HP Service Test

for the Windows operating systems

Software Version: 11.10

Tutorial

Document Number: T6553-90007
Document Release Date: February 2011
Software Release Date: February 2011

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

Trademark Notices

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

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

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

UNIX® is a registered trademark of The Open Group.

Documentation Updates

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

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

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

http://h20230.www2.hp.com/selfsolve/manuals

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

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

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

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

Support

Visit the HP Software Support web site at:

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

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

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

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

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

To find more information about access levels, go to:

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

Table of Contents

Welcome to the Service Test Tutorial **7**
Tutorial Lessons 7

Chapter 1: Introducing HP Service Test **9**
What is SOA? 9
Why should you automate SOA testing? 10
Understanding Service Test Terminology 11
What is the sample application for this tutorial? 12
How do I invoke the application? 13

Chapter 2: Build a Simple Test **15**
How do I create a new test? 16
How do I work with the Service Test panes? 17
How do I create a test step? 18
How do I connect test steps? 22
How do I map data from multiple sources? 24
How do I data drive the step? 26

Chapter 3: Test a Web Service **31**
How do I import a Web service? 32
How do I build a Web service test? 34
How do I integrate data into a test? 39
How do I use multiple data sources and custom code? 44

Chapter 4: Test REST Services **49**
How do I model a REST service? 50
How do I create a simple test for a REST service? 51
How do I create a template for a REST method? 54
How do I create a REST test using table data? 56

Welcome to the Service Test Tutorial

Welcome to the HP Service Test Tutorial, a self-paced printable guide, designed to lead you through the process of creating tests for Web services, REST services, and other GUI-less applications.

At the conclusion of this tutorial, you will be ready to design, run, and monitor a simple test on your own application. It is recommended that you move through the tutorial in the order in which the information is presented.

Tutorial Lessons

The following lessons are included in this tutorial:

Lesson 1: Introducing HP Service Test: Familiarizes you with the HP Service Test.

Lesson 2: Build a Simple Test: Introduces you to the basic creation of a test through dragging and dropping activities into the canvas. It also shows you how to data-drive the test with external data.

Lesson 3: Test a Web Service: Shows you how to create a test for a Web service and how to check the results.

Lesson 4: Test REST Services: Shows you how to model and create tests for REST services.

1

Introducing HP Service Test

HP Service Test is an extensible framework for the construction and execution of functional tests of headless systems, systems that do not have a user interface. This document describes how to get started with HP Service Test and create your first tests. It also introduces the major features in the product and how to incorporate them into your tests.

You create a test by dragging activities into a canvas. You set parameters, assign values, and run your test. You can check for expected values with the built-in checkpoint mechanism or through custom C# code.

What is SOA?

In recent years, outsourcing and advanced business needs required companies to collaborate and share information. In addition, with the rising number of mergers and acquisitions, businesses struggled to find a way to share data. If, prior to a merger, two businesses maintained their own proprietary computer systems, after the merger, the sharing of data could be very time consuming and costly.

For these issues, several vendors developed technologies to handle B2B (Business-to-Business) communication. Examples of B2B technologies are RMI, COM, CORBA, EDI, and Web services. In addition to allowing the systems to link up to one other, the technologies also handle permissions to allow networking with each other like an Intranet.

Web services are self-contained applications that can run across the Internet on a variety of platforms. They use XML and Simple Object Access Protocol (SOAP) as the base language, making it a developer-friendly solution. Since Web services are based on a set of standardized rules and specifications, they are more portable than other technologies.

SOA (Service Oriented Architecture) is an architectural style that lets multiple software services interact. A service is a unit of work done by a service provider as specified by a consumer. SOA requires that services interact with one another, but without interdependencies. The services are autonomic and loosely coupled, only requiring that they retain an awareness of one other, but no dependencies.

SOA systems are primarily based on Web services. In Web services, a client submits a request and the Web server provides a response using the SOAP protocol. HP Service Test lets you check the behavior of your Web services in an automated manner.

Why should you automate SOA testing?

Automated SOA Testing is a discipline that leverages products and processes to reduce the risks of application upgrades or deployment of new services. At its core, automated testing is about applying production workloads to pre-deployment systems while simultaneously measuring system performance and end-user experience. A well-constructed performance test answers questions such as:

- Does the service respond quickly enough for the intended users?
- Will the server respond with the correct values?
- How will the service handle exceptions and illegal values?
- Is the service stable under expected and unexpected user loads?

By answering these questions, you can design a test more effectively. An effective automated testing process helps you make more informed release decisions, reduces system downtime, and prevents availability problems.

Understanding Service Test Terminology

You may encounter the following terms when working with SOA testing:

HTTP	**Hypertext Transfer Protocol**, a communications protocol used to transfer or provide information over the World Wide Web. Users utilize HTTP to publish and retrieve HTML pages.
JMS	**Java Message Service**, a Java-based Message Oriented Middleware API for sending messages between two or more clients.
REST	**Representational State Transfer.** A style of software architecture for distributed systems.
SOAP	**Simple Object Access Protocol**, or **Service Oriented Architecture Protocol**, a protocol for exchanging structured and typed information between peers in a distributed environment using XML sent over HTTP or JMS. SOAP lets you serialize distributed components into XML documents for transport and deserialize them upon reaching their destination. This promotes interoperability between components that are based on different technologies.
Test	The steps that a user performs are described in a **test**. A test emulates the actions of real users using the application.
UDDI	**Universal Description, Discovery and Integration**, a platform-independent, online registry listing businesses worldwide. It is often used as a database for public Web services.
Web services	Standardized, Web-based applications using the XML, SOAP, WSDL and UDDI standards over an Internet protocol. XML is used to tag the data, SOAP is used to transfer the data, WSDLs describe the services and UDDIs list them.

WSDL	**Web services Description Language**, an XML-based language designed to describe a Web service. The WSDL document provides essential information about the Web service, such as its ports and operations, required for implementation.
XML	**Extensible Markup Language** is a general-purpose markup language. It is called extensible because it lets you define your own tags. It enables the sharing of structured data across different information systems, especially over the Internet. XML is used both to serialize data and encode documents.
XSD	**XML Schema Definition** files contain a set of rules that formally define the hierarchical structure of an XML document. An XML document must comply with these rules and constraints in order for parsers and processors to deem it valid. WSDL documents can reference an external schema or it can contain an embedded one.

What is the sample application for this tutorial?

This tutorial is based on a sample application included with Service Test—the **HP Flights Service**. It is available as both a Web service and a REST service.

The sample application works with a database of flight reservations. You can retrieve flights for specific destinations, create customer orders, update reservations, or delete them.

For details about the service's methods and operations, type "help" into the Sample Application's window.

Note: It is recommended to have Microsoft Excel installed on the Service Test machine to enable Excel functionality for Service Test data sources.

How do I invoke the application?

As a first step, you will invoke the sample flight application, so that it will be available for your test.

To start the HP Flights service:

1. Select **Start** > **(All) Programs** > **HP Service Test 11.10** > **Sample Application**. A Command window opens indicating that the application is available.
2. If the window issues a message that the default port 24240 is unavailable, edit the **<installation_directory>SampleApplication\HPFlights_Service.exe.config** file in a text editor. In the **appSettings** section's, replace the 24240 Port key with a valid one.
3. Minimize the Command window.

Important: Do not close the Command window, as this will stop the service.

Where To Go From Here

Now that you have invoked the application, you can begin creating tests for your headless applications using the Flights application. The following lessons will walk you through the process of creating a test for basic activities, Web, and REST services.

2

Build a Simple Test

This lesson will guide you through the steps of creating tests with simple actions.

This lesson contains the following sections:

- How do I create a new test?
- How do I work with the Service Test panes?
- How do I create a test step?
- How do I data drive the step?
- How do I connect test steps?
- How do I map data from multiple sources?

How do I create a new test?

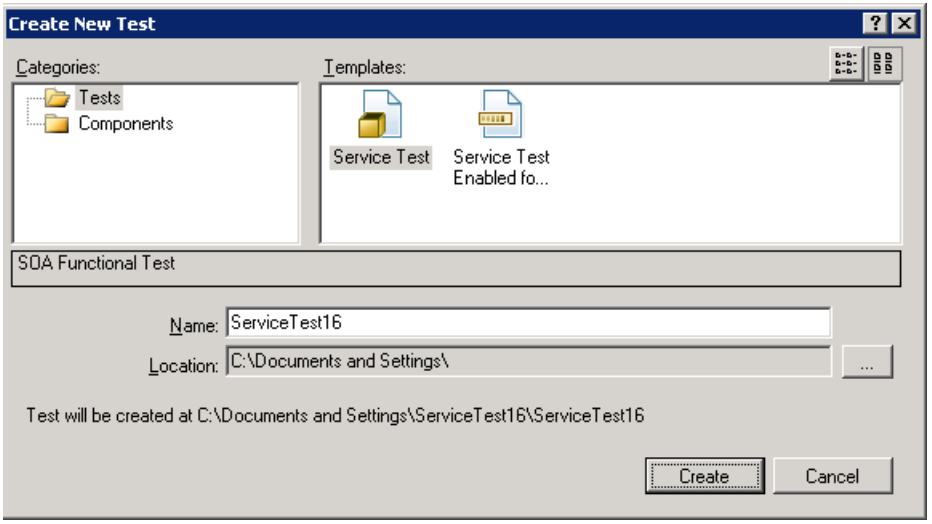
Service Test's canvas is the central console in which you build and run your test.

1 Open HP Service Test.

Choose **Start** > **(All) Programs** > **HP Service Test 11.10** > **HP Service Test 11.10**. The Start Page opens.

2 Create a new test.

Click **File** > **New** > **Test**. The Create New Test dialog box opens. Select the **Service Test** template. On machines with an installation of HP LoadRunner, the **Service Test Enabled for Load Testing** template is also visible.



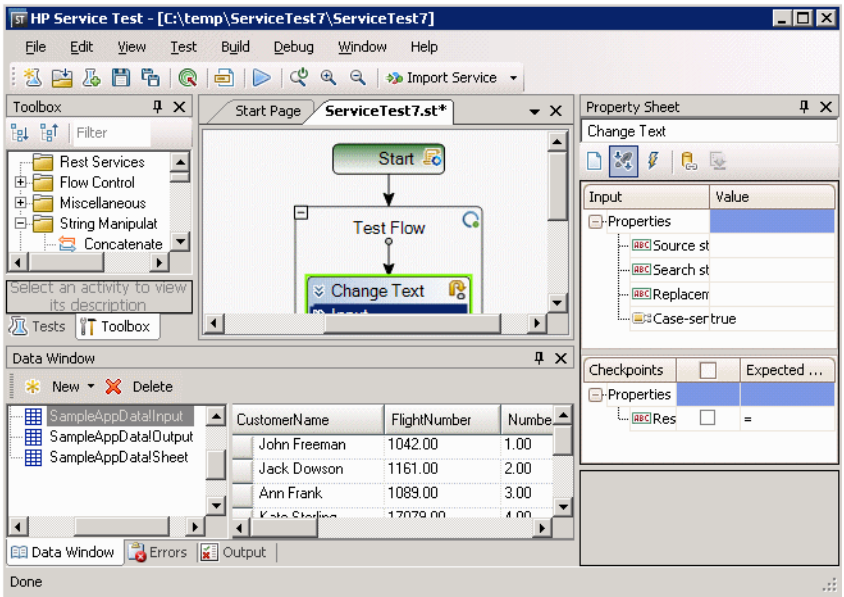
3 Generate the new solution.

In the **Name** box, replace the default name with **BasicTest**, and click **Create**. An empty test opens, with a canvas showing the **Start**, **Test Flow**, and **End** sections.

The **Test Flow** is the section of the test containing the activities whose functionality you want to test. The **Start** section is ideal for defining items that you want to initialize before the test, such as test variables.

How do I work with the Service Test panes?

Most of the Service Test panes are floating, dockable windows. To show the default panes in their original positions, select **Window** > **Reset Window Layout**.



Service Test contains the following primary windows:

- **Toolbox.** (upper left) Shows the list of the available activities, services and operations. From this pane, you drag activities into the canvas
- **Canvas**. (upper middle) The work area in which you organize the test steps.
- **Property Sheet**. (upper right) A multi-view window that lets you view and set properties for the step selected in the canvas. The common views are **General**, **Input/Checkpoints**, and **Events**. To change a view, click on one of the buttons in the Property Sheet's toolbar.
- **Data Window.** (bottom) The data to use with the test—either an imported Excel, XML, or database data source, or a manually defined table.
- **Error and Output.** (bottom) Tabs that provide information about the test run, such as output messages and errors.

How do I create a test step?

You create test steps by dragging activities from the toolbox into the canvas.

In this section you will create a simple test step to illustrate the use of the toolbox and the Service Test panes.

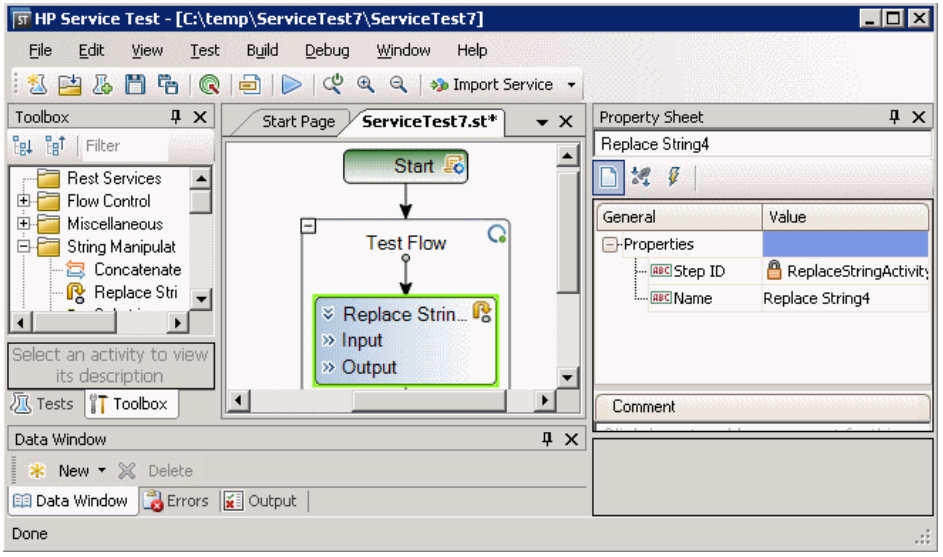
To create test steps

1 Locate the Replace String activity.

In the **Toolbox** palette, expand the **String Manipulation** category and select **Replace String**.

2 Create a step.

Drag the **Replace String** activity into the canvas and drop it on the arrow inside the **Test Flow**. This activity searches for text within a specific string, and replaces it with new text.



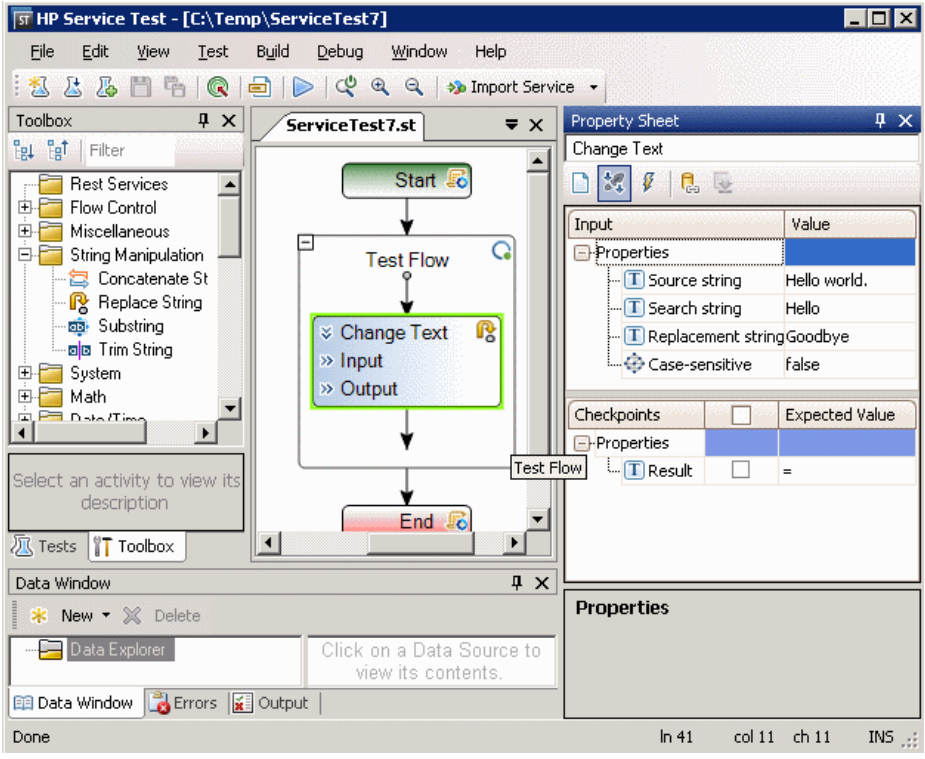
3 Change the step's display name in the General view.

Select the **Replace String** step in the canvas and click in the Property Sheet pane. In the Property Sheet, click the **General** view button. In the **Name** row, type **Change Text** and press ENTER. This changes the step name in the canvas.

4 Set the input properties.

In the Property Sheet, click the **Input/Checkpoints** button. Enter the following values:

- **Source string:** Hello world.
- **Search string:** Hello
- **Replacement string:** Goodbye
- **Case-sensitive:** false



5 Run the test.

Click the **Run Test** button or press F5 to compile and run the test. In the Run dialog box, select the **Temporary run results folder** option. Select the **Don't show this dialog again** option in the lower section of the dialog box. Click **OK**.

Tip: To show the Run dialog box again, toggle the option in the **Edit** > **Settings** menu.

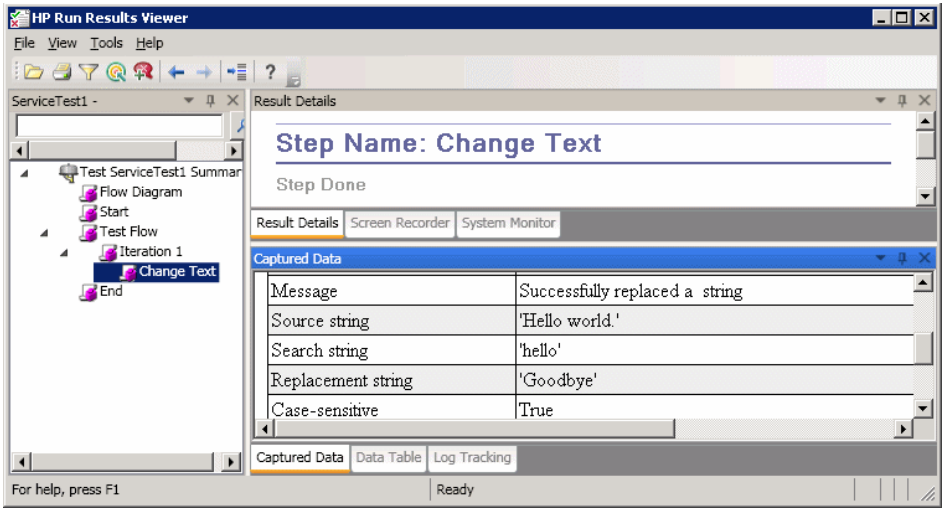
---

6 View the results.

If the Run Results Viewer is not in focus, click the **HP Run Results Viewer** tab on your machine's toolbar to bring it into focus.

Click the parent node and choose **Expand All** from the right-click menu. Click the **Change Text** node. View the source and replacement strings and note the result string, **Goodbye world**. This is in fact the expected string—the test passed.

When you are finished reviewing the results, close the Run Results Viewer.



7 Set a checkpoint.

In the previous step, you manually viewed the output to check if the result matched the expected value. Checkpoints allow you to see whether the action was successful without having to manually check the result. Checkpoints are the means to validate the test—a success or failure is determined by its checkpoints.

Return to the Property Sheet (right pane) and ensure that the **Input/Checkpoints** view is displayed. Click in the **Checkpoints** section and select the check box in the **Results** row to enable the checkpoint. In the **Expected value** column, type the expected string, **Goodbye world**.

Run the test again. In the Run Results Viewer, expand the nodes, and note the checkmark. This indicates that the checkpoint passed since the result matched the expected value.

When you are finished reviewing the results, close the Run Results Viewer.

How do I connect test steps?

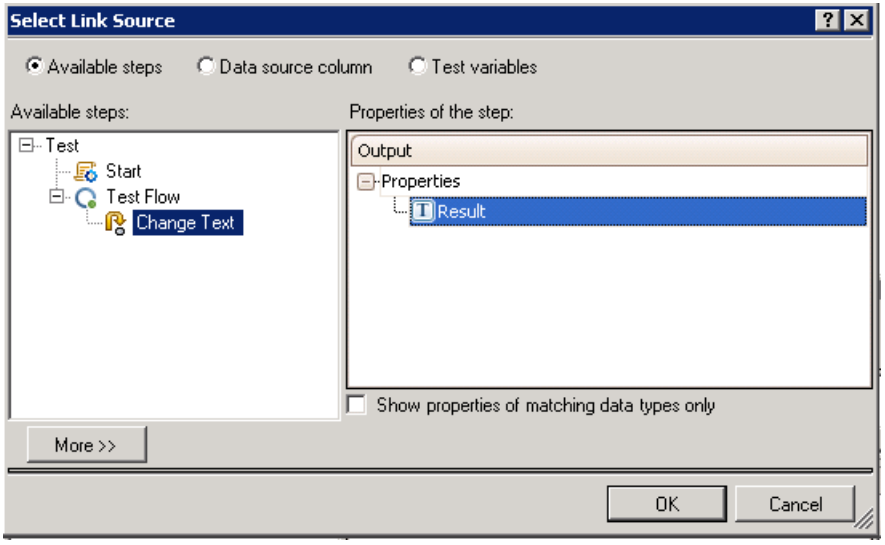
In this section, you will use the output of one step, as input for another.

1 Add a Concatenate String step.

In the **Toolbox**, expand the **String Manipulation** category and select **Concatenate String**. Drag the activity into the canvas and drop it below the Change Text step in the Test Flow. This activity concatenates two strings.

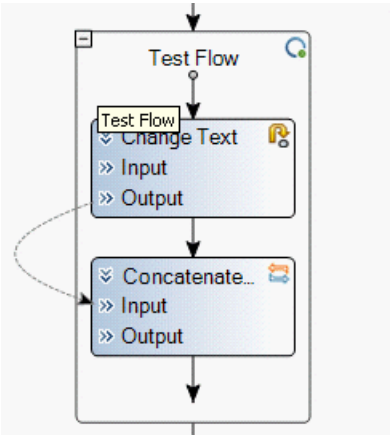
2 Set the prefix.

In the canvas, select the **ConcatenateString** step. In the Property Sheet, open the **Input/Checkpoints** view. In the upper pane, the **Input** section, move the mouse into the **Value** cell of the **Prefix** row. Click the **Link to a data source** button. The Select Link Source dialog box opens.



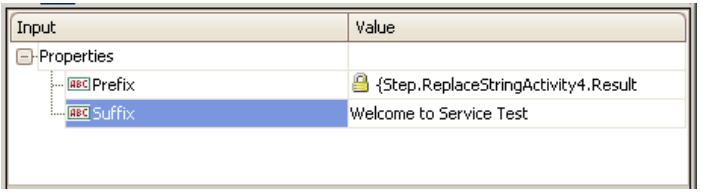
3 Map the data.

In the Select Link Source dialog box, select the **Available steps** option. Select the **Test Flow** > **ChangeText** node. In the right pane, double-click the **Results** node. The canvas now reflects that data is moving from **Change Text** to **ConcatenateString**.



4 Configure the suffix.

Type the text: **Welcome to Service Test**.



5 Run the test.

Click the **Run** button or press F5 to run the test.

6 View the report.

Expand the Run Results tree and select the **ConcatenateStringsActivity** node. The report shows the result of the concatenated strings: **Goodbye World.Welcome to Service Test.**

When you are finished reviewing the results, close the Run Results Viewer.

How do I map data from multiple sources?

Using the Select Link Source dialog box, you can link to one or more of the following data sources to provide input values: **Available steps**, **Data source column**, and **Test variables**. In the above section, you typed in data manually for one value, and used the **Available steps** source for another value.

Service Test lets you use multiple data sources to construct compound expressions. In this section, you will use the Select Link Source dialog box to create an expression for the **Suffix** property that uses both manual entry and automatic values from the **Available steps** option.

1 Set the prefix.


In the canvas, select the **ConcatenateString** step. Open the **Input/Checkpoints** view in the Property Sheet. Click in the **Value** cell of the **Prefix** row and click the **X** to clear the contents. Type a new prefix Hello world.

2 Open the Select Link Source dialog box.

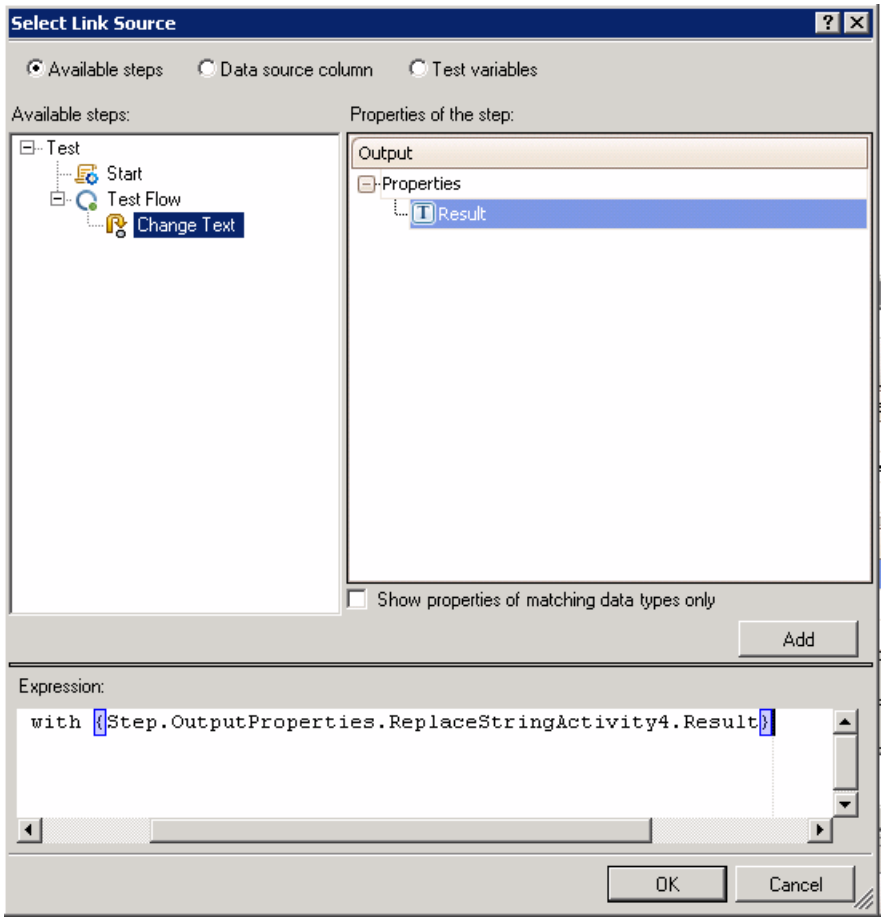

Click in the **Value** cell of the **Suffix** row and click **X** to clear its contents. Click the **Link data to source** button. The Select Link Source dialog box opens.

3 Edit the suffix.

In the Select Link Source dialog box, click the **More** button to expand it. In the **Expression** box, type was replaced with, adding a space before and after the phrase for readability.

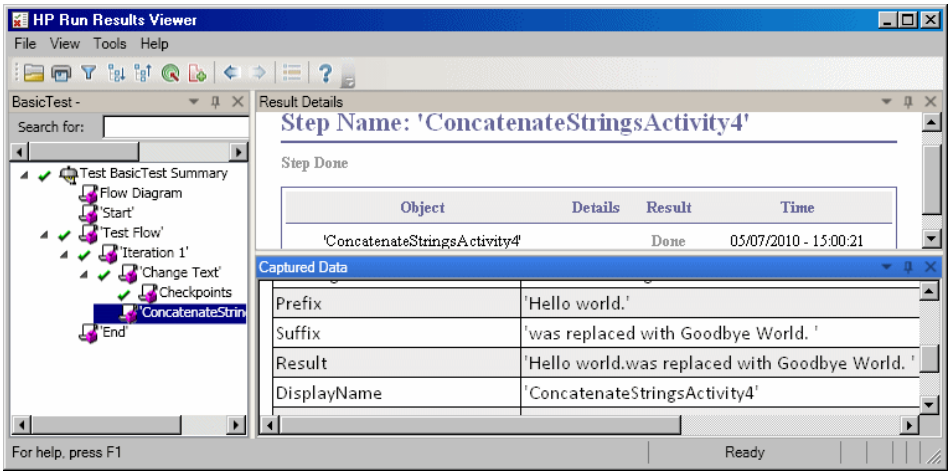
4 Add another source.

Select the **Available steps** option and select the **Change Text** node in the left pane. Select the **Result** node in the right pane, and click **Add**. The **Expression** box shows both sources.



5 Run the test and view the report.

Click the **Run** button to run the test. The report shows the result of the concatenated strings.



How do I data drive the step?

Data driving implies the assigning of data to test steps from an external source. The goal of data driving is to run the same business process with different values. It allows you to check your application in different scenarios, by modifying only the data tables.

To data drive test steps

1 Data drive the input arguments.

Select the Change Text step in the canvas. Open the **Input/Checkpoint** view in the Property Sheet, and click the **Data Drive** button. The Data Driving dialog box opens.

2 Specify a data provider.

In the Data Driving dialog box:

a Select an **Excel** type of Data provider.

b Enable data driving for **Both Input and Checkpoints**.

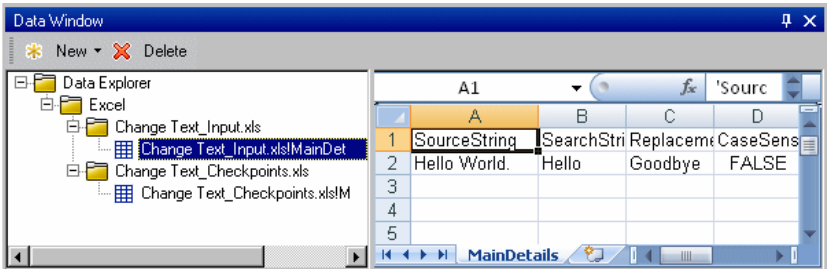
c Clear the **Configure 'Test Flow' as a ForEach loop using the new data source** option, which repeats the Test Flow according to the number of data rows. You will manually set the number of iterations in a later step.

d Click **OK** to close the Data Driving dialog box.

e Confirm the data driving popup message. Service Test replaces the constant values with the new expressions, **{DataSource.Change Text_Input.xls!MainDetails.SourceString}**.

3 View the Data Window.

Make sure the Data Window is visible. If not, choose **View** > **Data Window**. Expand the **Change Text_input.xls** node and select the **Change_Text_Input.xls!MainDetails** node. Service Test created a data table with a column for each input property, and one row of values corresponding to the values, **Hello World**, **Hello**, **Goodbye**, and **FALSE** (or empty check box for installations without Excel) that you entered earlier.



4 Add new data.

Add two additional rows to the table. Make sure to copy the text exactly, including punctuation where included.

SourceString	SearchString	ReplacementString	CaseSensitive
Hello world.	Hello	Goodbye	FALSE
I like eating broccoli.	broccoli	ice cream	TRUE
10 is the latest version of ST.	10	11	FALSE

5 Add checkpoint values.

Expand the **Change Text_Checkpoints.xls** node and select the **Change_Text__Checkpoints.xls!MainDetails** node. Add values to this column as shown below.

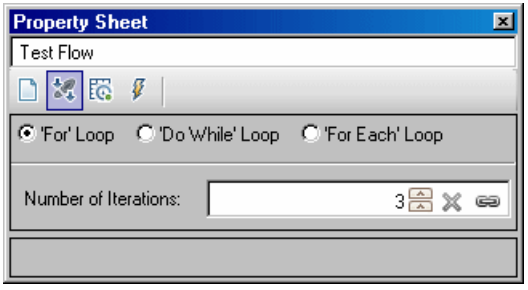
Note: In the third row, we will intentionally insert an exclamation point, !, in order to generate an error.

Result
Goodbye world.
I like eating ice cream.
11 is the latest version of ST!

6 Set the number of iterations.

The number of iterations is the number of times to repeat the step. We will set it to 3, corresponding to the number of rows of data in our table.

Return to the canvas and click inside the Test Flow frame—but not within a test step. Open the **Input/Checkpoint** view in the Property Sheet. Select **For Loop** and set the **Number of Iterations** to **3**.



7 Run the test and view the report.

Click the **Run** button or press F5 to compile and run the test. The test runs three times, using the three lines of data in the table.

After the test run, the Run Results Viewer opens. Expand the **Test Flow** node and drill down to the row with the red **X**, indicating a failed checkpoint. The checkpoint failed because the expected result contained an exclamation point, which was not present in the source string.

8 Correct the error and rerun the test.

Correct the data in the Data Window—in the third row of the **Results** column, for the checkpoint, replace the exclamation point with a period.

Run the script again and verify that you have no errors in the report.

Where To Go From Here

Now that you have learned to create simple test steps, you can create steps using Web services. The following lessons will walk you through the process of importing WSDLs and creating Web service tests.

3

Test a Web Service

Service Test allows you to create tests for your WSDL-based Web services.

This lesson contains the following sections:

- How do I import a Web service?
- How do I build a Web service test?
- How do I integrate data into a test?
- How do I use multiple data sources and custom code?

How do I import a Web service?

A WSDL file defines the operations in a Web service. To use the WSDL file, you import it into your test. This section shows you how to import the sample application's WSDL file.

1 Start the Sample Flight application.

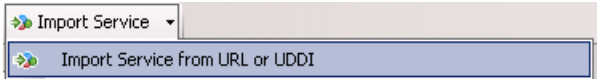
Make sure that the Flight Application service is available, as described in "How do I invoke the application?" on page 13.

2 Create a new solution.

Select **File** > **New** > **Test** and specify the name **WebServiceTest** for the new test. Click **Create**.

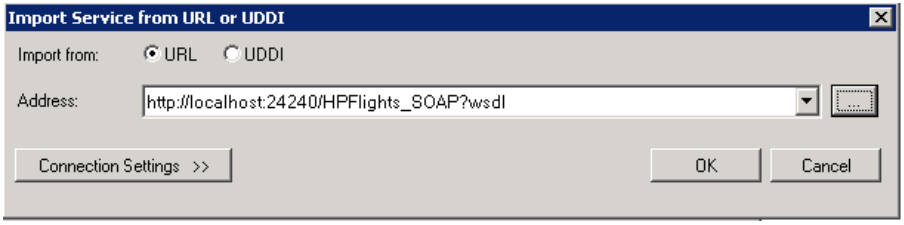
3 Open the Import Service dialog box.

Select **Import Service** > **Import Service from URL or UDDI** on the toolbar.



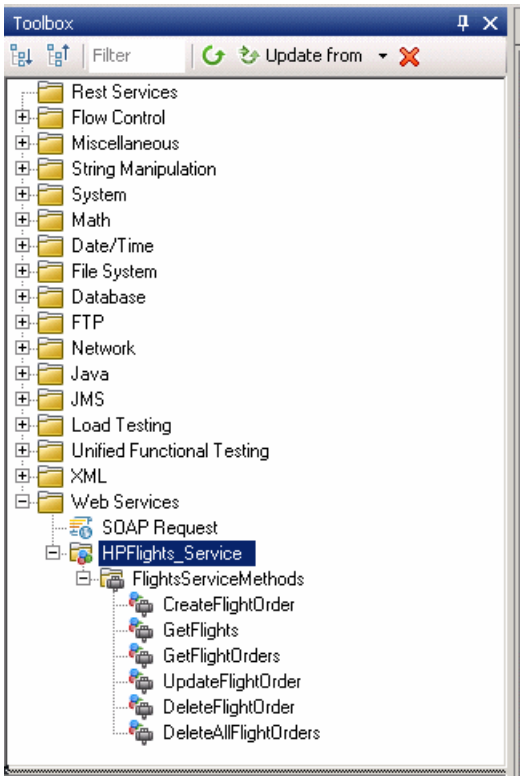
4 Specify an import source.

Select the **URL** option and specify the location **http://localhost:24240/HPFlights_SOAP?wsdl.** Click **OK**.



5 View the service's operations.

The import created a new branch of Web service operations in the **Toolbox**, under the **Web Services** category. Expand the node to view the operations.



How do I build a Web service test?

In this section, you will create a new flight order using the **HPFlights** Web service.

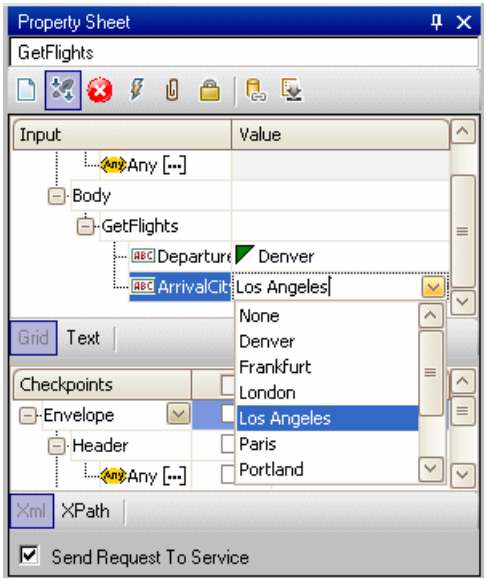
In order to create a flight order, you must first know the available flights. First you will run the **GetFlights** step that retrieves all of the flights to your destination. In the next step, you will use the first flight number returned, as input for the **CreateFlightOrder** step.

1 Create a GetFlights step.

Expand the **Web services** > **HPFlights_Service** node and drag the **GetFlights** activity into the Test Flow.

2 Assign values for DepartureCity and ArrivalCity.

Open the **Input/Checkpoints** view and expand the **Body** > **GetFlights** node. To select a city, click the arrow in the row to open a dropdown list. Choose Denver as the **DepartureCity** and Los Angeles for the **ArrivalCity**.



3 Create a CreateFlightOrder step.

Drag the **CreateFlightOrder** activity from the toolbox into the Test Flow, beneath the **GetFlights** step.

4 Set the values for the CreateFlightOrder step.

In the **Input/Checkpoints** view, expand the **Body** > **CreateFlightOrder** > **FlightOrder** node, and set the values for creating a flight order:

- **Class**—Select a class, such as Business from the dropdown list.
- **CustomerName**—any value
- **DepartureDate**—use the dropdown to open a calendar and select a date at least two days in the future.
- **FlightNumber**—leave blank for now. We will set it in the following steps.
- **NumberofTickets**—use the scroller to set any value.

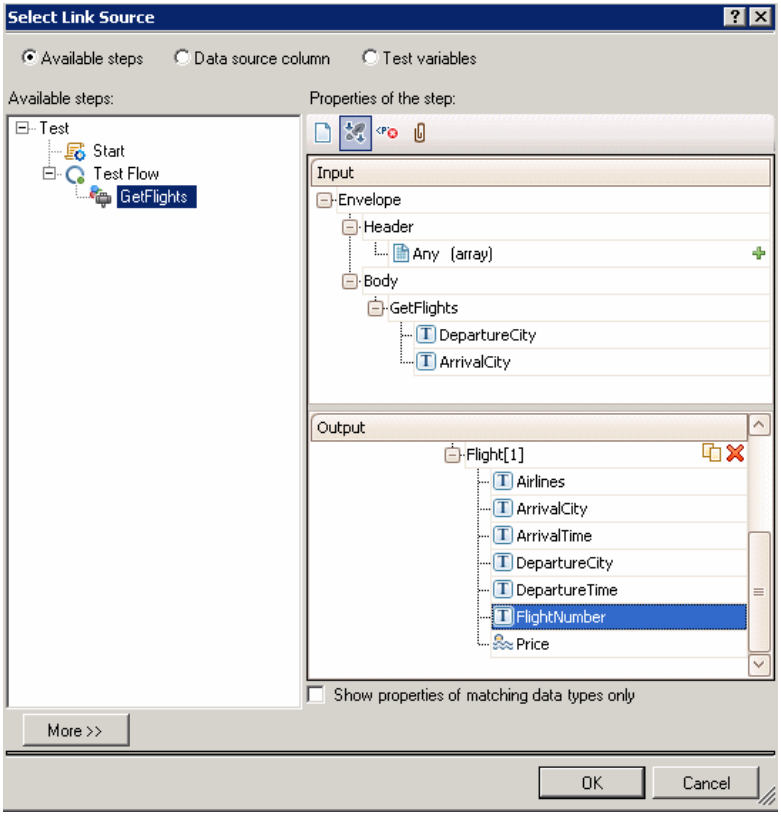
5 Link the output of GetFlights to the CreateFlightOrder step.

a Click the **Link to a data source** icon in the right corner of the **FlightNumber** row. The Select Link Source dialog box opens.

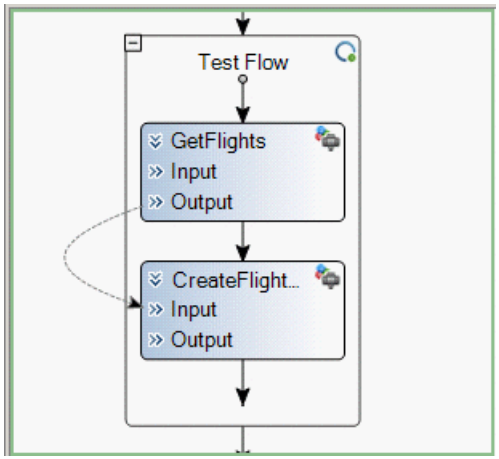
b Select **Available steps** and select the **GetFlights** node.

c In the right pane, select the **Input/Checkpoints** button. In the **Output** section, expand all nodes under the **Body** node. Click the **Add** button in the **Flight (array)** node row. Service Test creates the **Flight[1]** array.

d Expand the **Flight[1]** array, select the **FlightNumber** element, and click **OK**.



The canvas indicates a connection between the two steps.



6 Reset the number of iterations.

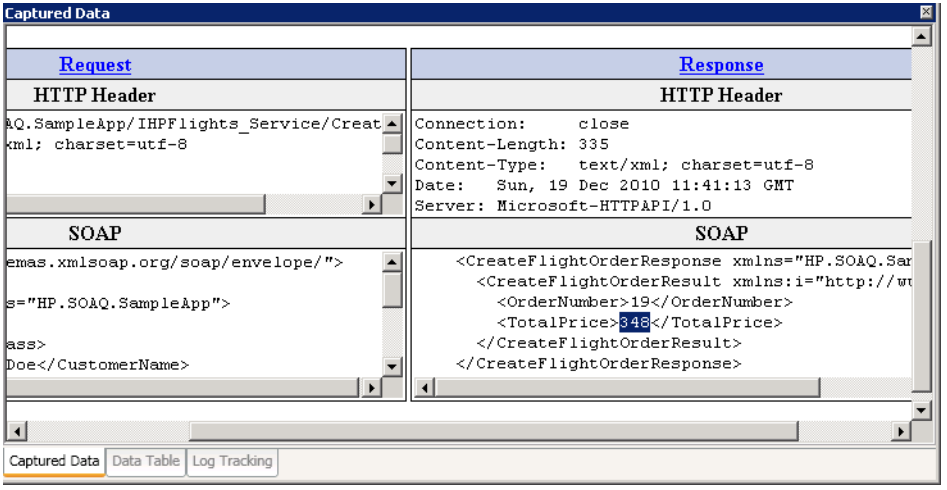
The number of iterations is the number of times to repeat the step. Return to the canvas and click inside the Test Flow frame—not within a test step. Open the Property Sheet's **Input/Checkpoints** view. Select **For Loop** and set the **Number of Iterations** to **1**.

7 Run the test.

Click the **Run** button. Observe the log in the **Output** tab. The Run Results Viewer opens automatically.

8 Check the results.

In the left pane, click the parent node and select **Expand All** from the right-click menu. Click the **CreateFlightOrder** node. In the **Captured Data** pane, scroll down to the Web service Call HTTP Snapshot section and look at the Response pane. Note the output of the request—**OrderNumber** and **TotalPrice**. Copy the **TotalPrice** value to the clipboard for use in the next step.



Tip: Click the **Request** or **Response** links to open the SOAP in a separate browser.

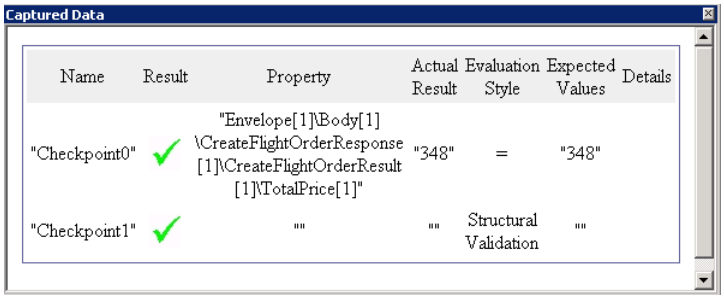
When you are finished viewing the results, close the Run Results Viewer.

9 Set a checkpoint.

Open the Property Sheet's **Input/Checkpoint** view, click in the Checkpoints grid, and expand the **CreateFlightOrderResponse** node. Paste the clipboard contents from the previous step into the **TotalPrice** field. Select the check box for the **TotalPrice** property, to include it as a checkpoint.

10 Run the test and view the checkpoint results.

Run the test again and expand the results tree. Select the **Checkpoints** node for **CreateFlightOrder**. The report shows a checkmark and indicates the expected and actual values. If the expected value was not returned by the server, the report indicates a failure.



When you are finished viewing the results, close the Run Results Viewer.

How do I integrate data into a test?

In this section you will learn how to integrate data from an existing source, and how to data drive the test. When you data drive a test, Service Test automatically creates a data table whose values you can edit.

1 Import Sample Data

In the Data Window tab, (at the bottom of the Service Test window), select **New** > **Excel**. The Add New Excel Data Source dialog box opens.

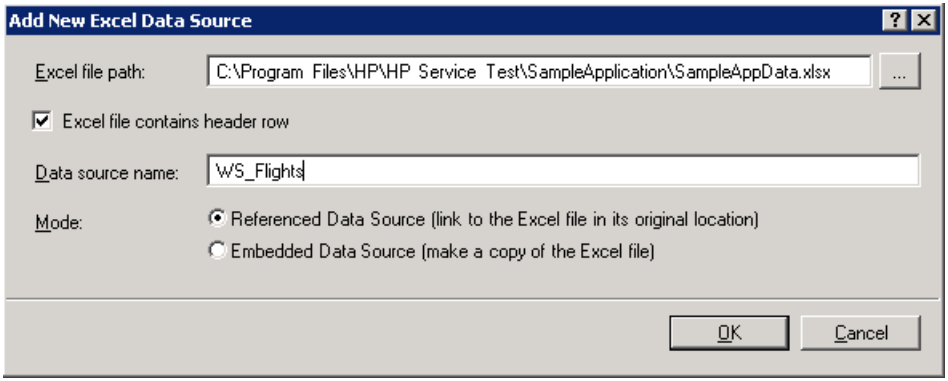
a Browse for the sample application's Excel file, **SampleAppData.xlsx**, in the <*installation directory*>/**SampleApplication** folder. By default, this folder is **C:\Program Files\HP\HP Service Test\SampleApplication**.

b Enable the **Excel file contains header row** option, since the sample file contains a header row.

c Enter WS_Flights as a **Data source name.**

d Select **Referenced Data Source** as the mode of import. This links to the Excel file at its original location, so that if data changes, your data source will be current.

Click **OK**.

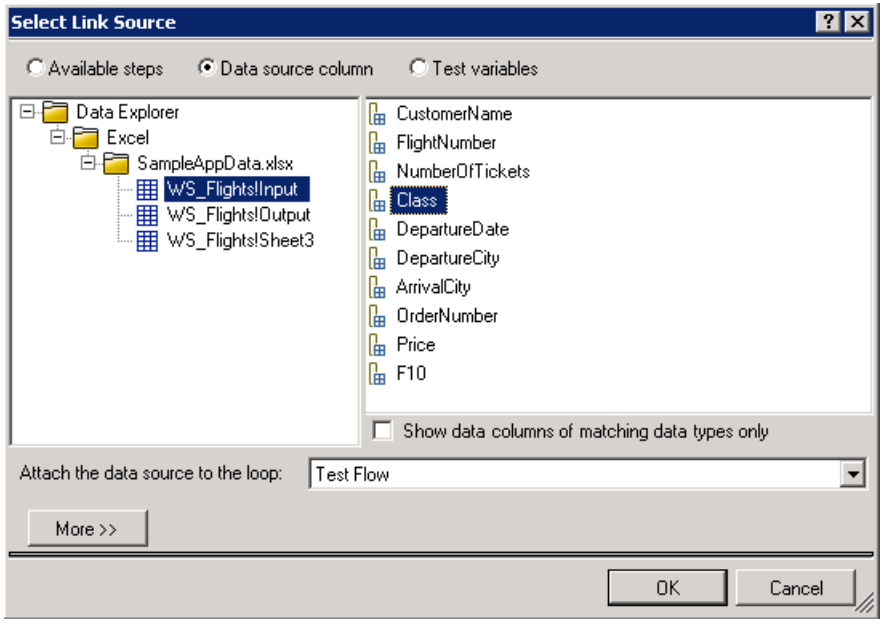


2 Open the Select Link Data dialog box.

Select the **CreateFlightOrder** step in the canvas and open the **Input/Checkpoints** view. In the **Input** section, select the **Class** row, and click the **Link to a data source** icon. The Select Link Source dialog box opens.

3 Select a value from the data source.

Select the **Data source column** option.



4 Use the sample Excel data.

Select the **WS_Flights!Input** node, and select **Class** in the right pane. Click **OK**. This instructs the test to refer to this column in the sample data during the test run.

Repeat this for the other input parameters: **CustomerName**, **DepartureDate**, **FlightNumber**, and **NumberofTickets**.

5 Disable the checkpoint.

In the **Input/Checkpoint** view, click in the Checkpoints grid. Clear the check box for the **TotalPrice** property, to exclude it as a checkpoint.

6 Set the navigation settings.

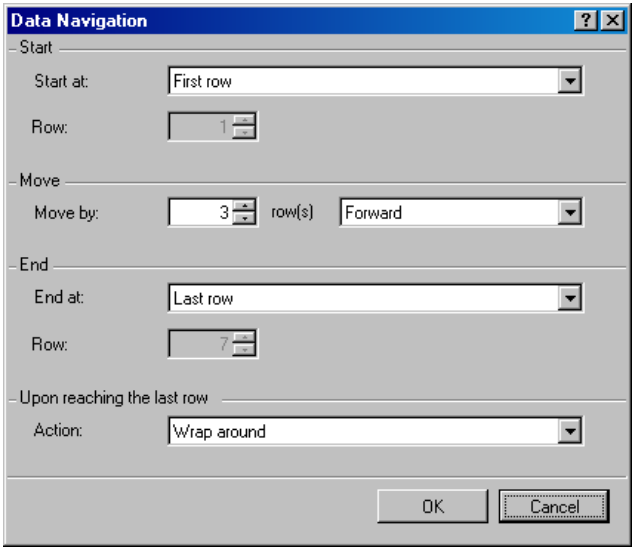
The navigation settings let you indicate how to use the data in your data source. You can specify from which row to begin, how many rows to advance, and in what direction to move for the next set of values. You can also specify what to do when reaching the end of the data table—wrap around or continue using the last line.

a In the canvas, click in the **Test Flow** but not within a step.

b In the Property Sheet, click the **Data Sources** view button.

c Select the **WS_Flights!Input** node and click **Edit** to open the Data Navigation dialog box.

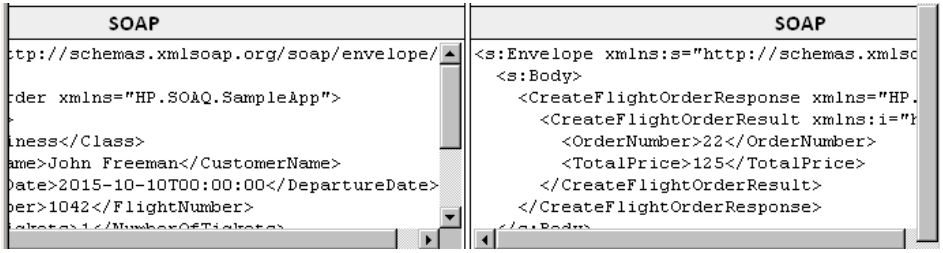
d Specify the data navigation details: **Start at**: First row, **Move**: Move by 3 rows Forward, **End at**: Last row, and **Upon reaching the last row**: Wrap around.



e Click **OK**.

7 Run the test and view the results.

Click the **Run** button and observe the results in the Output window. The Run Results Viewer opens automatically. Expand the result tree and select the **CreateFlightOrder** step. Scroll down within the **Captured Data** tab and note the data from the Excel file in the SOAP request (left pane), and the result in the SOAP response (right pane).



When you are finished viewing the results, close the Run Results Viewer.

How do I use multiple data sources and custom code?

This section describes how to define data using multiple data sources and sending information to the report through a custom code step.

1 Create a new test.

Create a new test called WebServicesCustom and import the HP Flights Services WSDL as described in "How do I import a Web service?" on page 32.

2 Create test steps.

Drag the activities into the canvas in the following order: From the **Web services** folder: **GetFlights** and **CreateFlightOrder**. From the **Miscellaneous** folder, drag in **Custom Code**.

3 Add a data source.

In the Data Window tab, select **New** > **Excel**. In the Add New Excel Data Source dialog box:

a Browse for the sample application Excel file. By default, this folder is **C:\Program Files\HP\HP Service Test\SampleApplication**.

b Select the **Excel file contains header row** check box.

c Enter WS_Flights as a **Data source name**.

d Select the **Referenced data source** mode.

4 Assign values for GetFlights.

Select the **GetFlights** step in the canvas and open the **Input/Checkpoints** view in the Property Sheet. In the **Input** section, select **DepartureCity**=Denver, and **ArrivalCity**=Los Angeles.

5 Assign values for the CreateFlightOrder.

Select the **CreateFlightOrder** activity in the canvas and open the **Input/Checkpoints** view. Expand the **Body** > **FlightOrder** node and set the input properties as follows:

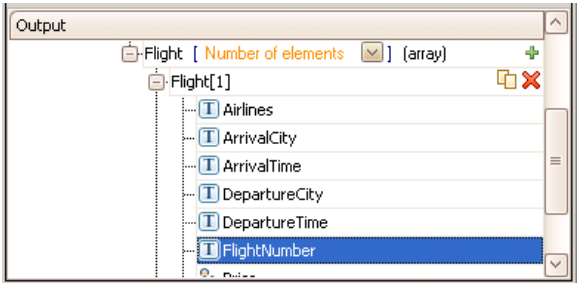
- **Class**. Economy
- **CustomerName**. Click the Link to a data source button in the right corner of the **FlightInfo** row. The Select Link Source dialog box opens. Select **Data source column**, and expand the tree to show the **WS_Flights!Input** node. In the right pane, select the **CustomerName** parameter. Click **OK**.
- **DepartureDate**. A date in the following format YYYY-MM-DDTHH:MM:SS For example, 2015-02-18T00:00:00. Use the drop down arrow to open the calendar. The date must be at least two days ahead of the current date.
- **NumberofTickets**. 3
- **FlightNumber**. Link from the previous step:

a Click the **Link to a data source** button in the right corner of the **FlightNumber** row.

b In the Link to Source dialog box, select **Available steps**, expand the **Test Flow** branch, and click **GetFlights**.

c In the right pane, select the **Input/Checkpoints** button.

d In the **Output** section, click the **Add** button in the **Flight (array)** node row. Service Test creates the **Flight[1]** array. Expand the array, select **FlightNumber**, and click **OK.**



6 Create a property for the custom code step.

Select the **Custom Code** activity in the canvas and open the **Input/Checkpoint** view in the Property Sheet. Expand the **Add Property** toolbar button and select **Add Input Property**. Create a new **String** type property called **FlightInfo**.

7 Define values for the custom code step.

In this step you will define a value using multiple sources. In this example, you will set a value which is a combination of the **CustomerName**, a constant string, and the **OrderNumber**.

a Click the **Link to a data source** button in the right corner of the **FlightInfo** row. The Select Link Source dialog box opens.

b Click **More** to expand the dialog box.

c Select the **Data source column** option. In the tree's **WS_Flights!Input** node, select **CustomerName**. Click **Add**.

d In the **Expression** box, type _OrderNumber_ (with the underscores) after the existing expression.

e Select **Available steps** and expand the **Test Flow** branch. Select the **CreateFlightOrder** node. In the right pane, select the **Input/Checkpoints** button. In the lower pane, expand the Output **Body** node, expand the tree, select the **OrderNumber** element, and click **Add**.

The CustomCode input property, **FlightInfo**, has the following value:

```
{DataSource.WS_Flights!Input.CustomerName}_OrderNumber_{Step.OutputPropert
ies.StServiceCallActivity5.Body.CreateFlightOrderResponse.CreateFlightOrderResu
lt.OrderNumber}
```

Click **OK** to close the dialog box.

8 Create an event.

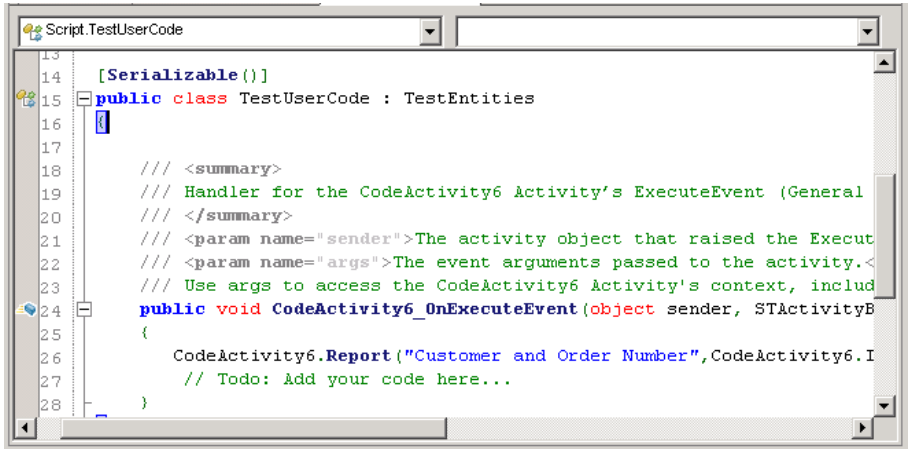
In this step you will create an event handler to use the Service Test API. You can add C# code to this section. Defining events let you adapt Service Test to your custom requirements, and perform actions that are not built-in to Service Test. In this example, you will add code that sends a custom string to the report.

Select the **Custom Code** step in the canvas. In the Property Sheet, click the **Events** button. In the **ExecuteEvent** row, click the drop-down arrow and select **Create a default handler**. Service Test creates an event called **CodeActivity6_OnExecuteEvent** and opens a new tab **TestUserCode.cs.**

9 Edit the "Todo" section.

Replace the commented text in the **Todo** section with the following:

```
CodeActivity6.Report("Customer and Order
Number",CodeActivity6.Input.FlightInfo);
```



10 Run test and check the results.

Drill down in the results to the **Custom Code** step. Note the new entry in the **Captured Data** pane: Customer and Order Number.

Tip: You can also use the **Report Message** activity under the **Miscellaneous** folder, to send text and property values to the report.

Where To Go From Here

Now that you have learned to create a test for a Web service, you can relate this to other types of application components. The next lesson will walk you through the process of creating a test for a REST service.

4

Test REST Services

Using Service Test, you can model and create tests for REST services.

This lesson contains the following sections:

- How do I model a REST service?
- How do I create a simple test for a REST service?
- How do I create a template for a REST method?
- How do I create a REST test using table data?

How do I model a REST service?

This section describes how to model a REST service using the sample application.

1 Start the Sample Flight application.

Make sure that the Flight Application service is running, as described in"How do I invoke the application?" on page 13.

2 Create a new solution.

Select **File** > **New** > **Test** and specify the name **RESTServiceTest** for the new test. Use the standard **Service Test** template. Click **Create**.

3 Create a REST service.

Right-click the **REST Services** node in the Toolbox and select **Add Service**. Rename the service to **SampleRESTService**.

4 Create a resource.

Right-click **SampleRESTService** and click **Add Resource**. Rename the resource to **FlightOrders**.

5 Create a method.

Right-click the **FlightOrders** resource and select **Add Method**. Rename the method to **ReserveOrder**.

How do I create a simple test for a REST service?

A best practice for working with REST services is to create a reusable template for the REST method. For details, see "How do I create a template for a REST method?" on page 54.

This section describes how to create a simple test for a REST service without a reusable template.

1 Create a test step.

Drag the **ReserveOrder** method you created above, into the **Test Flow** of the canvas in order to create a **ReserveOrder** step. Click the plus sign in the step's top left corner to expand the step, and click the **HTTP Activity** box.

2 Open the REST service Help page.

Go to the Sample Application command window, type "h", and press ENTER. A browser opens with the URLs for the REST service. Copy the URL for the **FlightOrders** > **ReserveOrder (POST)** operation, http://localhost:24240/HPFlights_REST/FlightOrders/ to the clipboard.

3 Enter the input values.

a Return to Service Test and open the **Input/Checkpoints** view in the Property Sheet.

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

c Set the **HTTP Method** to **POST**.

d Click the **Add** button in the **Request Headers (array)** row to add an array element.

e Expand the **Request Header** array. Using the method's details from the Help page, set the header name and value as follows: In the **Key** row, type Content-Type. In the **Value** row, type text/xml.

4 Copy the request body.

Return to the browser opened in the previous step, and copy the Request Body for **FlightOrders** > **ReserveOrder (POST)** to the clipboard.

```
<FlightOrderDetails xmlns="HP.SOAQ.SampleApp"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <Class>Business</Class>
  <CustomerName>John Doe</CustomerName>
  <DepartureDate>2012-12-12</DepartureDate>
  <FlightNumber>1304</FlightNumber>
  <NumberOfTickets>21</NumberOfTickets>
  </FlightOrderDetails>
```

5 Save the body to a text file.

Open a new text file in Notepad and paste in the contents of the clipboard. Save the file as **body.xml** to any location.

6 Provide the HTTP body.

a In the Property Sheet, open the **HTTP** view.

b Select the **XML** type from the **Request Body** drop-down list.

c Click the **Load XML** button (you may need to enlarge the window) and select the XML file, **body.xml**, that you saved in the previous step.

d Click the **Text** and **Grid** tabs beneath the Request Body drop-down, to display the data in different views.

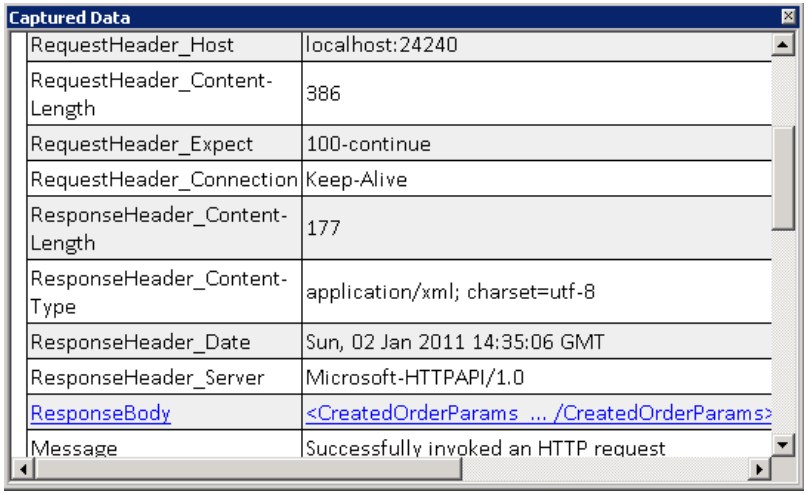
7 Run the test.

Click the **Run** button or press F5 to run the test.

8 View the report.

In the left pane, select **Expand All** from the right-click menu. Select the **HTTP** node.

In the **Captured Data** pane, verify that the **Response Body** row contains **OrderNumber** and **TotalPrice** values. This corresponds to the operation's description in the REST service Help page.



Click the **ResponseBody** link to open the response in a separate browser.

When you finish reviewing the results, close the Run Results Viewer.

How do I create a template for a REST method?

This section describes how to create an HTTP template for a specific method. The HTTP template contains the method's constant information, such as the request URL, method type, and custom parameters. Once you have the template, you can repeatedly use the method within your test.

1 Remove the previous step.

Click in the canvas and select the **ReserveOrder** step you created in the previous section. Press the **Delete** button on your keyboard. Click the **Save** button on the toolbar.

2 Create a new method.

In the Toolbox palette, select the **FlightOrders** resource. Use the right-click menu to create a new method, **FlightOrderTemplate** and select it.

3 Configure the method's HTTP properties.

Open the **HTTP Input/Checkpoints** view in the Property sheet.

4 Open the REST service help page.

If the browser with the REST service help is no longer open, reopen it by typing "h" in the Sample Application window. Copy the URL for the **FlightOrders** > **ReserveOrder (POST)** operation, to the clipboard:
http://localhost:24240/HPFlights_REST/FlightOrders/

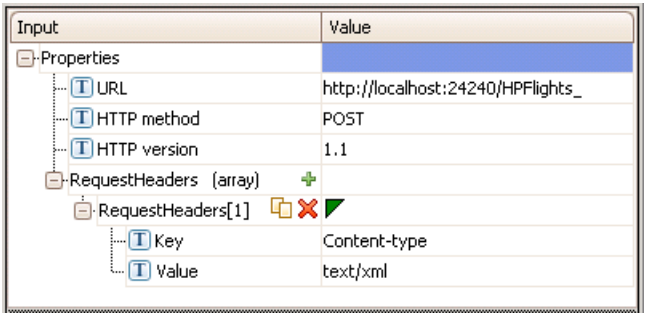
5 Set the input properties of the REST service.

a Paste the URL into the **URL** row.

b Set the **HTTP Method** to **POST**.

c Click the **Add** button in the **Request Headers (array)** row to add an array element.

d Expand the **Request Header** array. Using the method's details from the Help page, set the header name and value as follows: In the **Key** row, type Content-Type. In the **Value** row, type text/xml.



6 Create input properties.

a In the Toolbox palette, select the **FlightOrderTemplate** method. Open the **Input/Checkpoint** view in the Property Sheet.

b Expand the **Add Property** button and select **Add Input Property**.

c Add a **String** type property called **Class**. Add another property called **Customer_Name**.

7 Load the request body.

a In the Property Sheet, open the **HTTP** view.

b Select the **XML** type from the **Request Body** dropdown list.

c Click the **Load XML** button and load the **body.xml** file you saved in the previous section.

d Click the **Save** button.

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

How do I create a REST test using table data?

This section describes how to use the template you created, to easily create a test and add data from a data table.

1 Create a test step.

Drag the **FlightOrderTemplate** method into the **Test Flow** on the canvas.

2 Import sample data.

In the **Data Window** tab, select **New** > **Excel**. The Add New Excel Data Source dialog box opens.

a Click in the **Excel file path field**, and browse for the sample application's Excel file in the <*installation directory*>/**SampleApplication** folder. By default, this folder is **C:\Program Files\HP\HP Service Test\SampleApplication**.

b Select the **Excel file contains header row** check box since the sample contains a header row.

c Specify a **Data source name**, REST_Flights.

d Select **Referenced data source**. This links to the Excel file at its original location, so that if data changes, your data will be current.

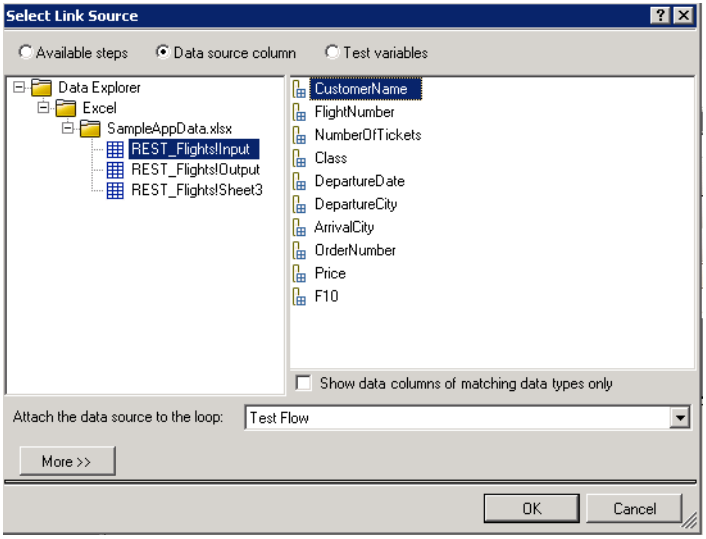
e Click **OK**.

3 Associate custom properties with a data source.

a Select the **FlightOrderTemplate** in the canvas and open the **Input/Checkpoint** view in the Property Sheet. Do not select the **HTTP** section within **FlightOrderTemplate**.

b Click the **Link to a data source** button in the right corner of the first row, **Class**. The Select Link Source dialog box opens.

c Select the **Data source column** option. Expand the **SampleAppData.xlsx** tree and select the **REST_Flights!Input** node. In the right pane, double-click the **Class** property.



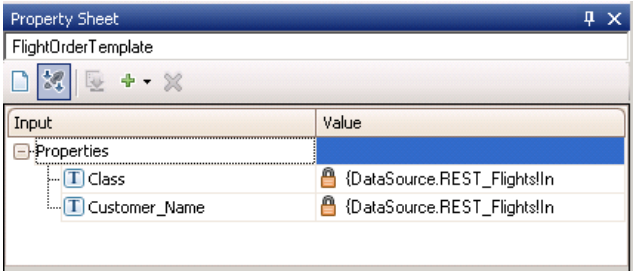
4 Associate the Customer_Name element with a data source.

a Select the **FlightOrderTemplate** in the canvas and open the **Input/Checkpoint** view in the Property Sheet.

b Click the **Link to a data source** button in the right corner of the second row, **Customer_Name**. The Select Link Source dialog box opens.

c Select the **Data source column** option. Expand the **SampleAppData.xlsx** tree and select the **REST_Flights!Input** node. In the right pane, select the the **CustomerName** property and click **OK**.

d Note the expressions in the **Value** column.



5 Run the test.

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

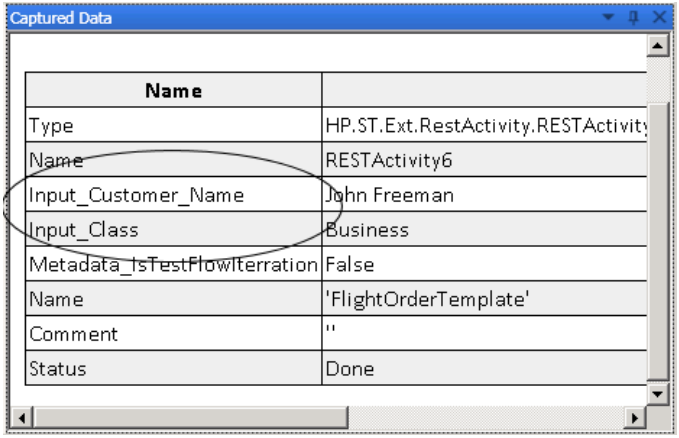
6 View the report.

In the left pane, select **Expand All** from the right-click menu. Open the **Captured Data** pane,

7 Verify the data.

a Select the **HTTPx** node. Click the **Response Body** link. Verify that the XML contains **OrderNumber** and **TotalPrice**. This corresponds to the operation's description in the REST service Help page.

b In the Run Results Viewer, select the **FlightOrderTemplate** node. Verify that the input properties you defined earlier, match the actual values.



When you finish reviewing the results, close the Run Results Viewer.

Where To Go From Here

Now that you have learned to create tests with standard activities, Web services, and REST services, you can create your own tests for your GUI-less applications.