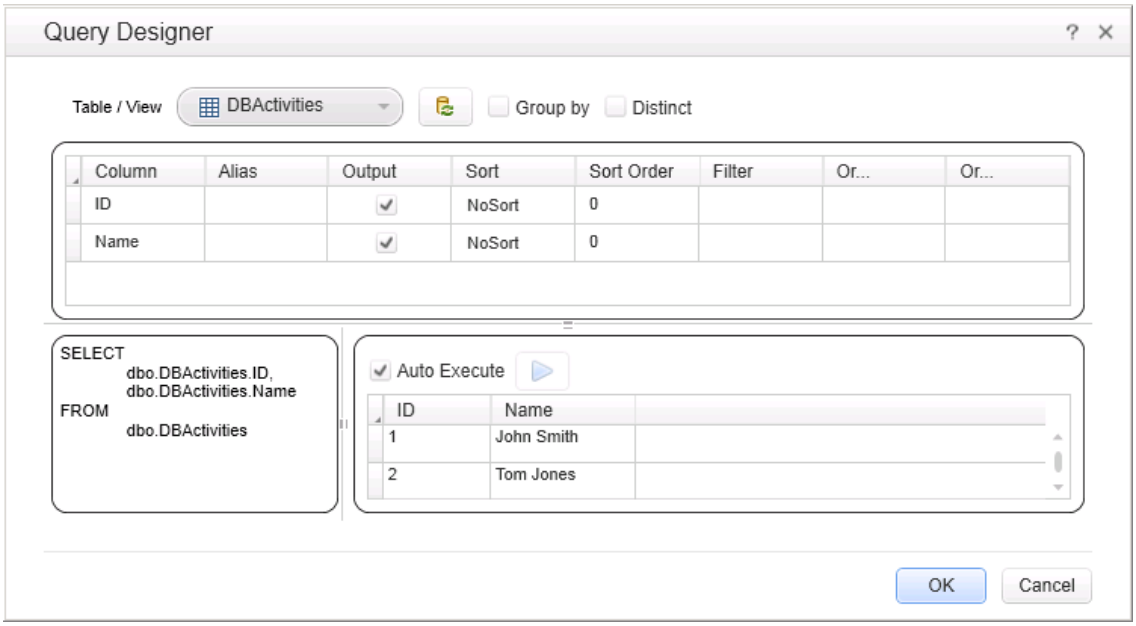
For task details, see "How to Assign Data to Test Steps" on page 621.

Creating a Query to Retrieve Database Tables

If you choose to create a data source from a database, Service Test lets you retrieve data from both OleDB and ODBC databases. A wizard guides you through connecting to a database and creating an SQL query statement. After you complete the wizard, the Data pane displays the database's data tables.

The connection builder assists you in creating a string for your connection. For details, see "Connection Builder Dialog Box" on page 453.

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.

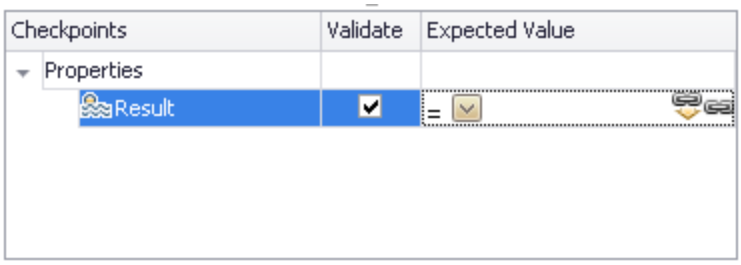


For details on how to use the Query Designer, see the "Query Designer Dialog Box" on page 456.

To learn more about using the wizard, see "Add a database data source" on page 91.

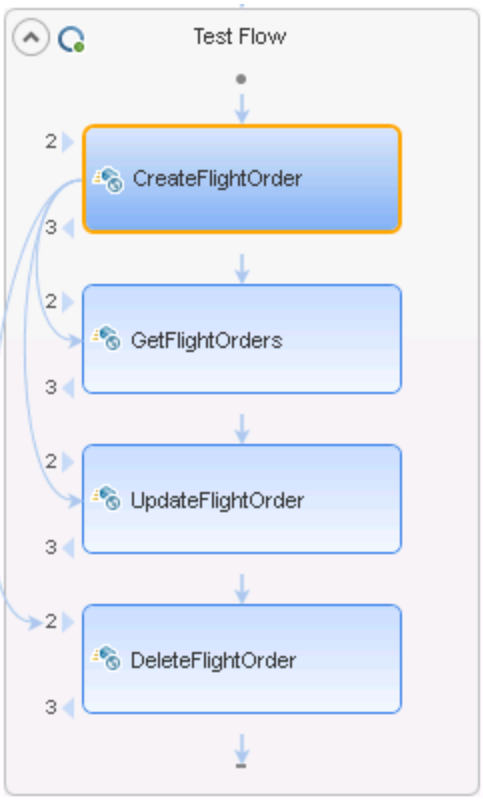
Outgoing Links

When you link a property to another step's property, Service Test indicates this with a **Outgoing Links** button in the Properties pane's **Input/Checkpoints** list.



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.

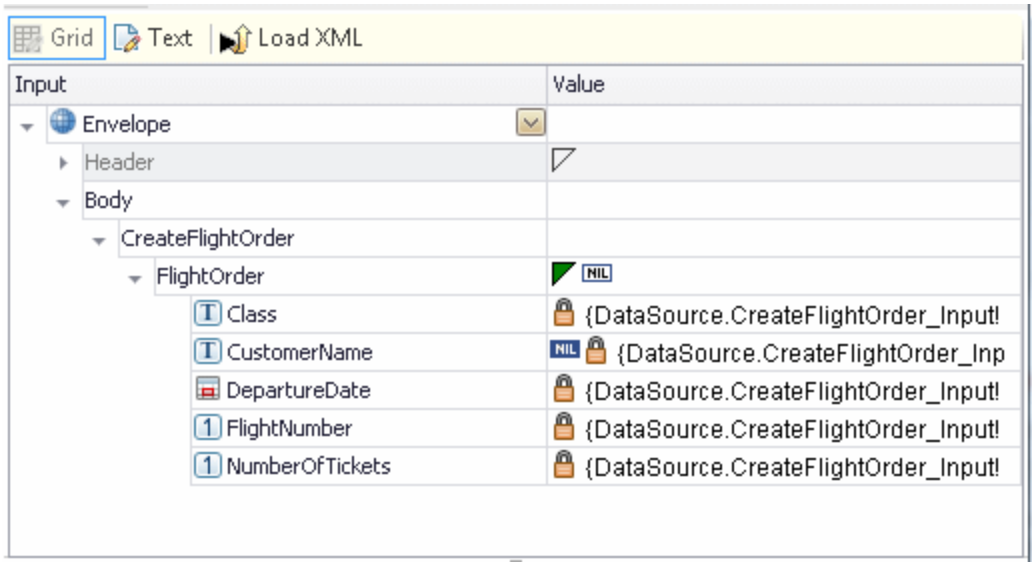
The following example shows three steps linking to a single output property of the `CreateFlightOrder` step.



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. This selects the target step on the canvas and the target property in the Properties pane.

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.

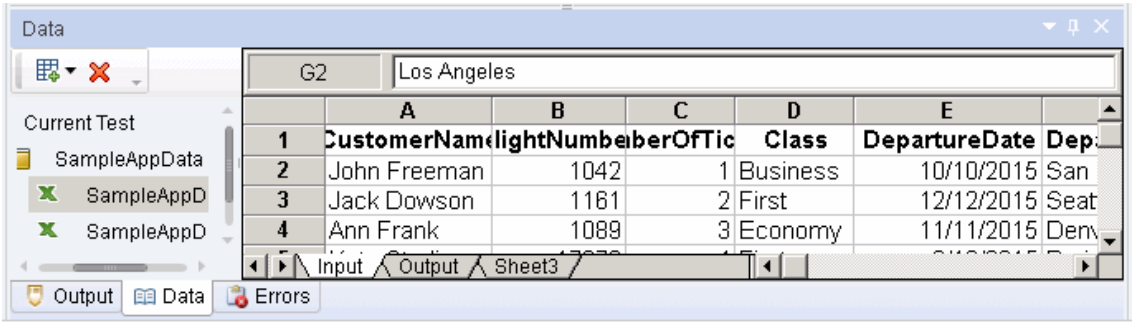


These expressions refer to data in a new editable Excel table or XML structure, depending on the type of provider you specified during the data driving.

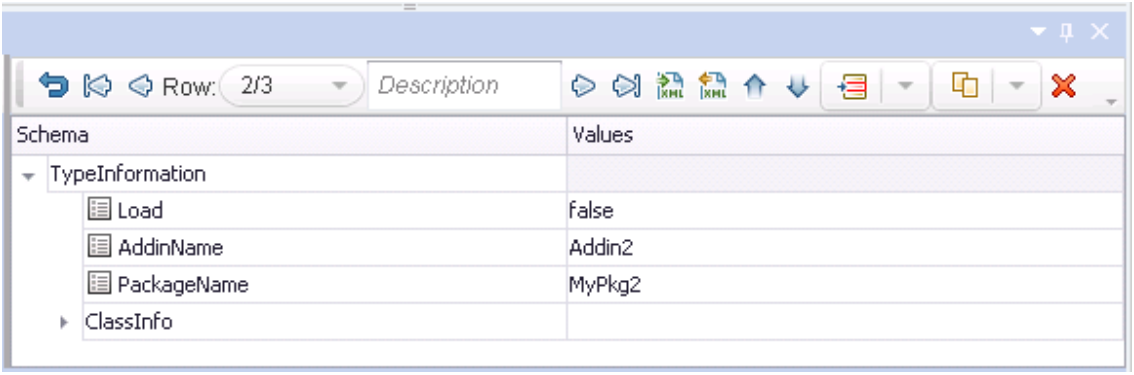
You can customize data-driving in the following ways:

- 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 625.
- Whether to use the same or two different data sources, when data driving two sections, such as Input and Checkpoint properties (Enabled only for Excel data sources).
- Whether or not to set the target step's loop, as a "For Each" type, based on the generated data source.

For Excel type tables, you can manually add to and edit the data in the Data pane.



For XML data sources, you can manually edit the XML root.



For task details, see "How to Data Drive a Test Step" on page 623.

For user interface details, see "Data Driving Dialog Box" on page 625.

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:

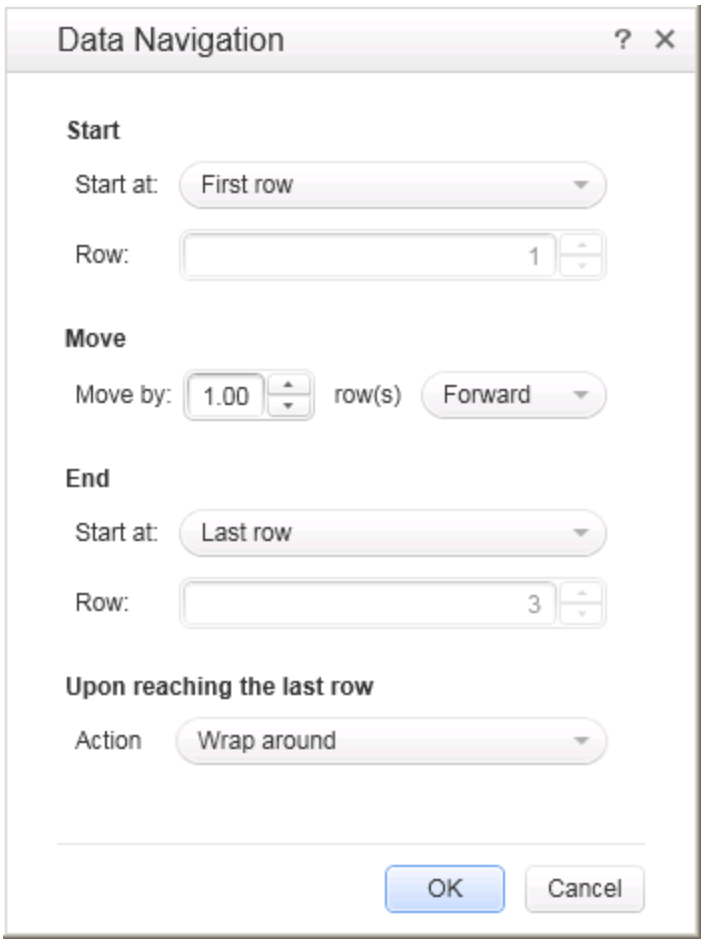
Navigating within the Data

Using the "Data Navigation Dialog Box" (described on page 634), you control the way you use the values in the data tables.

The Data Navigation settings work together with the loop settings described in "Loop - Input Properties" on page 418.

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.

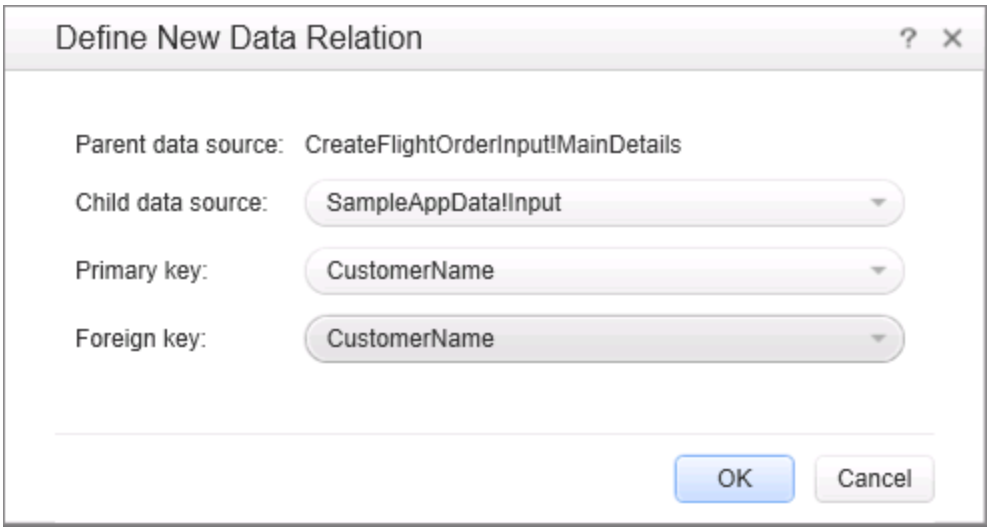
For data sources that are not designated as the loop collection, but whose values are fetched by steps within the loop, the Data Navigation policy affects the data differently. It indicates the order in which the values are fetched from the data source, and assigned to steps within the loop.



For user interface details, see "Data Navigation Dialog Box" on page 634.

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 94.

For user interface details, see the "Define New/Edit Data Relation Dialog Box" on page 637.

Data Keywords

You can use keywords to customize the test run and validation. For example, **SKIP** omits a property in the request or in the validation.

Service Test provides keywords for both input properties and checkpoints. You can set keywords in the following ways:

- Select the property and choose **Insert Keyword** from the right-click menu.
- Link to a data source containing the keyword.
- Type to keyword into the **Expected Value** column (checkpoints only).

For the manual options, make sure to use the required format by enclosing the keyword with hash (#) signs. Keywords are not case-sensitive.

Input Keywords

The following keywords are supported for input properties:

Keyword	Description
#SKIP#	Omits this element from the XML of the request. This is useful for elements of SOAP requests, for which `minOccurs = 0`
#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, this is reported to the log.

Checkpoint Keywords

The following keywords are supported for checkpoints:

Keyword	Description
#EXISTS#	Verifies that the element is present in the XML response. In this evaluation, the actual value is ignored—it only checks for the presence of a value. This is useful for SOAP response elements, for which `minOccurs = 0`.
#NOT_ FOUND#	Verifies that the element is not present in the XML response. In this evaluation, the value is ignored. This is useful for SOAP response elements, for which `minOccurs = 0`.
#SKIP#	Informs the run engine to ignore this value when evaluating checkpoints.

To check if an element has a **nil="true"** attribute in its response, select or clear its NIL icon in the user interface.

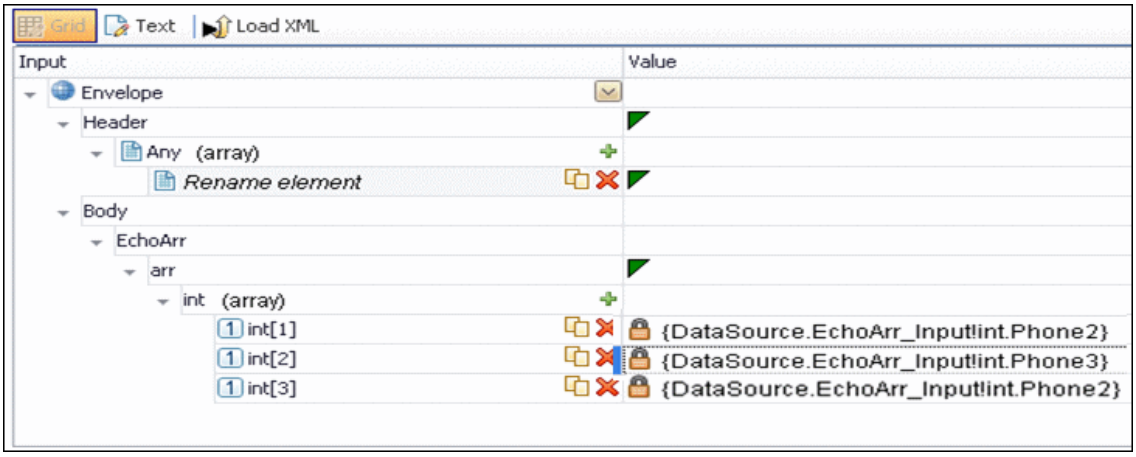
Data Assignment in Arrays

When your step has properties that are arrays, you can assign them data as a fixed-size array or through data relations:

Fixed-Size Array Assignment

In the fixed-size method, you assign each element of a fixed-size array to any column in a data table. You can assign each array element to a different data table column.

The following example shows three array elements assigned to different columns of a data table.



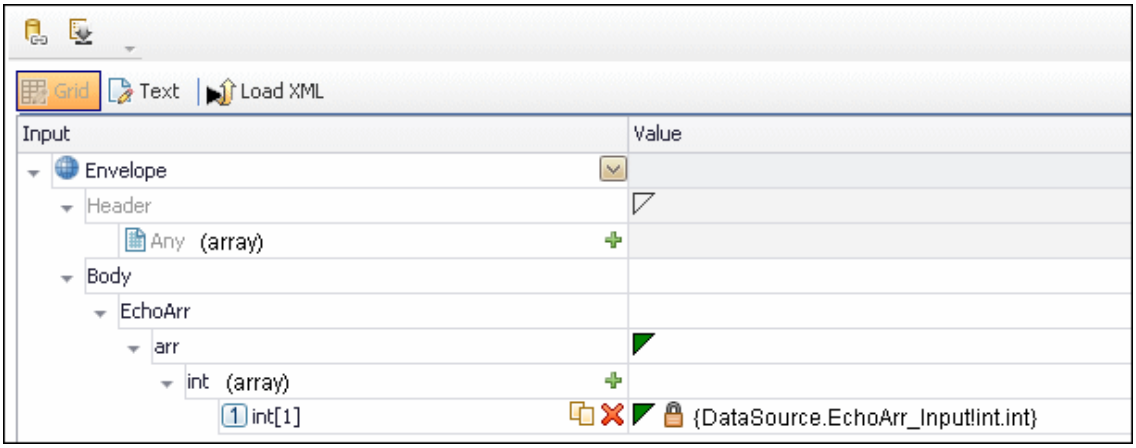
Data Relation Based Assignment

When you have one or more data relations defined, as described in "Create a new child relation" on page 94, Service Test assigns data from a single column. You link the first element of the array to a column in a data source.

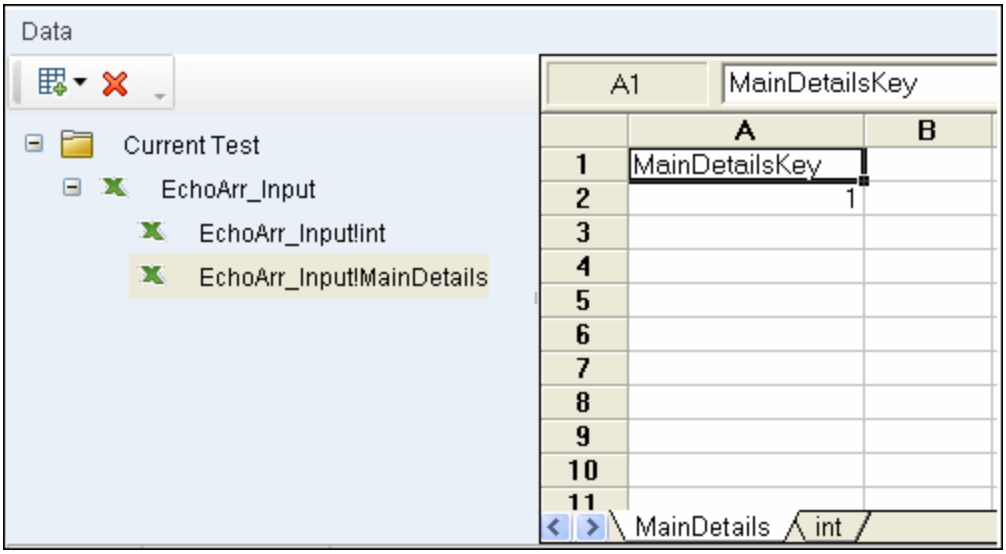
In order for Service Test to assign data based on a data relation, the following conditions must be present:

- The data source in the link is defined in a data relation.
- The parent data source is attached to the loop containing the step.
- Only the first element of the array is mapped to the child data source. If you map another array element to a different column, simple based assignment will be used.

The following example shows a single array element linked to a child data source. In this example, all elements of the array will take values from the same column, using the data-relation based assignment.



The link to the first element indicates the column from which the values for all of the array elements will be taken.



If you create a simple link to data, and the data source is already designated as a child in a data relation, the behavior will be relation-based.

Data-driven array elements will always be assigned using the data relation based assignment, using the column assigned to the first element.

Assignment Types

For relation-based data assignment, the drop down list adjacent to the name of the parent array node provides the following options for checkpoint validation:

- **Fixed**. Checks that each of the returned array elements matches the corresponding array element in the data table in the Data Pane. Each array is marked by an index number, as it checks the arrays by their index.
- **All**. Checks that all of the returned array elements match the array element in the Data Pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked >= 2 and the same property in another array element is set to <=10, the test run will check that all returned values are between 2 and 10.
- **Contains**. Checks that at least one of the returned array elements matches the value of the property in the Checkpoints pane. In this mode, arrays are not marked by an index number.
- **Fixed**. Checks that each of the returned array elements matches the corresponding array element in the data table in the Data Pane. Each array is marked by an index number, as it checks the arrays by their index.
- **All**. Checks that all of the returned array elements match the array element in the Data Pane. In this mode, arrays are not marked by an index number. For example, if a property in the first array is marked >= 2 and the same property in another array element is set to <=10, the test run will check that all returned values are between 2 and 10.
- **Contains**. Checks that at least one of the returned array elements matches the value of the property in the Checkpoints pane. In this mode, arrays are not marked by an index number.

Tasks

How to Assign Data to Test Steps

This task describes how to link the test step properties to data sources through the following steps:

- "How to Assign Data to Test Steps" above
- "Select a linking option" below
- "Assign data to array elements - optional" below

Select a linking option

1. In the "Select Link Source Dialog Box" (described on page 630), select a data source option:
   - **Available steps.** Select a step in the left pane, **Select a step**, and a property in the right pane, **Select a Property**.
   - **Data source column.** In the left pane's tree, select a data source. In the right pane, double-click on a data column or element. For properties that only have arrays, see the next step, about assigning data to array elements.
   - **Test variables.** Select a user or system variable. For details on defining user variables and profiles, see "Define user variables" on page 60.
2. To use a combination of multiple data source types for a property value, click **Custom Expression** to expand the Select Link Source dialog box. Select the data sources one-by-one and click **Add**. If necessary, edit the expression manually.
3. If the source to which you want to link is an array element or descendant of an array, a message prompts you to approve the creating of a loop around the current step. This enables you to iterate through all of the array elements.

   Click on the **Loop** frame to set the loop settings. For details, see "Loop - Input Properties" on page 418.

   If you do not need to run through the array elements, click **No**. Service Test links the first array element to the destination property.

   To continue without linking any of the elements, click **Cancel**.
4. Click **OK** to complete the linking of the data. Service Test copies the values or expressions into the **Value** column of the grid.

   Note:
   - You can only link properties to data sources if the steps are within a loop, such as **Test Flow** or custom loops.
   - 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 630.

Assign data to array elements - optional

Determine the type of data assignment that is required for the array values:

- To link each array element to a different column in the data table, use the **Link to Source** button to link each element to a different column. For details, see "Fixed-Size Array Assignment" on page 618.
- If you have data relations defined, or if you performed data driving on the array, then you link the first element of the array to a data column. All array elements will take values from that column.

Note:

In Web service calls, if you add an element to an **Any** type array and rename it, the **Link to Source** button is not immediately available.

Workaround: Reselect the step in the canvas and then access the **Link to Source** button.

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 615.

This task includes the following steps:

- "Open the Data Sources tab" below
- "Select a data source" below
- "Open the Data Navigation dialog box" below
- "Set the navigation properties" below

1. ### Open the Data Sources tab

   In the canvas, click within the loop frame, but not within a test step. In the Properties pane, open the Data Sources tab (visible only for the current loop).

2. ### Select a data source

   Click **Add** to select a data source. The "Attach Data Source to Loop Dialog Box " (described on page 629) opens. Double-click a data source.

3. ### Open the Data Navigation dialog box

   In the Data Sources tab, select the data source with your data. Click **Edit** to open the "Data Navigation Dialog Box" (described on page 634).

4. ### Set the navigation properties

   a. In the "Data Navigation Dialog Box" (described on page 634), 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. Alternatively, you can indicate to move to a random row.

   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 634.

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 614.

This task includes the following steps:

1. ## Prerequisites

   Create a step in the canvas. For details, see "Create the test flow - optional" on page 383.

2. ## Open the Input Properties tab

   Display the step's properties grid. Select **View > Properties** and click the **Input/Checkpoints** tab.

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" (described on page 625) 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. If you specified an **Excel** provider and data-driving for both sections, specify whether to place the data for both sections in the same Excel worksheet or different ones.

   e. Indicate whether you want to **Configure '<loop name>' as a For Each loop using the new data source**. This option repeats the steps in the loop frame 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 via the loop properties. This setting overrides the general loop properties. For details, see "Loop - Input Properties" on page 418.

   f. Click **OK**.

   g. 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 properties are within an array, you must create at least one array element to allow data driving. To add an array element, select the array's parent node and click the **Add** button . If you do not add at least one element, then the array will not be data driven.

For user interface details, see the "Data Driving Dialog Box" on next page.

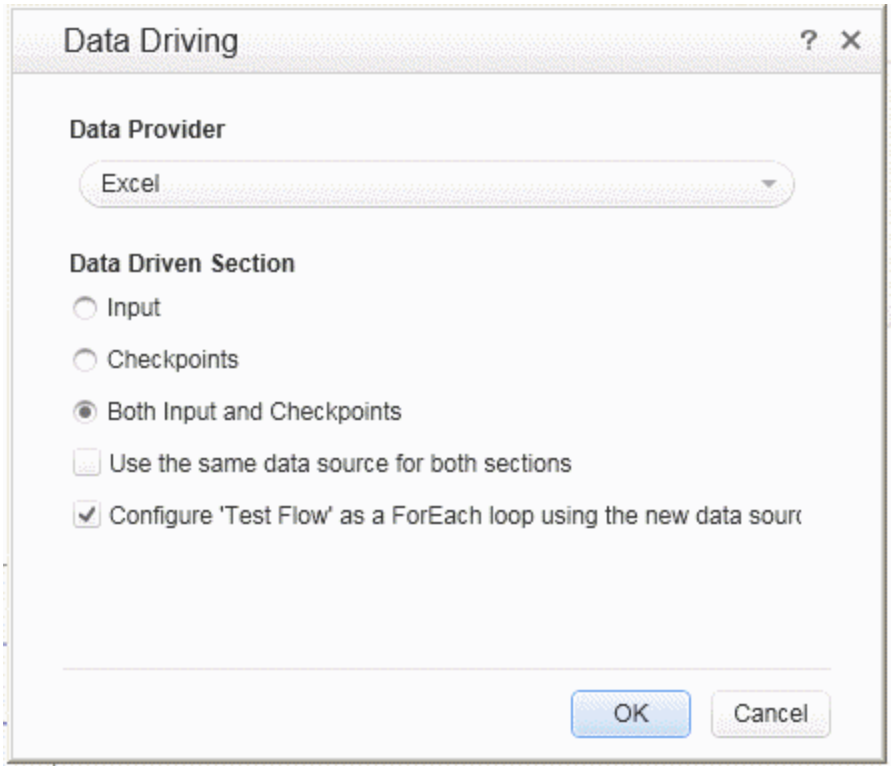
5. ## Enter values

   In the Data pane (**View > Data**), edit the contents of the newly created data tables or XML source.

Reference

Data Driving Dialog Box

Enables you to populate the input property and checkpoint values for a step, with data expressions.



To access	1. Select a step within the canvas. 2. Click in the Properties pane's toolbar.
Relevant tasks	"How to Data Drive a Test Step" on page 623
See also	"New/Change Excel Data Source Dialog Box" on page 98

The following elements are included:

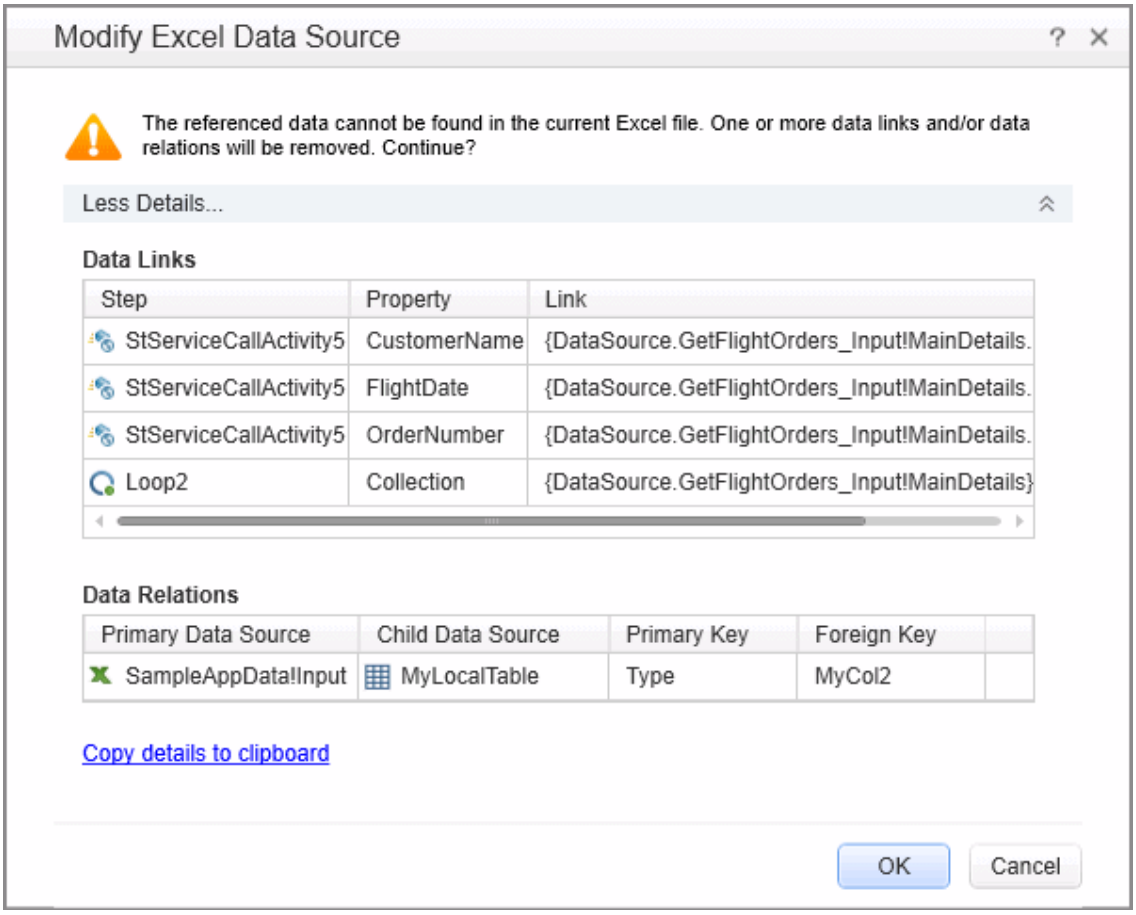
UI Elements	Description
Data Provider	The source type of the data: **Excel** or **XML**.

UI Elements	Description
Data Driven Section - for Input/Checkpoint 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 properties, you must create at least one array element in order to data drive the array. • **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 Section - for Request/Response	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:** These options are only available after importing a schema into the **Input** and/or **Checkpoints** sections.
Data Driven Section - for SOAP Fault properties	**Fault Properties.** The SOAP Fault properties to data drive. This option is available when invoking data driving in the Properties pane's **SOAP Fault** view. **Note:** These options are only available for Web Service and SOAP Request steps.
Data Driven Section - for Multipart 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.

UI Elements	Description
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. This setting is applied to the innermost loop containing the data-driven step. **Note:** This option overrides the configuration that you may have set for the Test Flow/Loop settings. For details, see "Loop - Input Properties" on page 418.

Data Link/Relation Message Box

This message box lists the elements that were not found in the new Excel file, when you changed the data source's existing Excel file.



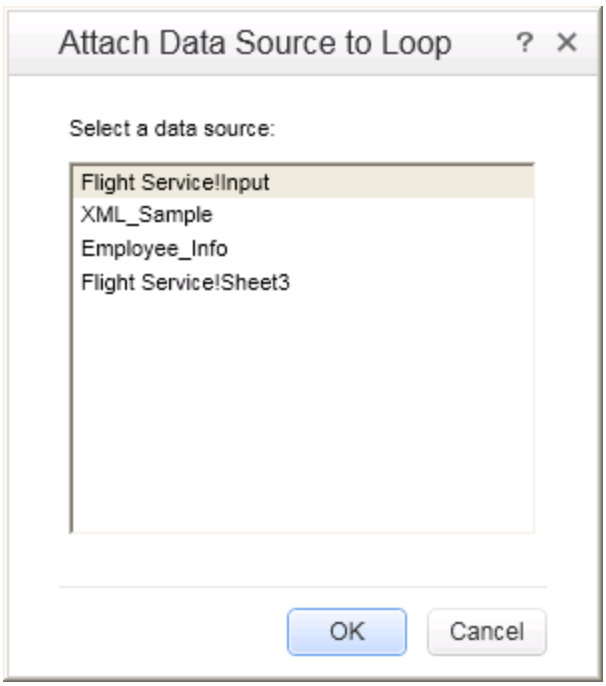
To access	Do the following: 1. Save the test. 2. In the Data Pane, select the main node of an existing Excel data source. 3. In the Properties pane, click **Change Excel File**. 4. In the "New/Change Excel Data Source Dialog Box""New/Change Excel Data Source Dialog Box"(described on 98), specify a new Excel file. 5. Click **OK** to permanently remove all of the listed items and save the test. **Note:** This dialog box only opens if the new Excel file has renamed or missing worksheets or columns, which affect data links or data relations.
Relevant tasks	• "Add an Excel data source" on page 90 • "How to Assign Data to Test Steps" on page 621
See also	• "New/Change Excel Data Source Dialog Box" on page 98 • "Data Relations and Navigation" on page 615

User interface elements are described below:

UI Elements	Description
More Details/Less Details	Expands or Collapses the **Details** section to show or hide data links or data relations that will be removed when you click **OK**.
Data Links table	A table of all of the affected data links. The table columns are **Step**, **Property**, and **Link**.
Data Relations table	A table of all of the affected data relations. The table columns are **Primary Data Source**, **Child Data Source**, **Primary Key**, and **Foreign Key**.
Copy details to clipboard	Copies the details to the clipboard, in a comma separated format.

Attach Data Source to Loop Dialog Box

This dialog box enables you to add a data source to the current loop.



To access	1. Create or open a test or component. 2. Define data sources in the Data pane 3. Open the Properties pane (CTRL+ALT+P). 4. Click within the **Test Flow**. 5. Open the **Data Sources tab** . 6. Click the **Add** button.

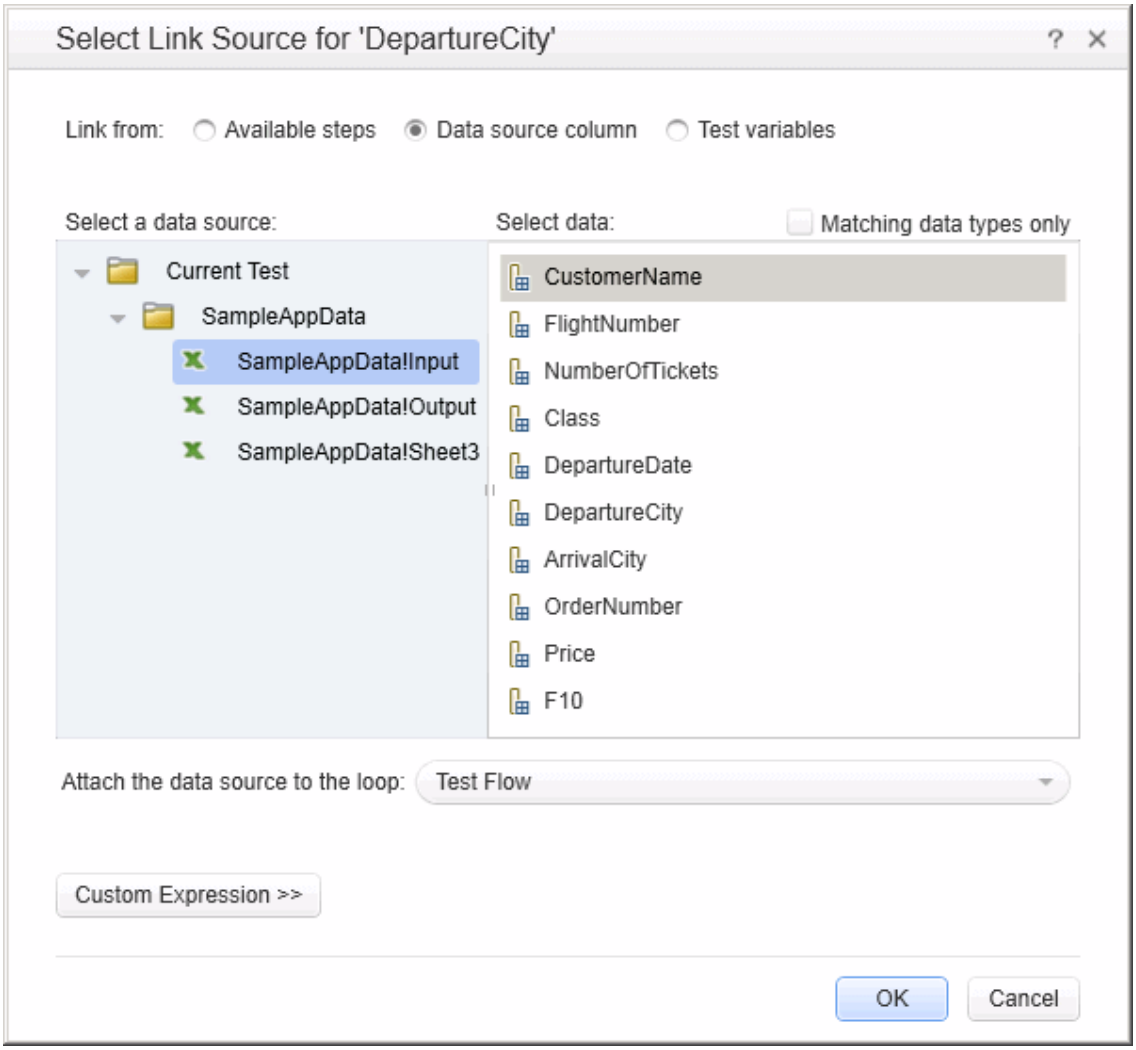
Important	You must first define data in the Data pane before you generate a list of data sources. When you link any step in the loop to the data, it automatically adds a data source to the loop. For details, see "How to Assign Data to Test Steps" on page 621.
Relevant tasks	"How to Set the Navigation Properties" on page 622
See also	"Data Navigation Dialog Box" on page 634

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, move the cursor to the right of the property value area. • 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 621

See also	• "How to Create Data Sources" on page 90 • "How to Set the Navigation Properties" on page 622

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

UI Elements	Relevant for	Description
Link from:	All types	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.
Select a step	Available steps	A tree hierarchy 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.
Select a property	Available steps	The step's linkable properties. The displayed sections depend on the step type. For most step types, the Checkpoints section is displayed.
Select data	• Available steps • Data source column	• For table-based data sources such as Excel, local tables and database: a list of the columns in the data table. • For XML data sources: a hierarchy of nodes.
Matching data types only	Data source column	Hides all columns (or schema rows for XML data sources) whose data types do not match the target property whose value you want to set.
Attach the data source to the loop	Data source column	A drop down list of the loops in the test containing the step. The data source is only associated with the selected loop. **Default:** Innermost loop containing the step.
<user variable grid>	Test variables	A list of the user-defined test variables and their values in the current profile.

UI Elements	Relevant for	Description
<system variable grid>	Test variables	A list of the system environment variables and their values: • IsLoadTest • TestDir • TestName • LocalHostName • OS • OSVersion • ProductDir • ProductName • ProductVer • UserName • CurrentDate • QcUrl • QcUserName • QcPassword • QcDomain • QcProject The following variables are available when running the test on a remote agent: • CurrentRunID • CurrentTestSet • CurrentTestSetID • CurrentTestInstanceID
Custom Expression	All types	Expands the dialog box to allow you to manually edit the link and create a custom expression. This is useful for creating an expression that uses multiple sources.

Data Retrieval Options for Load Test Enabled Tests

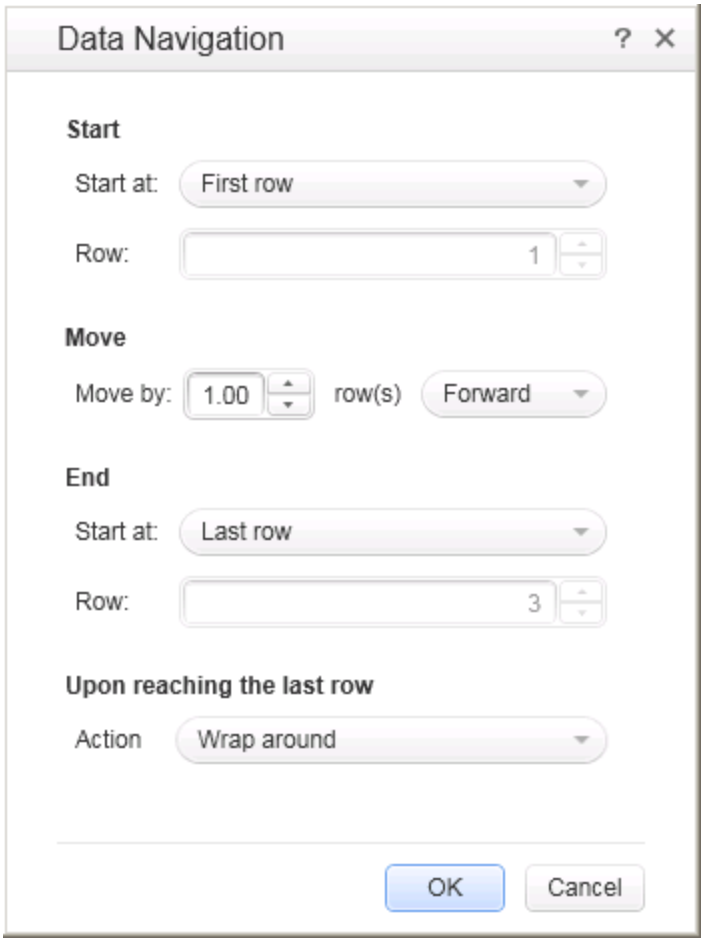
For Load Test Enabled tests, the Select Link Source dialog box provides additional controls for the Input properties, when choosing the **Data source column** option.

User interface elements are described below:

UI Elements	Description
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.**.

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	1. Create or open a test or component. 2. In the Properties pane, link a property to a data source . 3. Click within the step's loop such as Test Flow in the canvas. 4. Open the Properties pane's **Data Sources** tab . 5. Select a data source in the list attach and click **Edit**.
Relevant tasks	"How to Set the Navigation Properties" on page 622
See also	"Navigating within the Data " on page 615

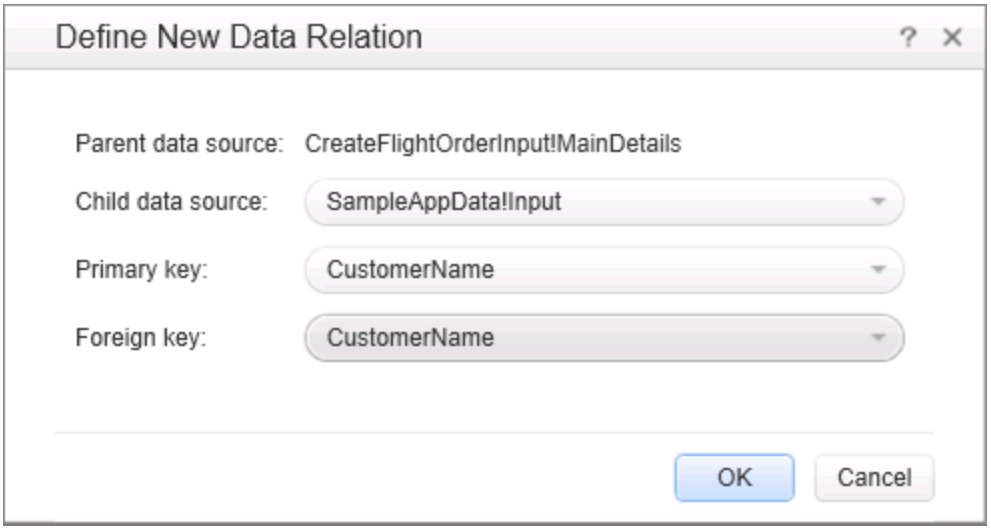
User interface elements are described below:

UI Elements	Description
Start	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 (not relevant for **Direction > To a random row**). • **Direction.** The direction in which to move: **Forward**, **Backward,** or **To a random row**.
End	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 "Loop - Input Properties" on page 418) 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 the **To a random row** option, each row is visited once in random order. The navigation ends after all of the rows were visited.

UI Elements	Description
Upon reaching the last row - Action:(Move Forward)	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:(Move Backward)	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 New/Edit 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	1. Create or open a test or component. 2. Make sure you have at least one data source in the Data pane. 3. Select a data source in the Data pane. 4. Click in the Properties pane. The **Data Source Properties** view opens. 5. Click **Add** or **Edit**.
Relevant tasks	"Create a new child relation" on page 94

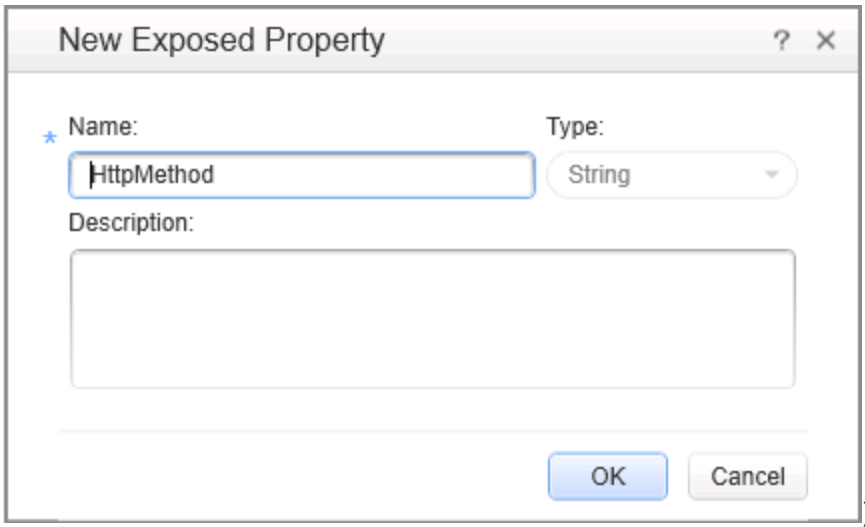
User interface elements are described below:

UI Elements	Description
Parent data source	The name of the parent data source, selected in the Properties pane (read-only).
Child data source	The sheet or table to use for the child 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.

New Exposed Property Dialog Box

Note: This dialog box is relevant only if you are working with tests created in Service Test 11.51 or earlier.

This dialog box enables you to expose an internal HTTP property and make it available to other steps, from the wrapper. This is primarily useful for REST service steps with inner HTTP Requests.



)

To access	Do one of the following: **From within the Add/Edit REST Service dialog box:** • Define a REST service method. • In the right pane, click the **Custom Input/Checkpoints** tab. • Click within a property row. • Select **Expose as Input Property** or **Expose as Output Property** from the right-click menu. **From within the canvas:** • Add a REST method to the canvas. Expand it and select its inner **HTTP** step. • Open the Properties pane and click within a property row. • Select **Expose as Input Property** or **Expose as Output Property** from the right-click menu.
Relevant tasks	"How to Create a REST Method" on page 475
See also	"Add/Edit REST Service Dialog Box" on page 495

The following elements are included:

UI Elements	Description
Name	A name for the exposed property.
Type	A drop down list of simple data types, such as `String`, `Integer`, and so forth. This field is read-only as it is determined by the exposed property's type.
Description	A description of the property.

Troubleshooting and Limitations - Data

This section describes troubleshooting and limitations for working with data.

General

- If a specified data source is or becomes inaccessible, the test will not fail. The Errors pane and report, however, indicate that there was an error in retrieving the data.
- Keywords such as **#SKIP#**, and so forth, are not supported in the Input property grid.

  Workaround: Link to a data source that contains the keyword.
- When adding a referenced Excel data source, if the file requires special credentials (for example, a location on another domain or a drive requiring authentication), you must verify that the operating system will allow access to the file.
- 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.
- Linking to file names is not supported for **HTTP Request** and **HTTP Receiver** activities that use the **File** type message body.
- For data sources with child relations: If you change the name the Key column in the child sheet, the Define New/Edit Data Relation dialog box does not reflect the new column name in the drop down lists.

Data Driving

- Data driving for JSON requests or responses, is only supported for Excel.
- Data driving for an Excel data source is only effective for the first 254 properties of a step. If a step has more than 254 properties, they will not be data-driven.
- Data driving for an Excel data source is only supported for property values consisting of 255 characters or less. If a property value has more than 255 characters, the data driving mechanism offers to truncate the string.
- When data driving a test step that has an array with two or more nested levels, the data driving engine only copies the first element of each array to the Excel data tables.
- Data driving for JSON requests or responses, is only supported for Excel.
- Data driving for an Excel data source is only effective for the first 254 properties of a step. If a step has more than 254 properties, they will not be data-driven.
- Data driving for an Excel data source is only supported for property values consisting of 255 characters or less. If a property value has more than 255 characters, the data driving mechanism offers to truncate the string.
- When data driving a test step that has an array with two or more nested levels, the data driving engine only copies the first element of each array to the Excel data tables.

Chapter 31: Coding Service Test Events

This chapter includes:

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 Tab (Properties Pane)" on page 193.

The **Custom Code** activity enables 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 Editor enables you to use autocompletion to prepare the event handler. For task details, see "How to Open a Window for Writing Custom Code" on page 644.

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. To remove the handler, you must delete the code manually in the `TestUserCode.cs` tab.

For more details about the Editor, including how to navigate files from the Editor, see "Testing Documents Overview" on page 55.

This section also includes:

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 **CodeActivity** step:

```
CodeActivity4.Output.secondOutput = " Previous Code Activity Output
```

```
: " +
this.CodeActivity2.Output.FirstOutputParameter
```

Note: 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 IntelliSense. Instead, you need to parse the XML using an XPath expression.

To retrieve a property's XPath expression, select it in the Properties pane 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.

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.

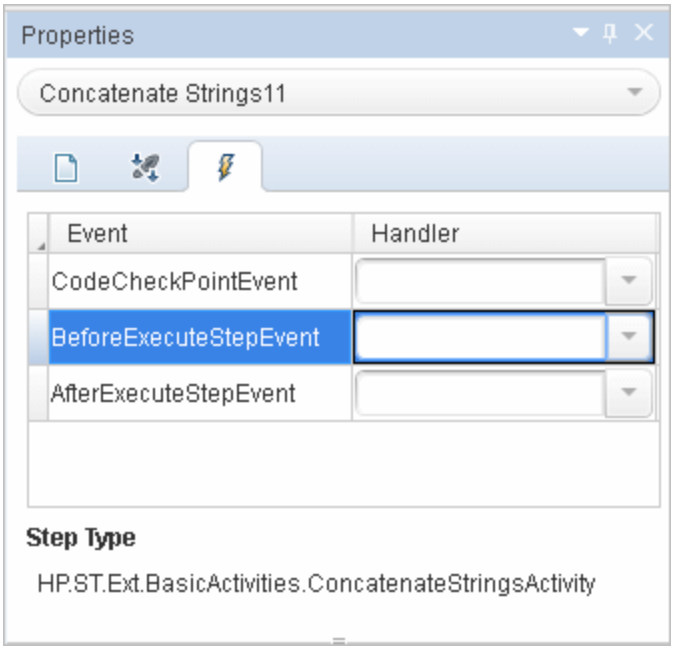
The checkpoint event handlers help you verify the output values in your test. You can use a Report, Assert, or Log function to gather information about your service.

This task includes the following steps:

- "Open the Events tab" below
- "Select an event" below
- "Edit the code" on next page
- "Enable access to the activity's properties" on next page
- "Save the changes" on next page

1. Open the Events tab

Select an activity title in the canvas and open the **Events** tab in the Properties pane.



Note: The Events tab can also be opened by double-clicking on a Custom Code activity in the canvas.

2. Select an event

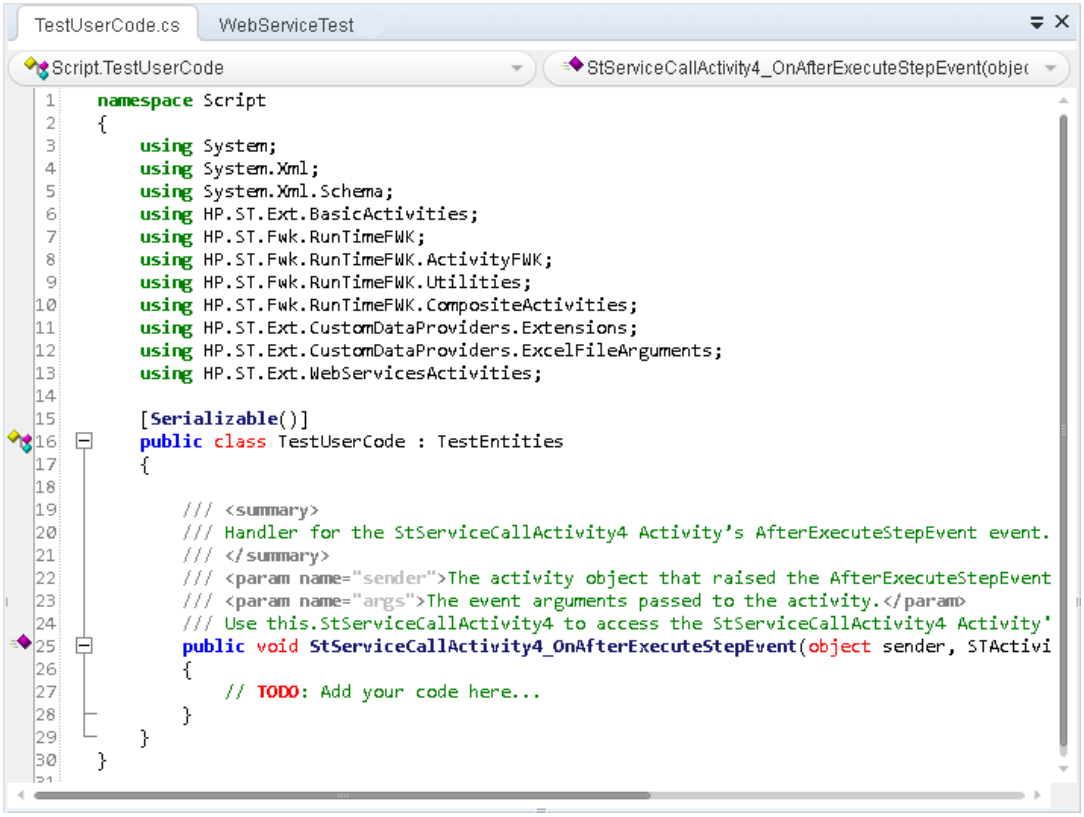
Select the event for which you want to provide code. Double-click in the **Handler** column. A

`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.

   Note: Changes you make in the user event code using the auto-completion drop down are not reflected in the canvas until you save the document.



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 the custom code and the test.

How to Access Input and Output Properties

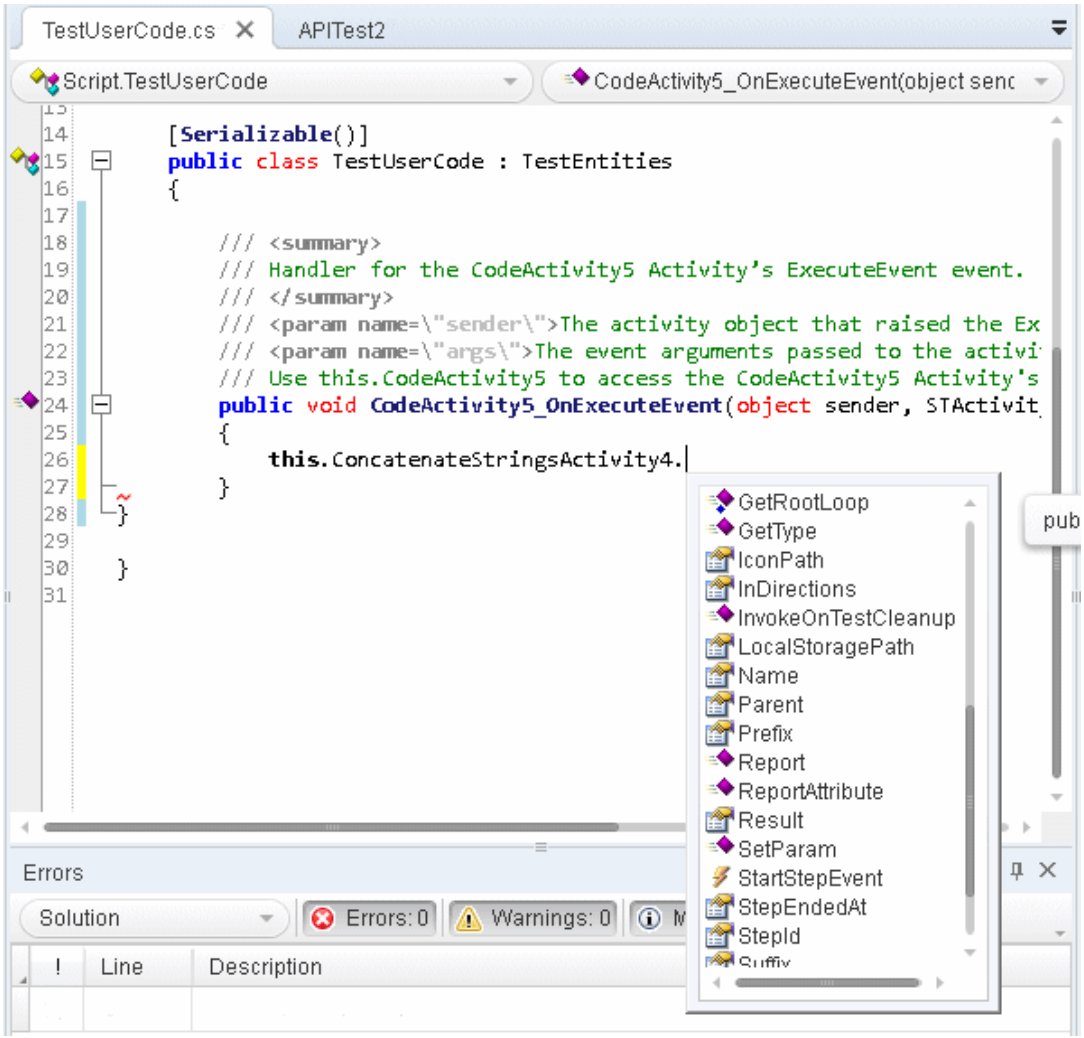
You can set the values of input and output properties through event handlers.

This task includes the following steps:

- "Select a property" below
- "Access the activity's parent" below

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 includes **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" below
- "Select a property" below
- "Set the activity's property values - optional" below
- "Access the activity's parent - optional" below

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 Manipulate Data During Run Time

You can manipulate data during run time using the `GetDataSource` method. For a list of the data source methods, use autocomplete or see "Data Source Methods" on page 657.

This task includes the following steps:

- "Prerequisite" below
- "Get the data source name" below
- "Retrieve a value from a data source" below
- "Set a property value" below
- "Import a file" on next page
- "Export a file" on next page

Prerequisite

Add at least one data source to your test through the Data pane. Save the test. Create a default event handler for the event you want to invoke for the test.

Get the data source name

The **GetDataSource** function requires the data source name as it appears in the Data pane. Select a node in the Data pane's left pane. Use the right-click command **Copy name** to copy the name of a data source node to the clipboard.

Retrieve a value from a data source

The following example illustrates the **GetValue** method retrieving a value from the first row of the data source and converting it to a string:

```
/ this.GetFlights4.FlightNumber =
GetDataSource("GetFlights4_Input!MainDetails").GetValue(0, "Flight_
Number").ToString();
```

To retrieve the row corresponding to the current iteration, indicate the current iteration in place of the row number. Note that the **GetValue** function is zero-based—it begins with row 0, while **CurrentIterationNumber** is one-based. For example:

```
/ this.GetFlights4.FlightNumber =
GetDataSource("GetFlights4_Input!MainDetails").
GetValue(this.Loop4.CurrentIterationNumber, "Flight_
Number").ToString();
```

Set a property value

The following example illustrates the **SetValue** method setting the value for the `Flight_Number` and `Tickets` ordered values.

```
GetDataSource("CreateFlightOrder4_Input!MainDetails").SetValue(0,
```

```
"Flight_Number", "Y22");
GetDataSource("CreateFlightOrder4_Input!MainDetails").SetValue(0,
"Tickets_Ordered", "2");
```

Import a file

The following example illustrates the **Import** and **ImportFromExcelFile** methods:

```
ExcelFileImportInputArgs a = new ExcelFileImportInputArgs
(@"C:\DemoExcel.xls", "MainDetails", true);
GetDataSource("CreateFlightOrder4_Input!MainDetails").Import(a);
GetDataSource("CreateFlightOrder4_
Input!MainDetails").ImportFromExcelFile(@"C:\DemoExcel.xls",
"MainDetails", true);
```

Export a file

The following example illustrates the **Export** and **ExportToExcelFile** methods:

```
ExcelFileExportInputArgs a = new ExcelFileExportInputArgs
(@"C:\ExportedExcel.xls");
GetDataSource("CreateFlightOrder4_Input!MainDetails").Export(a);
GetDataSource("CreateFlightOrder4_
Input!MainDetails").ExportToExcelFile(@"C:\ExportedExcel.xls");
```

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:

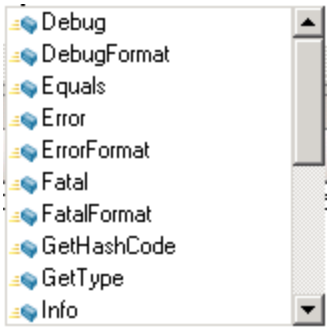
1. **Select the UserLogger object**

   Use autocompletion from the activity's **Context** object, and select the **UserLogger** object.

   ```
   this.StServiceCallActivity4.Context.UserLogger
   ```

2. **Choose a log type**

   Use autocomplete to select a log type.



For a list of the log types, see "Logging Options" on page 658.

3. **Complete the function**

   Provide to necessary arguments to complete the function as indicated by the tooltip.

   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 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** to open the Output pane.

   To view the output log file:

   a. Run the test at least once (**F5**).

   b. Select **Open Containing Folder in Explorer** from the test tab's right-click menu and open the `Log` folder.

   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 opening the Run Results Viewer from the Start menu.

Note: For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

This task includes the following steps:

- "Select the Report object" below
- "Complete the function" below
- "View the run results" below

1. ### Select the Report object

   Select the **Report** function 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 check the results in the Run Results Viewer. 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" below
- "Select the Checkpoint object" below
- "Validate a checkpoint - optional" on next page
- "Ignore selected checkpoints - optional" on next page
- "Customize the checkpoints reporting - optional" on page 653
- "View the results" on page 653

1. ### Create a checkpoint handler

   Double-click the **Handler** column of the `CodeCheckPointEvent` row in the activity's **Events** tab.

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 658.

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 (second argument) equals the expected value (first 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.
ConcatenateStringsActivity4.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 Properties pane's **Input/Checkpoints** tab. 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 enables 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 enables 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" below
- "Get or Set a variable" below

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" below
- "Decrypt the password - optional" on next page

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 Properties pane's **Events** tab for a Web Service's step.

This task includes the following steps:

- "Create an event handler" below
- "Add the Using text" below
- "Define an XmlDocument and load the XML" on next page
- "Create a new message" on next page

1. ### Create an event handler

   Select a Web Service step and provide values for its properties. Open the Properties pane's **Events** tab and double-click the **Handler** column in the **OnReceiveResponse** row. A code editor containing the new event handler opens.

2. ### Add the Using text

   In the namespace section, make sure that **System.Text** appears in the list of using statements.

```
namespace Script
{
using System;
using System.Xml;
using System.Text;
using System.Xml.Schema;
using HP.ST.Ext.BasicActivities;
...
```

3. ### Define an XmlDocument and load the XML

   In the **Todo** section of the **StServiceCallActivity1_OnOnReceiveResponse** function, add variable definitions for the message argument and envelope. Use autocompletion to select the **XmlDocument** method. Define a name for the envelope and assign it the value of the XML envelope.

```
var tmp = args.Message;
var envelope = Encoding.UTF8.GetString(tmp);
XmlDocument tmpXdoc = new XmlDocument();
tmpXdoc.LoadXml(envelope);
```

4. ### Create a new message

   Save the OuterXml content to a string, and apply UTF-8 encoding.

```
var newStrMessage = tmpXdoc.OuterXml;
args.Message = Encoding.UTF8.GetBytes(newStrMessage);
```

How to Set HTTP Headers for Web and REST Services

The following section describes how to set HTTP headers for Web service and REST service steps.

HTTP Headers for Web Service Calls

To dynamically add an HTTP header to a Web service call:

1. Open the Events tab in the Properties pane and implement the **OnBeforeApplyProtocolSettings** event.
2. Implement the method as follows:

```
this.StServiceCallActivity4.HttpRequestHeaders.Add("<key>",
"<value>");
```

HTTP Headers for REST Service Calls

To dynamically add an HTTP header to a REST service or to a REST sevice's inner HTTP Request step (when working with tests created in Service Test 11.51 or earlier):

1. Open the **Events** tab in the Properties pane and implement the **OnBeforeExecuteStepEvent** event.
2. First allocate the array the desired length. The following example allocated two headers.

```
(args.Activity as HTTPActivity).RequestHeaders =
new HP.ST.Shared.Utilities.Pair<string, string>[2];
```

3. Instantiate each array element with the **<HeaderName>** and **<HeaderValue>**. For example:

```
 (args.Activity as HTTPActivity).RequestHeaders[0] = new
HP.ST.Shared.Utilities.Pair<string, string>("HeaderName1",
"Value1");
 (args.Activity as HTTPActivity).RequestHeaders[1] = new
HP.ST.Shared.Utilities.Pair<string, string>("HeaderName2",
"Value2");
```

Note: You can also set the HTTP headers values by expanding the RequestHeader node in the Properties pane's Input/Checkpoints tab. You then link to a data source from the **Name** and **Value** rows. For details, see "Select Link Source Dialog Box" on page 630.

If you modify the headers using code in an event handler, it will override the values in the Properties pane during the test run.

Reference

Data Source Methods

This section describes the **GetDataSource** method and the data source methods that allow you to manipulate data in existing data sources during run time. These methods will not change the data displayed in the Data pane.

The **GetDataSource** method's only argument is the name of a data source node. The following list shows the common data source methods:

Method	Arguments	Relevant for Data Source Type
SetValue	row number, column name	All except XML
GetValue	row number, column name, value	All
Import	import argument (casted in a previous statement)	Excel
ImportFromExcelFile	path of Excel file, sheet name, is first row a header	Excel
Export	export argument (casted in a previous statement)	Excel
ExportToExcelFile	path of an Excel file	Excel
RowsCount	none	All

All of the methods and their syntax are available in the event handler window, using autocomplete.

For task details and examples, see "How to Manipulate Data During Run Time" on page 648.

Note: The **Query** function implemented in versions 11.20 and earlier is not supported in version 11.50. To modify runtime data through an event handler, replace all occurrences of the **Query** function with **GetDataSource**, using the arguments described above.

Context Methods

This section describes some of the common **Context** methods.

UI Elements (A-Z)	Description
	Denotes a function.
	Denotes a string.

UI Elements (A-Z)	Description
	Denotes an event.
EncryptionMngr	Enables you to encrypt and decrypt password strings.
TestInputParameters	Enables you to retrieve the test's input parameters.
TestIteration	Provides information about the test iteration.
TestOutputParameters	Enables you to retrieve the test's output parameters.
TestProfile	Enables 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 59.

For a complete list of the functions, use autocomplete.

Logging Options

The Service Test API provides the following logging options, available from the **Context.UserLogger** object:

UI Elements (A-Z)	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 enables 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 649.

Assert Options

The Assert object provides the following comparison operators:

UI Elements (A-Z)
Equals

UI Elements (A-Z)
Greater
GreaterOrEqual
Less
LessOrEqual
NotEquals

For task details, see "How to Write Checkpoint Events" on page 651.

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.
- When calling a test or action, you cannot access the data associated with the called test or action using runtime API coding, even if overriding the data was enabled (through the **Allow other tools to override the data** option).
- When working on non-English operating systems, you must install a localized version of .NET framework to obtain localized compilation errors.
- By default, on machines running 64-bit Windows, the ActiveX object, **HP.ST.Test**, is only registered in 32-bit mode—not 64-bit.
  Workaround: Register the assembly manually using the following command line code:

```
cd "<Service Test installation folder>\bin" %windir%\
Microsoft.NET\Framework64\v4.0.30319\
RegAsm.exe HP.ST.Fwk.SOAReplayAPI.dll /codebase
```

Chapter 32: Extensibility in Service Test

The HP Service Test installation includes the capabilities to create custom activities for the **Toolbox** pane. After defining a custom activity, you can drag it onto the canvas as you would with any other built-in activity.

This chapter includes:

Concepts

Creating New Activities - Overview

Service Test enables you to create custom activities to extend the capabilities of the product.

Once you create a new activity through this mechanism, it will be available in the **Toolbox** pane for all future tests.

For most custom activities, you can use the Activity Wizard. For C# users, the wizard creates a Visual Studio project into which you can add your own C# code. Java users can edit `.java` files which will be compiled into `.class` files. You then deploy the new activity into Service Test. For details, see "Activity Wizard" on page 684 and "How to Use the Wizard to Create a Custom Activity - C#" on page 673.

Advanced users can build custom activities manually, without the wizard. For details, see "How to Manually Create a Custom Activity in C#" on page 676.

Custom Activity Files

This section describes the structure and content of the files required to manually define a new activity in Service Test. The following information is not relevant if you are using the Activity wizard.

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 next page.

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 next page.

The **Addin** file is an XML file that references all of the activity component data. For details, see "Addin Files " on page 670.

In addition, you can also define resource files to store text strings used by your activity. For details, see "Resource Files " on page 672.

The product's installation includes a sample project in the `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 `<Installation_folder>\addins\CustomerAddins\<addin_name>` folder. This enables Service Test to load them during startup.

For more details, see "How to Manually Create a Custom Activity in C#" on page 676.

Note: This is a preliminary version of the SDK (Software Development Kit). It enables you to extend the capabilities of the product. 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.

Runtime Files

In addition to the signature and addin files, you must provide a DLL to run when executing the activity. You create a solution and customize the code as required. You can use the sample `ReportMessageActivitySample.sln` located in the product's `ExtensibilitySamples` folder as a basis for your project.

Use Microsoft's IntelliSense to determine the method's arguments and syntax.

Here are some guidelines that you must follow:

- As shown in the sample, you must use methods that are included in the **STActivityBase** class.

```
public class ReportMessageActivitySample : STActivityBase
```

- **STExecutionResult** is the return value of the **ExecuteStep** method. It receives one or two parameters: **STExecutionStatus Status** and optionally, **string msg**. STExecutionStatus is an **enum** type that can be set to the following values: `ActivityFailure`, `ActivityStopTest`, `ApplicationUnderTestFailure`, `Equals`, `ReferenceEquals`, `Success`, `TestStopped`. In the following example it receives the **Success** value.

```
return new STExecutionResult(STExecutionStatus.Success);
```

- Place your executable code within the **ExecuteStep** function.

```
protected override STExecutionResult ExecuteStep()
  { ...
```

Note: You must compile the DLL with a **Target Framework** of Framework 4.0, available in Microsoft Visual Studio 10.

For task details and an example, see "How to Create a Runtime File" on page 679.

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.

Resource Element

The **Resource** element has the following attributes:

Attribute	Description
type	The type of entity, in this case **Activity**.
id	A unique string that identifies 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** pane, for example **String Manipulation**. To add it to an existing group, specify a group name as it appears in the **Toolbox** pane. If the group does not exist, it adds a new group.
shortname	The short name of the activity as shown in the **Toolbox** pane's hint area (above the description).
description	The description of the activity as shown in the **Toolbox** pane's hint area.
assembly	The DLL file to call when running the activity, stored in the same folder as the signature and addin files.
className	The class implemented by the activity. **Note:** The class must inherit from the STActivityBase class.
image	An 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** pane.
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 primary **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.
isSharedMetadata	When **true**, enables 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 Properties pane.

The following table describes the sub-elements of **Section**:

Element	Description
Tab	The tabs to appear in the Properties pane 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** pane's toolbar. • **CanBeInPropertySheet.** If `true`, shows the tab in the Properties pane. • **CanBeInDataLinkDialog.** If `true`, shows the tab in the "Select Link Source Dialog Box" (described on page 630). **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 "Properties Pane" on page 184.
Alert	Enables 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.** Enables you to create an event handler to run when the test is verifying checkpoints.
	  ▪ **BeforeExecuteStepEvent.** Enables you to create an event handler to run before executing the activity.
	  ▪ **AfterExecuteStepEvent.** Enables 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 "Updating Web Services" on page 521.
	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 Properties pane, for example **Step ID** and **Name**.
- **InputProperties.** Defines the input properties located in the Input pane of the of the Properties pane's **Input/Checkpoints** tab.
- **OutputProperties.** Defines the output properties located in the **Checkpoints** pane of the of the Properties pane's **Input/Checkpoints** tab.

This section includes:

Elements and Sub-Elements

The elements and attributes are defined in the standard XML schema file, http://www.w3.org/2001/XMLSchema, or the built-in `types.xsd` schema, 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" />`
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 **String** type values, use `base="xs:string"`.
xs:enumeration	A tag indicating the beginning of list of values in the drop-down list 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 Properties pane.

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 the **Types.xsd** schema, use a **types:** prefix in the attribute name. For example `types:displayName`.

Attribute	Description
name	The internal name of the property or grid in the Properties pane. This is the name referenced by other calls and by the event handlers code. This is not the name displayed in the Properties pane's **Name** column.
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**".
maxOccurs	The maximum number of array elements the user may provide. To allow an unlimited amount, specify "**unbounded**"
types:visible	When **true**, enables 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 Properties pane.

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 Properties pane'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" below
- "Manifest Section" on next page
- "Runtime Section" on next page

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. • **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.
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** pane. • **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 "Prerequisite - create a runtime file" on page 677.

Tasks

How to Use the Wizard to Create a Custom Activity - C#

This task describes how to create a new activity, using C#, and deploying it in Service Test.

This task includes the following steps:

1. Run the Activity Wizard

Open the Activity Wizard from the product's start menu (**Start > All Programs > HP Software > HP Service Test> Tools > Activity Wizard**).

In the wizard's **General Properties** pane, select the **C#** as the **Language**.

Define the relevant properties and proceed to the last screen of the wizard. For details, see the "Activity Wizard" on page 684.

> **Note:** For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

2. Open the folder

On the final screen of the wizard, click **Open Folder** to open the `<Activity Name>` folder, corresponding to the activity name you specified in the wizard. Navigate to the following subfolder `SourceCode` and locate the `<Activity Name>.cs` file.

3. Add execution code

Add your execution code to the **ExecuteStep** function inside the `.cs` file.

```
protected override STExecutionResult ExecuteStep()
{
try
{
//**************************
// Execution code goes here
//**************************
```

```
...
```

4. **Add Logger code - optional**

   Add information for the log using the **LogInfo**, **LogDebug,** or **LogError** statements. For example:

```
protected override STExecutionResult ExecuteStep()
{
try
{
    LogInfo("Log Message 1");
    LogDebug("Log Message 2");
    LogError("Log Message 3");
...
```

5. **Add a Report statement - optional**

   Add a **Report** statement. For example:

```
protected override STExecutionResult ExecuteStep()
{
try
{
        DetailsReport = DetailsReport.Replace("\\n", "<BR>");
        this.Report("Message", DetailsReport);    ;
...
```

6. **Compile the project into a DLL**

   Build the project and make sure the current `<Activity Name> .dll` file is in the new activity folder that you specified in the wizard.

7. **Deploy the activity**

   In the final wizard screen, click **Deploy in Service Test**. Click **Finish** to close the wizard. Restart Service Test.

8. **Add a test step**

   - Open a new test and add the new custom activity (by default under the **Miscellaneous** category) into the Test Flow.
   - Provide values for the properties you defined for the activity in the wizard.
   - Run the test and observe the Output log and Run Results Viewer.
   - Enable checkpoints to verify the results and rerun the test.

How to Use the Wizard to Create an Activity -Java

This task describes how to create a new activity using Java code, and deploy it in Service Test.

This task includes the following steps:

1. ## Prerequisite

   Make sure you have a `JAVA_HOME` environment variable defined on your machine indicating the parent JDK folder.

2. ## Run the Activity Wizard

   Open the Activity Wizard from the product's start menu (**Start > All Programs > HP Software > HP Service Test > Tools >Activity Wizard**).

   In the wizard's **General Properties** pane, select the **Java** as the **Language**.

   Define the relevant properties and proceed to the last screen of the wizard. For details, see the "Activity Wizard" on page 684.

   Note: For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.

3. ## Open the folder

   On the final screen of the wizard, click **Open Folder** to open the **<Activity Name>** folder, corresponding to the activity name you specified in the wizard. Navigate to the following subfolder `<Activity Name>\hp\st\ext\java` and locate the `MyLogic.java` file.

4. ## Edit the code

   Edit the `ExecuteLogic` function inside the `MyLogic.java` file. Make sure to keep the **Properties** definition.

```
public Properties Props = new Properties();
public ExecutionResult ExecuteLogic()
{
    try{
    //*************************
    // Execution code goes here
    //*************************
```

```
    return ExecutionResult.Success;
    }
...
```

5. ### Add Logger code - optional

   Add information for the log using the **Logger.LogInfo**, **Logger.LogDebug,** or **Logger.LogError** statements. For example:

```
    try{
    ...
        Logger.LogInfo("Log Message 1");
        Logger.LogDebug("Log Message 2");
        Logger.LogError("Log Message 3");
    ...
    return ExecutionResult.Success;
    }
```

6. ### Add a Report statement - optional

   Add a Report statement, **Reporter.Report**, using key value combinations. For example:

```
    try{
    ...
        Reporter.Report{"Name","John");
    ...
    return ExecutionResult.Success;
    }
```

7. ### Compile the Java into a class

   Run the `CompileJavaFiles` batch file in the `<Activity Name>\hp\st\ext\java` folder to compile all java files into a class. This utility only compiles the files in its folder.

8. ### Deploy the activity

   In the final wizard screen, click **Deploy in Service Test**. Click **Finish** to close the wizard. Restart Service Test.

9. ### Add a test step

   - Open a new test and add the new custom activity (by default under the **Miscellaneous** category) into the Test Flow.
   - Provide property values in the Properties pane.
   - Run the test and observe the Output log and Run Results Viewer.
   - Enable checkpoints to verify the results and rerun the test.

How to Manually Create a Custom Activity in C#

This task describes how to create a new activity and implement it 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 `<Installation_Folder>\addins\CustomerAddins\<addin_name>` folder.

This task includes the following steps:

- "Prerequisite - create a runtime file" below
- "Create a signature file" below
- "Create an addin file" on next page
- "Provide a graphic for your activity - optional" on page 679
- "Check the implementation" on page 679

1. ## Prerequisite - create a runtime file

   Create a C# project that implements your activity's actions in the `addins\CustomerAddins\<addin_name>` folder. For task details, see "How to Create a Runtime File" on page 679.

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 details about each of the elements, see "Resource Element " on page 664.

```
<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 665.

d. Add properties to the relevant sections. For details and examples, see "Property Definitions" on page 667.

   - **GeneralProperties.** Properties displayed in the Properties pane'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 Properties pane's **Input/Checkpoints** tab, in the **Input** pane.
   - **OutputProperties.** Properties displayed in the Properties pane's **Input/Checkpoints** tab, in the **Checkpoints** pane.

e. Specify any external resource files as described in "Resource Files " on page 672.

f. Close the file with the `</Resource>` tag.

```
</Resource>
```

For more details about the structure of the signature file, see "Signature Files " on page 663.

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 670.

```
<?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 670.

d. Define post-build tasks such as `resgen`, as described in "Resource Files " on page 672.

e. Compile the project and copy the DLL to the `<Installation_Folder>\addins\CustomerAddins\<addin_name>` folder.

For additional details, see "Runtime Files " on page 663.

4. **Provide a graphic for your activity - optional**

a. Copy an icon image for your activity into the `<Installation_Folder>\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 the application 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 Create a Runtime File

This section contains the following topics:

- "Set the status" on page 683
- "Compile the runtime file" on page 683

Note: You must compile the DLL with a **Target Framework** of Framework 4.0.

1. ## Add Using statements

   Provide the mandatory **using** statements. In your solution, you must also add a reference to the `.dll` filess. The `.dll` filess are located in the products installation's `/bin` folder. You must always add a reference to `HP.ST.Fwk.RunTimeFWK.dll`. If you are using internal logging, you must also add a reference to `log4net.dll`.

   The following example shows the **Using** statements in the sample `.cs` file.

   ```
   using HP.ST.Fwk.RunTimeFWK;
   // If you need to implement Internal Logging
   using log4net;
   ```

2. ## Specify the namespace and class

   Define the namespace and provide the activity's runtime code. The class you define for your custom activity must inherit from the `STActivityBase` class. For example:

   ```
   namespace ReportMessageActivitySample
   {
   [Serializable]
   public class ReportMessageActivitySample : STActivityBase
   {
   ```

3. ## Set the internal logging

   Use the 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. For example:

   ```
   /// <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 649.

4. ## 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**. For example:

```
/// <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;
}
```

5. ## Retrieve the property values

This section retrieves or sets the input property values. For example:

```
/// <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" (described on page 630) and link directly to the array element.

6. ## Define events

Define one or more custom events, that you will invoke later. For example:

```
public event EventHandler CustomerEvent;
private void InvokeCustomerEvent(EventArgs MyArg)
{
EventHandler handler = this.CustomerEvent;
if (handler != null)
{
handler(this, MyArg);
}
}
```

7. ## Execute the step

Execute the step and send the runtime information to the log. Use the **STExecutionResult** data type and its **ExecuteStep** function defined in the **STActivityBase** class. For example:

```
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();
```

```
}
```

8. ## 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. For example:

```
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);
}
```

9. ## Compile the runtime file

   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 reference the `.dll` from the signature file, and the signature file from the addin file.

Reference

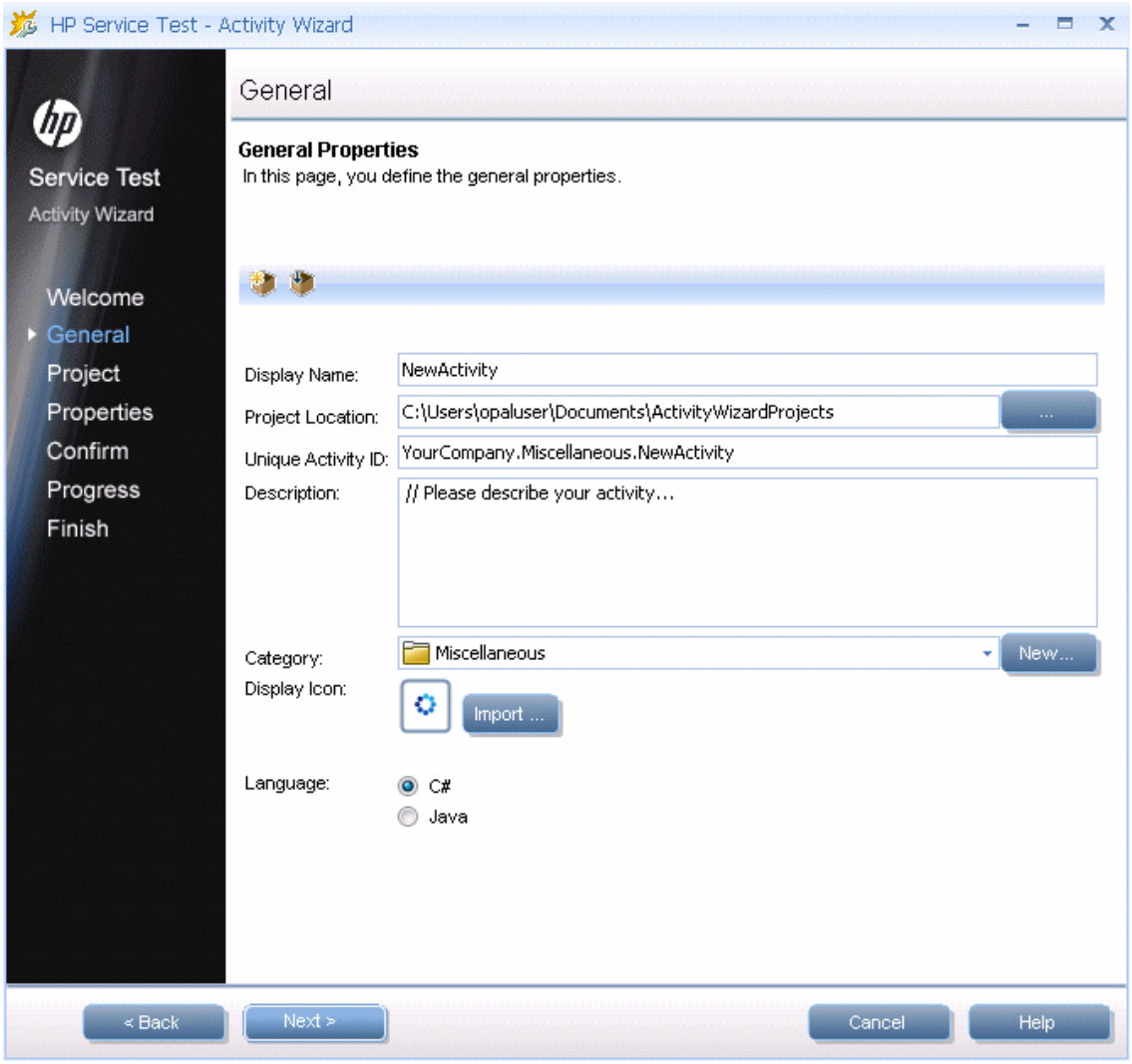
Activity Wizard

This wizard enables you to create new custom activities in the **Toolbox** pane.

To access	**Start > All > Programs > HP Software > HP Service Test > Tools > Activity Wizard** **Note:** For details on accessing Service Test tools and files in Windows 8, see "Accessing Service Test in Windows 8 Operating Systems" on page 33.
Wizard map	This wizard contains: **Welcome** > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "
See also	• "Creating New Activities - Overview " on page 662 • "How to Manually Create a Custom Activity in C#" on page 676 • "How to Use the Wizard to Create a Custom Activity - C#" on page 673

General Properties Page

This wizard page enables you to set the General type properties for the activity.



Important information	General information about this wizard is available here: "Activity Wizard" on previous page.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

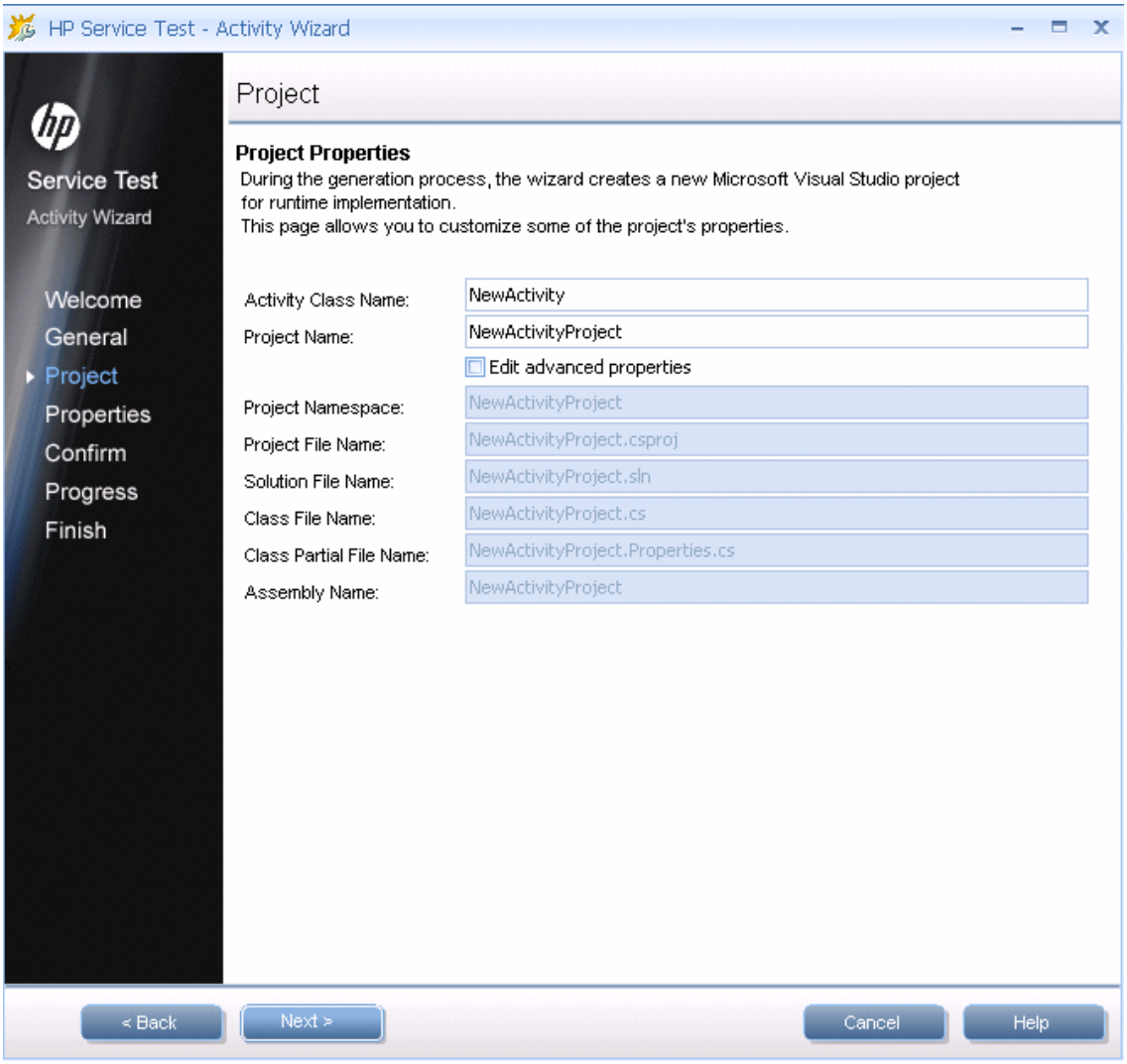
User interface elements are described below:

UI Elements	Description
	**Reset Values.** Resets all of the values to their original defaults and discards all changes that you made.
	**Open Wizard Project.** Opens a Visual Studio activity project that you created earlier.

UI Elements	Description
Category	The category under which to deploy the activity. A drop down list contains all of the built-in or previously created categories. **Clicking on New** opens the Add New Category dialog box.
Description	A meaningful description that will appear in the **Toolbox** pane's hint area when you select the activity.
Display Icon	An icon image for the activity for the **Toolbox** pane and canvas. Click **Import** to upload a new image file for the icon. The icon size should be 16 x 16 pixels.
Display Name	The name of the activity as it will appear in the canvas and **Toolbox** pane.
Language	The language in which to generate the code: `C#` or `Java`. **Note:** When you select **Java**, the wizard creates a Java file to which you can add the logic for your activity, and references to other required files. In addition, a Visual Studio project is created to manage the integration between the API Test and your Java activity.
Project Location	The location in which to save the project.
Unique Activity ID	A unique ID for the new activity. This ID can be used to reference the activity.

Project Properties Page

This wizard page enables you to set the properties for the activity project.



Important information	General information about this wizard is available here: "Activity Wizard" on page 684.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

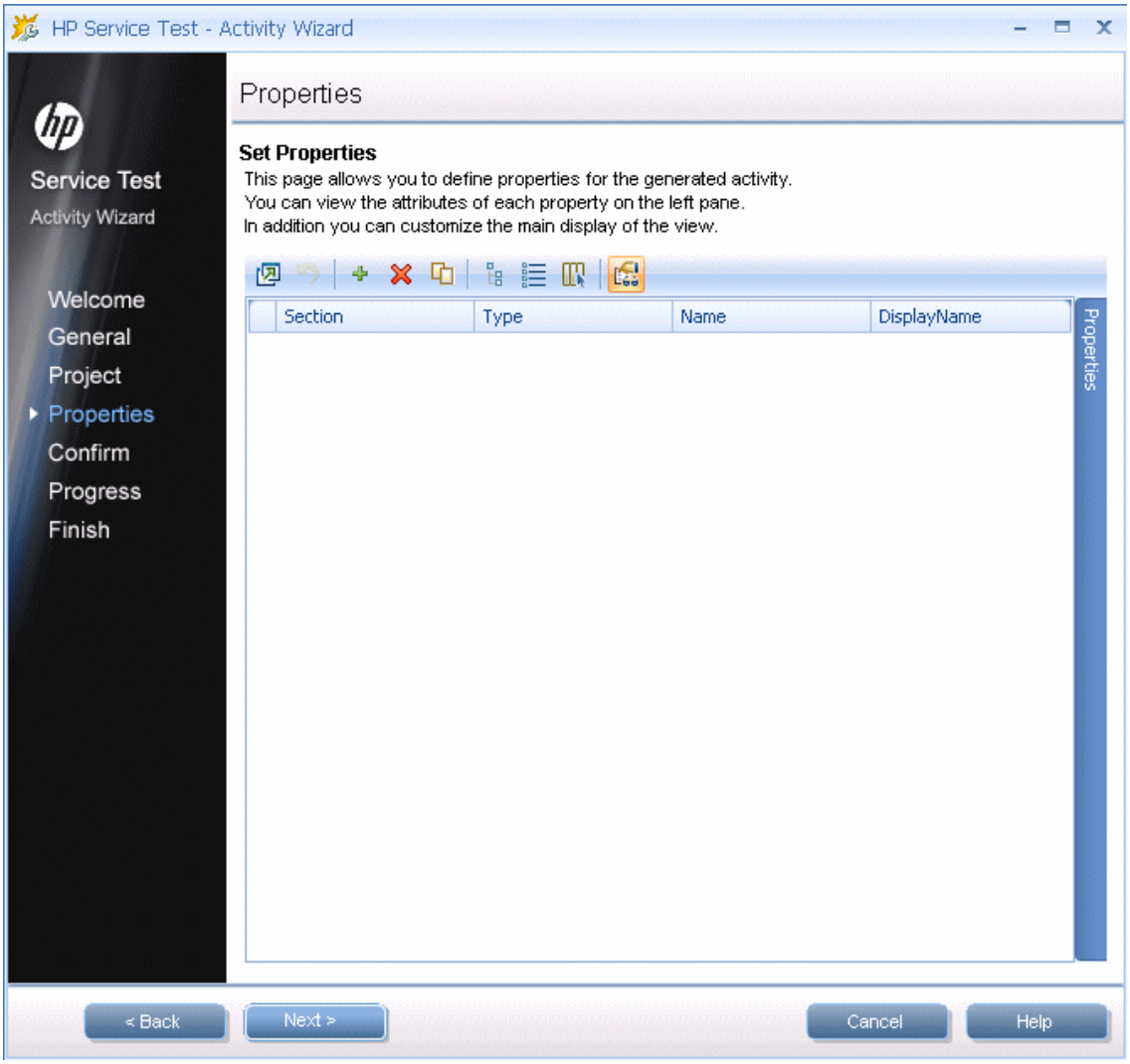
User interface elements are described below:

UI Elements	Description
Activity Class Name	A class name for the activity that will be generated.
Project Name	The name of the project.

UI Elements	Description
Edit advanced properties	Enables the manual editing of additional project properties.
Project Namespace	The namespace to create for the project.
Project File Name	The name of the project file. By default, this is the **Project Name** with a `.csproj` extension.
Solution File Name	The name of the solution file. By default, this is the **Project Name** with an `.sln` extension.
Class FIle Name	The name of the class file. By default, this is the **Project Name** with a `.cs` extension.
Class Partial FIle Name	The name of the class file. By default, this is the **Project Name** with a `.cs` extension.
Assembly Name	The name of the assembly file. This name will be associated with the `.dll` extension.

Set Properties Page

This wizard page lets you to define Input and Output properties for the activity and add new General properties.



Important information	General information about this wizard is available here: "Activity Wizard" on page 684.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

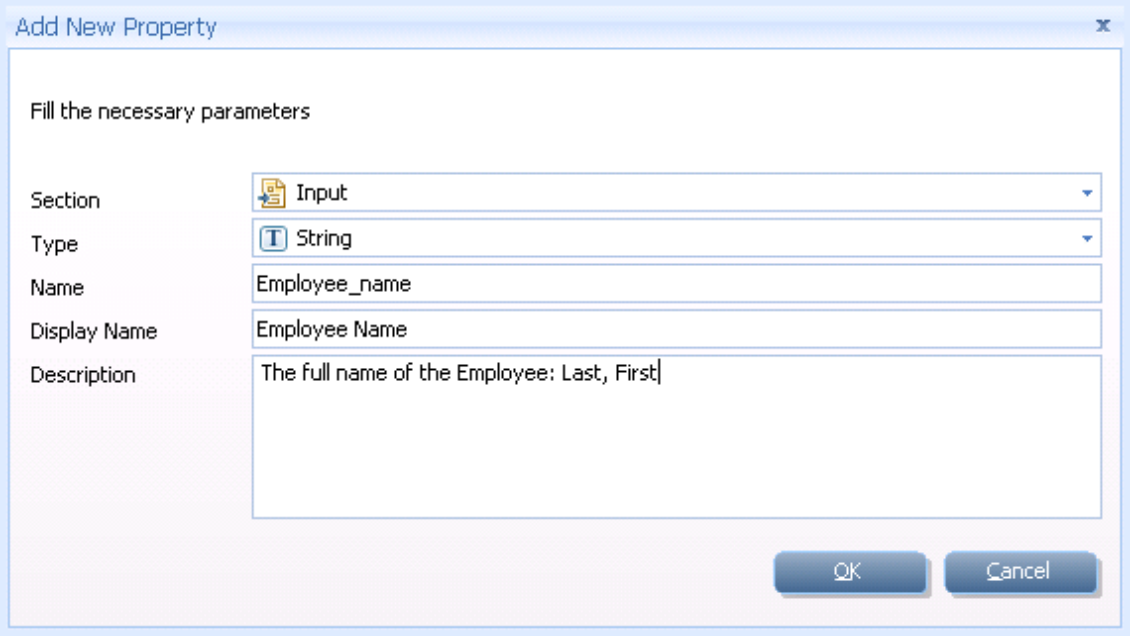
User interface elements are described below (unlabeled UI elements are shown in angle brackets):

UI Elements	Description
	Open in Separate Window. Opens a list of the properties in a separate window.
+	**Add Property.** Opens the Add New Property dialog box. For details, see "Add New Property" on next page.

UI Elements	Description
	**Remove Property.** Deletes the selected property from the activity.
	**Clone Property.** Makes a copy of the selected property.
	**Show Properties Tree.** Shows the properties in a tree hierarchy, by category—General, Input, and Output.
	**Show Properties List.** Hides the property tree hierarchy and shows all of the properties in a single grid.
	**Add/Remove Columns.** Opens the Add/Remove Columns dialog box enabling you to indicate which columns will be displayed in the Set Properties wizard page.
	**Show Properties Pane.** Opens the Properties pane in a dockable window. You can edit all of the property values— even ones that are not displayed in the wizard screen columns.
	**Hide Properties Pane.** Closes the Properties pane.
<property list>	A grid representation of the activities properties. **Tip:** • To show different or additional columns in the grid, click the **Add/Remove Columns** button and select the desired columns. • To change the order of the columns in the display, drag to column title to the desired location.

Add New Property

The Add New Property dialog box lets you create new properties for the activity.



To access	Do the following: 1. Open the Activity Wizard. For details, see "Activity Wizard" on page 684. 2. Advance to the **Set Properties** page. 3. Click the **Add Property** button in the toolbar .

User interface elements are described below:

Tree Elements (A-Z)	Description
Description	An optional description of the property, that will appear in the hint area when you select the property in the Properties pane.
Display Name	The name of the property as it will be displayed in the property list.
Name	The name of the property as it will be called within the test and by event handlers.
Section	The section in which to add the property: **Input**, **Output** (Checkpoint), or **General**.
Type	The data type of the property, such as **String**, **Int**, and so forth.

Confirm Page

This wizard page shows you a summary of your settings before generating the activity.

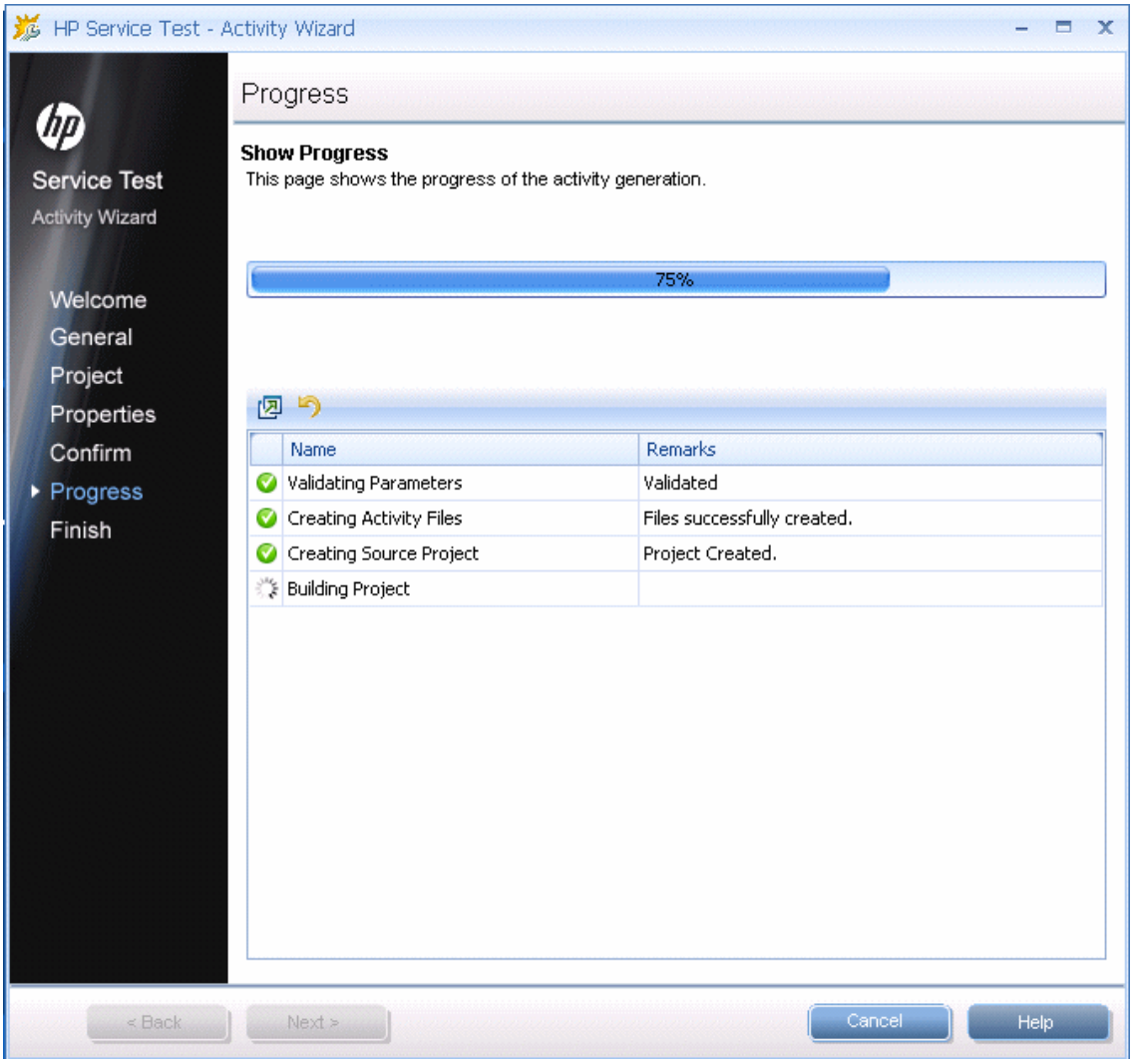
Important information	General information about this wizard is available here: "Activity Wizard" on page 684.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

User interface elements are described below:

UI Elements	Description
<summary text>	A message indicating that the wizard is ready to begin the generation.

Progress Page

This wizard page shows you the generation progress.



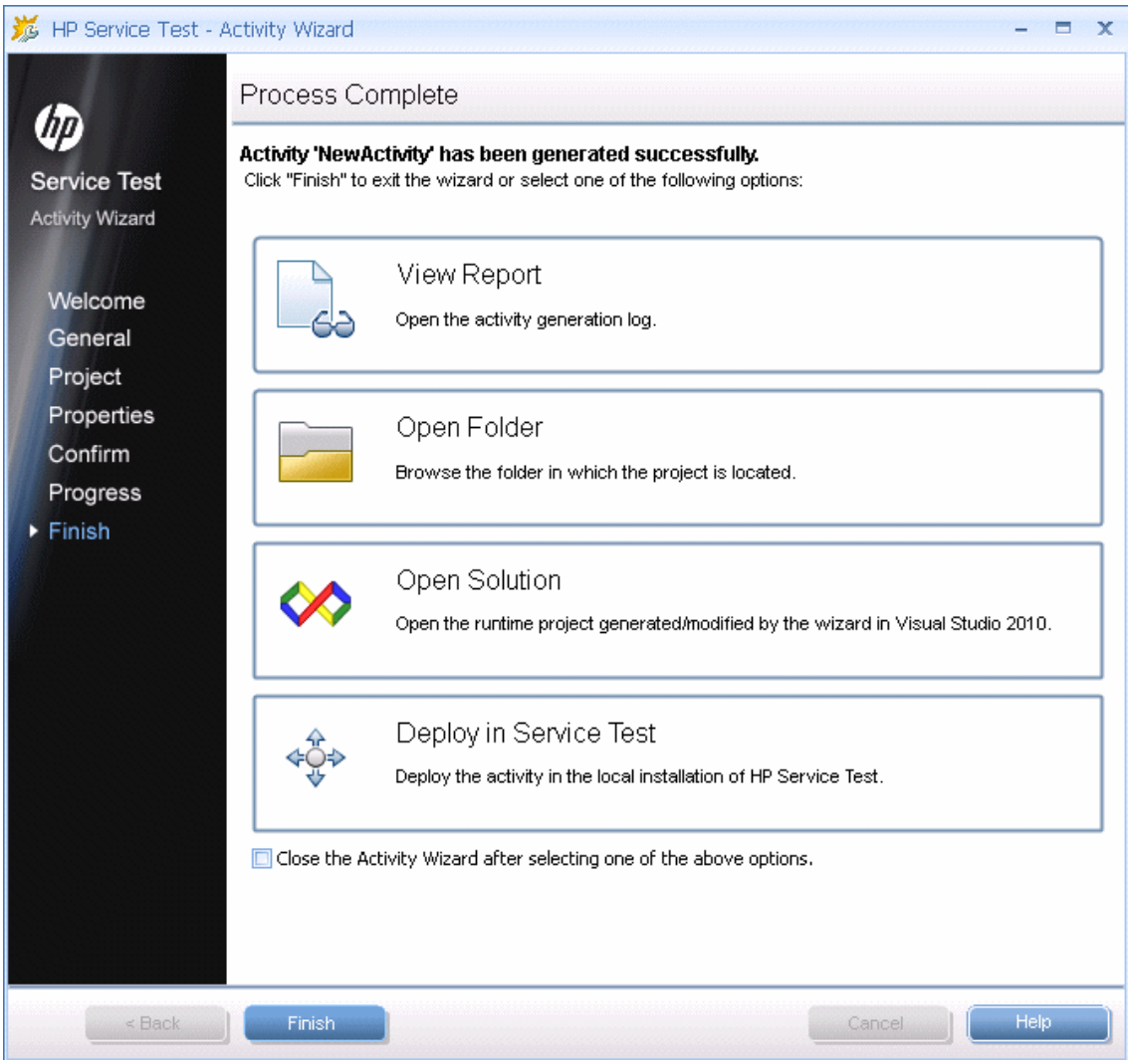
Important information	General information about this wizard is available here: "Activity Wizard" on page 684.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

User interface elements are described below:

UI Elements	Description
<progress information>	A log indicating progress of the generation.

Finish Page

This wizard page enables you to view the generation log, open the solution and its folder, and deploy the activity.



Important information	General information about this wizard is available here: "Activity Wizard" on page 684.
Wizard map	This wizard contains: Welcome > "General Properties Page" > "Project Properties Page" > "Set Properties Page " > "Confirm Page " > "Progress Page " > "Finish Page "

User interface elements are described below:

UI Elements (A-Z)	Description
View Report	Opens a log of the generation process. The log file is located in the `%temp%\ ActivityWizard\Reports` folder. Its title contains the generation date, `WizardLog_#datetime#.log`.
Open Folder	Opens the folder in which the activity files were generated. This folder is the Output directory you specified in the wizard. For example, you can open this folder to locate the java files to which you can add your custom code. The default subfolder is `hp\st\ext\java`.
Open Solution	Opens the activity's solution file in Visual Studio 2010. Use this option to edit and compile the source code.
Deploy in Service Test	Deploys the activity in Service Test by adding it to the **Toolbox** pane.
Close the Activity Wizard after selecting one of the above options	Closes the wizard screen automatically after selecting the **View Report**, **Open Folder**, **Open Solution**, or **Deploy in Service Test** options.

Troubleshooting and Limitations - Extensibility

This section describes troubleshooting and limitations for creating and using custom activities.

- The following error may occur if you place the signature file beneath a sub-folder of the **addin** folder.

  ```
  ServiceTest was unable to drag and drop the activity: Type
  'http://hp.vtd.schemas/types/v1.0:GeneralPropertiesType' is not
  declared.
  ```

  Workaround: Modify the relative path of the Types schema. For example:

  ```
  .schemaLocation="../../dat/schemas/Types.xsd".
  ```

- If you modify the activity structure in the signature file, you will be unable to open tests using that activity. To modify an activity structure, create a new activity with the new structure, replace all of the test steps using the old activity, and then remove the old activity implementation.

Part 7: Appendix

Chapter 33: Where's My UI? (Changes from Previous Versions)

This chapter describes changes that have been made to the user interface betweenService Test11.20 and this version of Service Test:

Pane Changes

The table below lists the panes whose names in this version of Service Test are different fromService Test 11.20. The functionality of the pane remains similar.

Previous Pane Name	Current Pane Name	Additional Information
Data Explorer	**Data**	For details, see "Data Pane" on page 96.
Property Sheet	**Properties**	For details on the Properties pane, see "Properties Pane" on page 184.
Tests	**Solution Explorer**	For details on the **Solution Explorer** pane, see "Solution Explorer Pane Overview" on page 235.
Toolbox Palette	**Toolbox**	For details, see "Toolbox Pane Overview" on page 255.

Menu Command Changes

Some of the menu commands from Service Test 11.20 are relocated in this version of Service Test, to improve product usability. Use this section to locate the commands that moved.

This section discusses the following menu changes:

File Menu Changes

The following table lists commands from theService Test 11.20 **File** menu and their location in this version of Service Test.

Service Test 11.20 Menu Option	Current Option	Additional Information
File > ALM/QC Connection	**ALM > (menu option)**	All menu command to manage an ALM connection have been moved to a stand-alone **ALM** menu. **Note:** The ALM Connection is also available by pressing the **ALMConnection** button  in the toolbar.
ALM/QC Version Control > Check Out		
ALM/QC Version Control > Undo Check Out		
ALM/QC Version Control > Check In		
ALM/QC Version Control > Version History		
ALM/QC Version Control > Baseline History		

Edit Menu Changes

The following table lists commands from theService Test 11.20 **Edit** menu and their location in this version of Service Test.

Service Test changes

Service Test 11.20 Menu Option	Current Option	Additional Information
Select All		This command is not relevant for this version of Service Test.
Find and Replace	**Search > Find** **Search > Replace**	
Settings > Show Run Test dialog box		This command is not relevant for this version of Service Test.
Settings > Show Results after Run	**Tools > Options > General** tab **> Run Sessions** node	Select the **View results when run session ends** option to enable this command.
Settings > Use Relative Paths in Service Test Steps	**Tools > Options > API Testing** tab **> General** node	Select the **Use relative paths to call assets** option to enable this command.
Settings > Activity Repositories	**Tools > Options > API Testing** tab **> General** node	Choose the Activity Repositories path in the **Activity Repositories** area of **General** pane.

View Menu Changes

The following table lists commands from the Service Test 11.20 **View** menu and their location in this version of Service Test.

Service Test 11.20 Menu Option	Current Option	Additional Information
Tests	**View > Solution Explorer**	
Data Window	**View > Data**	
Property Sheet	**View > Properties**	
Window Theme > (suboptions)		This command is not relevant for this version of Service Test.

Test Menu Changes

The **Test** menu items from Service Test 11.20 are incorporated into other menus in this version of Service Test.

Service Test 11.20 Menu Option	Current Option	Additional Information
Test > Add Reference	**Design > Add Reference**	
Test > Enable Test for Load Testing	**Design > Enable Test for Load Testing**	Only available when HP LoadRunner is installed on the machine.
Test > Open Containing Folder	Context menu in document pane	Right-click on a test tab in the document pane and select **Open Containing Folder in Explorer**.
Test > View Test Results	**View > Last Run Results**	

Build Menu Changes

The **Build** menu from Service Test 11.20 is incorporated into the **Run** menu in this version of Service Test.

Service Test 11.20 Menu Option	Current Option	Additional Information
Build Solution	**Run > Compile submenu**	
Rebuild Solution		
Clean Solution		
Build <test name>		
Rebuild <test name>		
Clean <test name>		
Abort Build		

Service Test 11.20 Menu Option	Current Option	Additional Information
Set Configuration > Debug	T**ools > Options > API Testing tab > General node**	Select the appropriate option from the drop-down menu for **Compile as** option.
Set Configuration > Release		

Debug Menu Changes

The **Debug** menu fromService Test 11.20 is incorporated into the **Run** menu in this version of Service Test.

Service Test changes

Service Test 11.20 Menu Option	Current Menu Option
Run Test	**Run** submenu
Stop Process	
Break	
Continue Debugging	
Step Over	
Step Into	
Step Out	
Toggle Breakpoint	

Help Menu Changes

The following **Help** menu options fromService Test 11.20 are changed in this version of Service Test.

Service Test changes

Service Test 11.20 Menu Option	Current Menu Option	Additional Information
Index		Not available in this version of Service Test.
Search		
Contents		
User Guide	**Help > HP Service Test Help**	
HP Service Test Web Site	**Help > Useful Links > Product Page**	

Toolbar Changes

The sections describe the locations of toolbar buttons or toolbar icons fromService Test 11.20 that changed.

The following table shows the new icons for Service Test 11.20 toolbar buttons, as well as the menu options you can use for buttons that are no longer available.

Service Test 11.20 toolbar button		Current Location		Additional Information
Open Test		**Toolbar button:** Open		This toolbar button also provides a drop-down list of document types to open.
Add New Test		**Toolbar button:** Add		This toolbar button also provides a drop-down list of document types (both new and existing to add to a solution).
ALM/QC Connection		**Toolbar button:** ALM Connection		
Enable Test for Load Testing		**Menu command:** Design > Enable Test for Load Testing		Only available when HP LoadRunner is installed on the machine.
Default Zoom		**Button on canvas**		
Zoom In		**Button on canvas**		
Zoom Out		**Button on canvas**		


Java™
COMPATIBLE