

Rational Integration Tester



# Getting Started Guide

*Version 8.0.1*



**Note**

Before using this information and the product it supports, read the information in “Notices” on page 65.

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

© Copyright IBM Corporation 2001, 2012.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>About this Publication.....</b>	<b>v</b>
Intended Audience.....	vi
Scope .....	vi
Typographical Conventions.....	vi
Contacting IBM Support .....	vi
<b>Getting Started.....</b>	<b>1</b>
Prerequisites .....	2
Starting Rational Integration Tester .....	3
Creating a New Project .....	5
<b>Working with Rational Integration Tester.....</b>	<b>7</b>
Initial View .....	8
Rational Integration Tester Perspectives.....	9
GUI Overview .....	10
Architecture School .....	10
Environments.....	18
<b>Testing a Web Service.....</b>	<b>19</b>
Synchronizing With WSDLs .....	20
Verifying WSDL Resources .....	25
Creating Tests .....	29
Running the Tests .....	37
Creating Test Suites.....	39
Viewing Test Reports.....	43

---

<b>Data Validation</b>	<b>45</b>
Overview	46
Exercise	47
<b>Test Data Sets</b>	<b>49</b>
Overview	50
Creating Data Sources	51
Adding Log Actions to Tests	52
Configuring Tests	54
<b>Functions and Decisions</b>	<b>57</b>
Overview	58
Exercise	59
<b>Glossary</b>	<b>63</b>
<b>Notices</b>	<b>65</b>
Trademarks and service marks	68

# About this Publication

## **Contents**

### **Intended Audience**

### **Scope**

### **Typographical Conventions**

### **Contacting IBM Support**

This guide provides a brief introduction to IBM® Rational® Integration Tester. It demonstrates some of the basic features of Rational Integration Tester, including basic GUI elements, synchronizing with WSDLs, and validating data.

Examples in this guide are described using SOAP/HTTP (it is not possible to show examples using every supported messaging product). Regardless of the messaging used in your environment, the basic principles are the same and you should be able to relate the examples to your own environment.

---

## Intended Audience

This document is intended to be read by those with a fair understanding and exposure to the concepts involved in both testing and development and in enterprise integration. It is aimed at first-time users of IBM Rational Integration Tester, and those who are interested in its capabilities and ease of use. No prior knowledge of Rational Integration Tester is assumed.

## Scope

This document provides a quick introduction to Rational Integration Tester. (For information about installing Rational Integration Tester, refer to *IBM Rational Integration Tester Installation Guide*.)

Detailed descriptions of Rational Integration Tester features, syntax, and functionality can be found in *IBM Rational Integration Tester Reference Guide*. More detailed examples for specific messaging systems can be found in the various plugin guides that are provided with Rational Integration Tester.

## Typographical Conventions

The following typographical conventions are observed throughout this document:

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

## Contacting IBM Support

To contact IBM Support, see: [www.ibm.com/contact/us/en/](http://www.ibm.com/contact/us/en/)

# Getting Started

## **Contents**

### **Prerequisites**

### **Starting Rational Integration Tester**

### **Creating a New Project**

This chapter describes how to start using Rational Integration Tester, including some prerequisite tasks (that is, database setup and Rational Integration Tester installation).

---

## 1.1 Prerequisites

Before you can start using Rational Integration Tester, the product must be installed and the project database must be created – Oracle, SQL Server, and MySQL are supported.

For more information about both of these tasks, refer to *IBM Rational Integration Tester Installation Guide*.



---

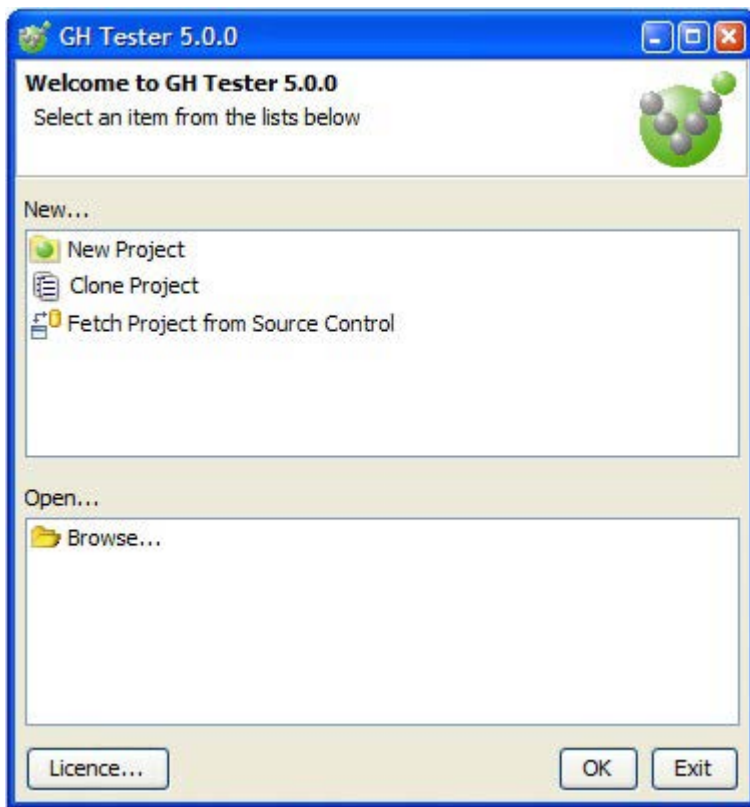
## 1.2 Starting Rational Integration Tester

On the Windows platform, Rational Integration Tester can be launched in several ways:

- Select **Rational Integration Tester** from the program group under the **Start** menu (for example, **Start > Programs > IBM > Rational Integration Tester**).
- Double-click the **Rational Integration Tester** shortcut on the desktop.
- From the **Run** dialog (**Start, Run**):
  - Enter the full path to the Rational Integration Tester application (for example, "C:\Program Files\IBM\RationalIntegrationTester\GHTester.exe") and click **OK**.
  - Click **Browse** to locate the Rational Integration Tester application, select it, then click **OK**.
- From a command prompt:
  - Enter the full path to the Rational Integration Tester application from any command prompt:  
"C:\Program Files\IBM\RationalIntegrationTester\ghtester.exe"
  - Change to the Rational Integration Tester installation directory and enter ghtester.exe (or just ghtester).
- Browse to the Rational Integration Tester installation directory and double-click **GHTester.exe**.

---

When Rational Integration Tester starts you will be presented with the **Welcome** screen.



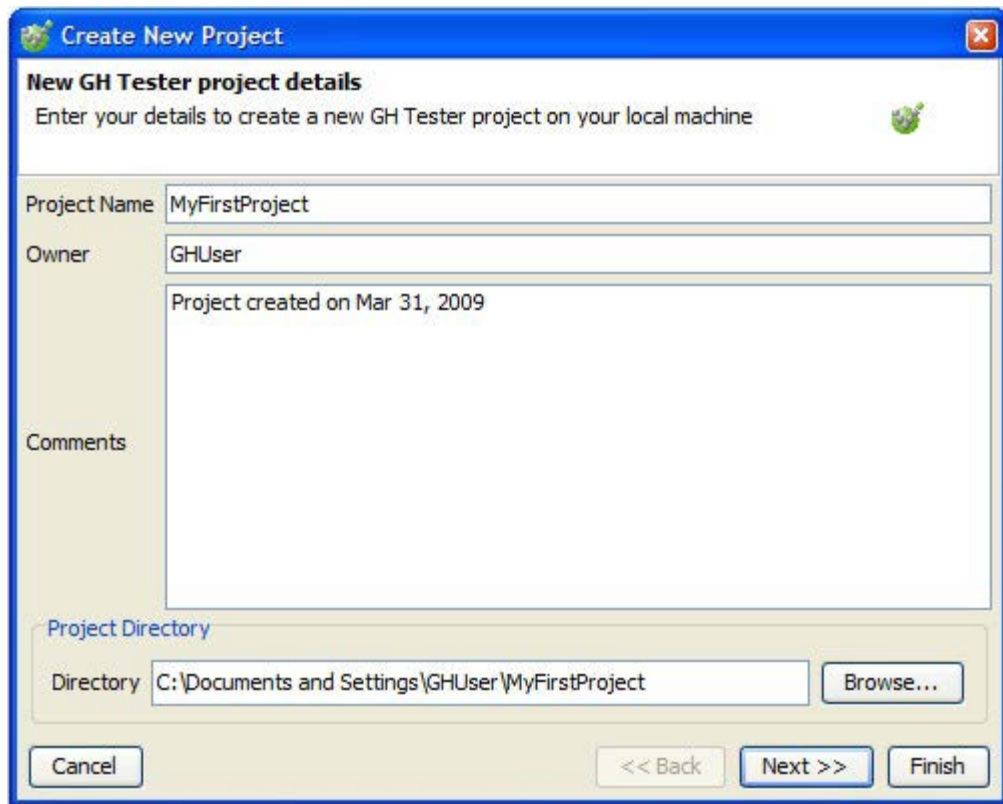
---

## 1.3 Creating a New Project

To create a new project:

1. In the Welcome screen, select **New Project** and click **OK**.

The **Create New Project** wizard will be launched.



The screenshot shows a Windows-style dialog box titled "Create New Project". Inside, the "New GH Tester project details" section prompts the user to "Enter your details to create a new GH Tester project on your local machine". There are three input fields: "Project Name" with the text "MyFirstProject", "Owner" with the text "GHUser", and a "Comments" text area containing the text "Project created on Mar 31, 2009". Below these fields is a "Project Directory" section with a "Directory" text box containing "C:\Documents and Settings\GHUser\MyFirstProject" and a "Browse..." button. At the bottom of the dialog are four buttons: "Cancel", "<< Back", "Next >>", and "Finish".

2. Enter a name for the new project and enter any desired comments. You may also change the default project owner if you like.
3. If desired, change the project directory, which does not have to have the same name as the project.

**NOTE:** If the directory does not exist you will be prompted to create it. It is possible to store the project on a network drive but overall performance would be affected adversely.

4. When ready, click **Next** to proceed.

---

The **Project Database** screen is displayed.

**Create New Project**

**Project Database**

Configure the project database for storing all test results.  
Schema version required: 1.9.17

Database Provider: Oracle

**Connection Details**

Database URL: jdbc:oracle:thin:@localhost:1521:XE

User Name: gh

Password: ..

Test Connection

**Database Details**

Property	Value
----------	-------

Cancel << Back Next >> Finish

**NOTE:** Configuring a proper connection to the project database is necessary if you want to access historical test results to produce reports.

5. Enter the Database URL, user name, and password (according to how your own project database was set up) and click on **Test Connection** to verify the connection to the database.
6. Once the database connection has been enabled, click **Finish** to create the new project and open it in Rational Integration Tester.

**NOTE:** See the next chapter, [Working with Rational Integration Tester](#), to learn more about the Rational Integration Tester workspace.

# Working with Rational Integration Tester

## **Contents**

### **Initial View**

### **Rational Integration Tester Perspectives**

### **GUI Overview**

### **Architecture School**

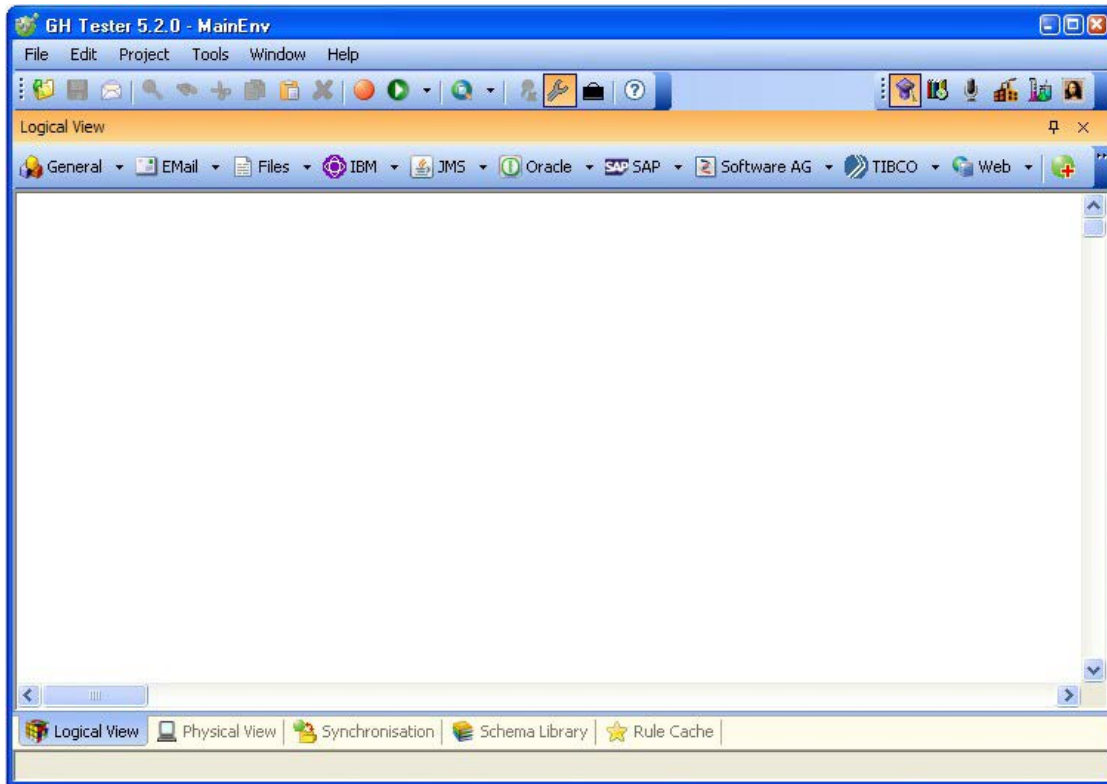
### **Environments**

This chapter describes the Rational Integration Tester interface (the Workbench) and the perspectives and views that it contains.

---

## 2.1 Initial View

After launching Rational Integration Tester for the first time and creating a new project, the following screen is displayed:




This is Architecture School, one of Rational Integration Tester's six perspectives. More information about perspectives is provided in the next section.


---


## 2.2 Rational Integration Tester Perspectives


The workbench can be viewed from one of six perspectives, selected from the Perspectives toolbar:




 Architecture School (F7) - define the architecture of the system under test, including service components and logical and physical resources

 Requirements Library (F8) - create requirements, in the form of messages, that will help other users to create tests and test data more quickly and more accurately

 Recording (F9) - monitor systems and processes to record events that are captured by Rational Integration Tester

 Test Factory (F10) - create tests and test resources (tests, test suites, and test data sets)


 Test Lab (F11) - execute test resources that are created in the Test Factory

 Results Gallery (F12) - view historical test data and various reports for any store test run

**NOTE:** Each of the Rational Integration Tester perspectives are described in greater detail in *IBM Rational Integration Tester Reference Guide*.



---

## 2.3 GUI Overview

The Rational Integration Tester workspace utilizes ‘docking’ (that is, sub-windows are attached or ‘docked’ to one of the sides of the main window). These windows can be minimised automatically when not in use. To keep them fully visible they may be pinned to the workspace using the  icon. Once a window is pinned, it can be dragged to a different side of the workspace.

As you drag a window around the workspace, a grey outline of the window is displayed to show you where the window will be placed when you release it.

**NOTE:** If you want to restore the workspace to its default configuration, select the **Window > Reset Current Perspective** menu option.

To allow more viewing space in some of the configuration windows, you have the option to hide sections. For example, you may want to hide the Message Header section in a Publish test action so that you can see more of the message editor. To toggle the display of such areas, use the hide  and show  buttons.

## 2.4 Architecture School

Architecture School is Rational Integration Tester’s default perspective, and it is the first one displayed when the application is first started. In this perspective, users build the system to be tested in a simple, graphical fashion. The user defines the logical and physical system components and their relationships to one another.

Users can do the following in Architecture School:

- Create and edit Service and Infrastructure Components
- Define a Service Component's operations and its dependencies to other operations and Infrastructure Components
- Create and edit Physical Resources
- Define the bindings of Infrastructure Components to their Physical Resources for different testing environments

Architecture School contains five views (accessed by named tabs at the bottom of the perspective): the Logical View, the Physical View, the Synchronization view, the Schema Library, and the Rule Cache.

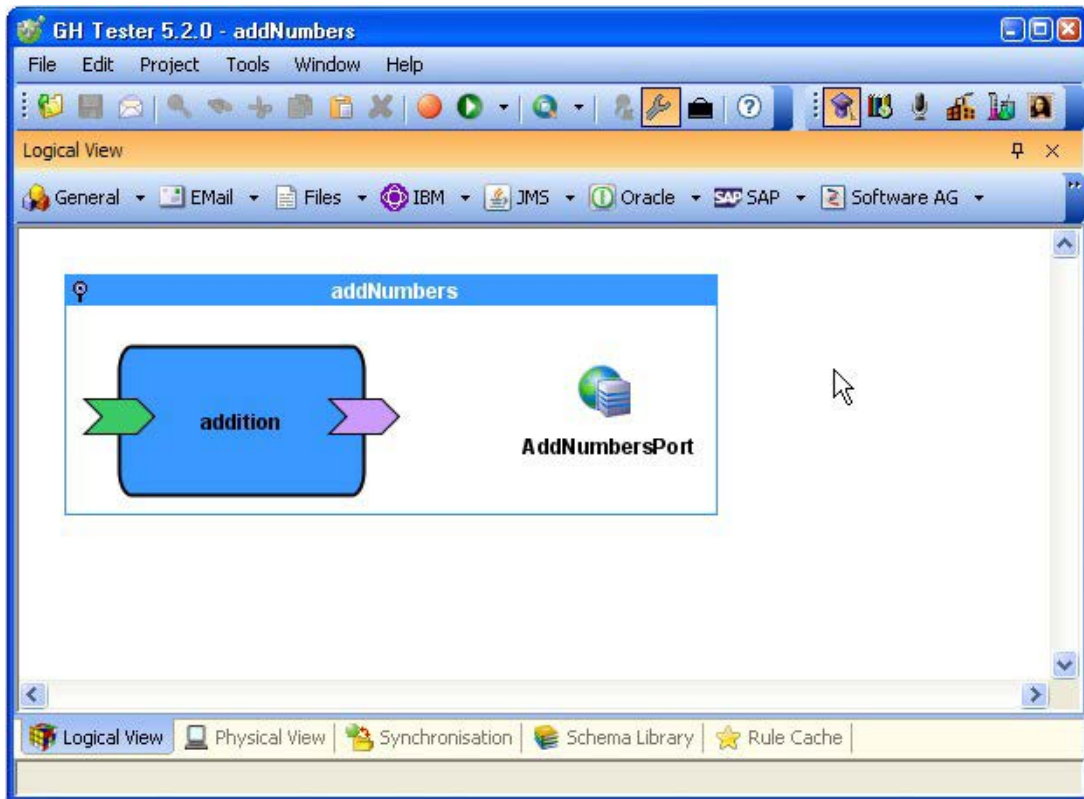
**NOTE:** The following screenshots are examples only – no items or data are pre-configured in Rational Integration Tester.









---

### 2.4.1 The Logical View

The **Logical View** displays the services and components of the system under test, and the dependencies between them. It allows you to look at the system in an abstract manner, without having to worry about the physical locations of the different parts of the system.

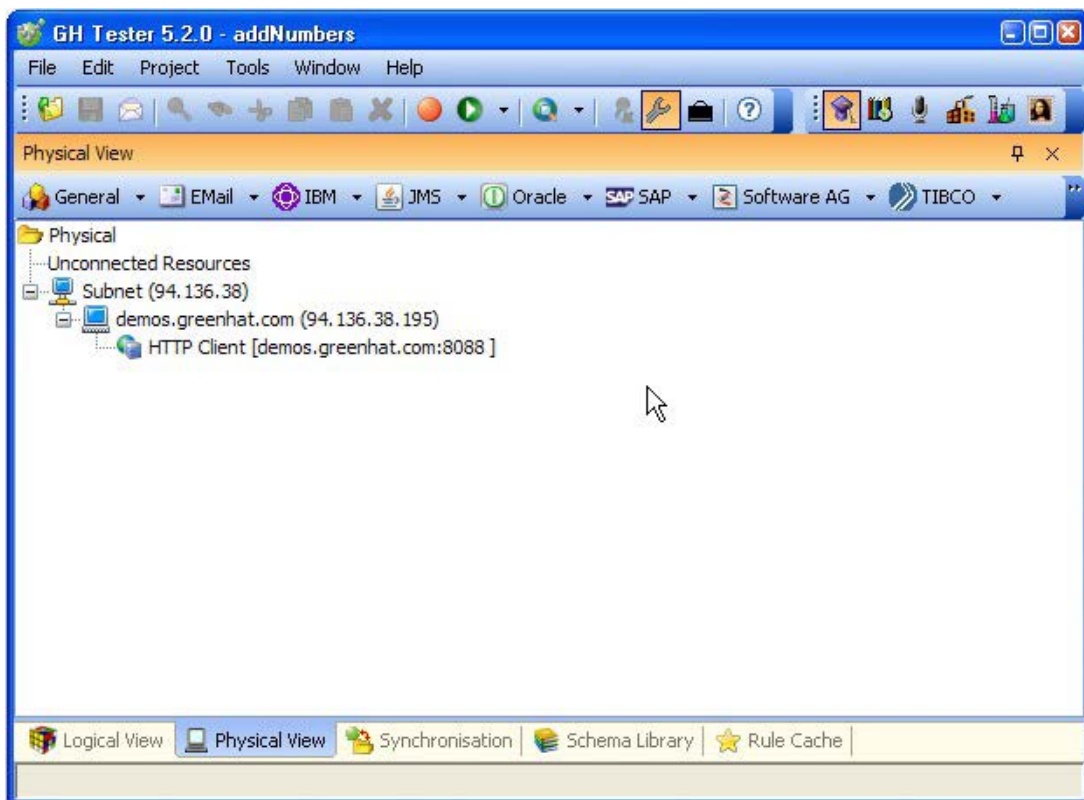


- To navigate the logical view you can use the horizontal and vertical scrollbars, or select the **Pan** cursor  to 'grab' the screen (left-click) and drag it around.
- To adjust the zoom level you can use either the zoom buttons ( and ) , or hold down the **Ctrl** key and use the mouse wheel.
- To move any of the components or operations around, ensure the **Select** cursor  is active, then left-click on the item to move and drag it to the desired location.
- To have Rational Integration Tester organize the layout of all components and operations, click the **Layout All Nodes** button .
- To set the zoom level so that the entire diagram fits inside the screen, use the **Fit to Contents** button .

---

## 2.4.2 The Physical View

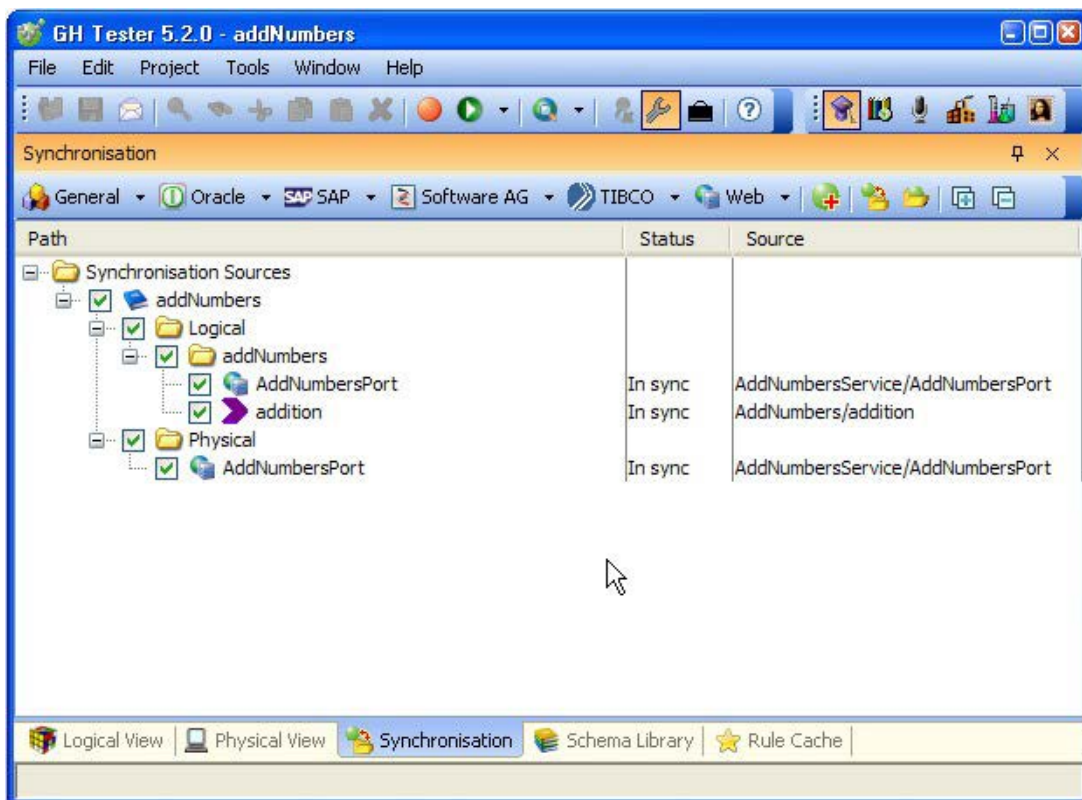
The **Physical View** displays available physical resources and their location within the enterprise. Resources in this view are organized by subnet and host. If a resource is not associated with a subnet or host, it will be displayed under **Unconnected Resources**.



---

### 2.4.3 The Synchronisation View

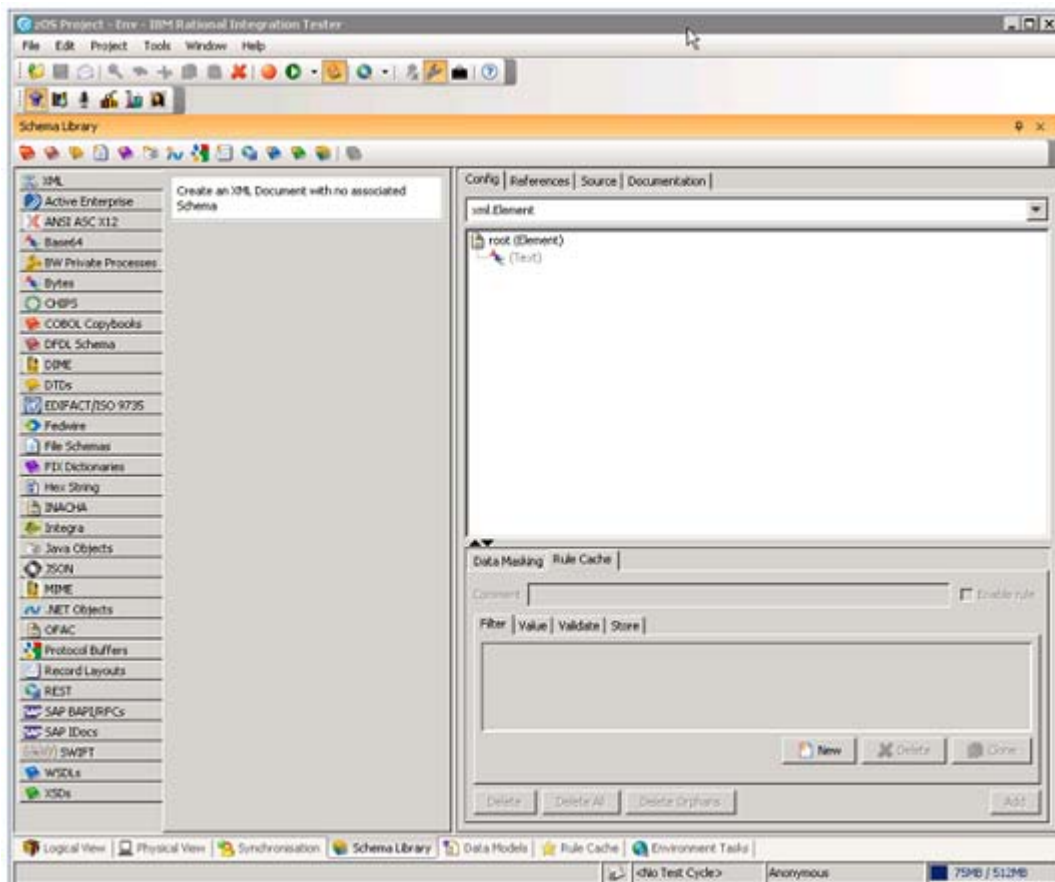
The **Synchronisation View** enables you to add external resources to your project, view resources that have been added, and synchronize those resources with your local system. Currently, you can synchronize with a WSDL, a Web Methods Integration Server, and a TIBCO Business Works Project.



---

## 2.4.4 The Schema Library

The **Schema Library** enables you to manage and view the schemas and message formats that are available in your project.



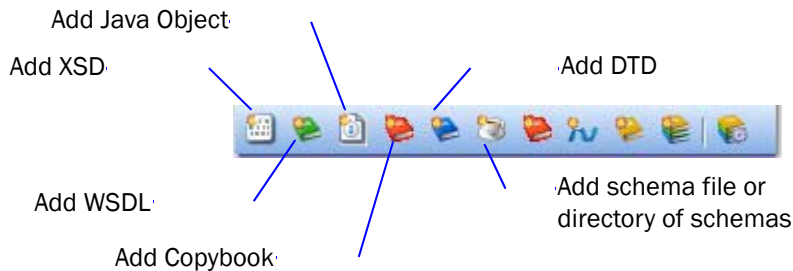
Each schema category and format is listed under its own tab on the left side of the view. When a category is selected, all of the schemas available in the project are displayed in a tree format in the view's center frame.


The schemas can be sorted by namespace or location by selecting the appropriate option button above the tree view, and the types of schemas displayed (files or URLs) can be limited using the **Filters** option.

---

## Adding Schemas

Schemas can be added to the project using the appropriate icons at the top of the Schema Library view.



Click the **Rebuild...** icon  to rebuild the selected schema and view any changes that may have been made to the source.

## Schema Details

On the far right of the view you can view and manage the details of any selected schema.

- Under the **Config** tab you can view or modify the location of the schema. For WSDLs, you can analyze the file in the same way as in the WSDL import wizard. You can view a preview of the schema roots and the format of all available schema messages.
- Under the **References** tab you can see all of the artefacts within the project that contain a reference to the selected schema.
- Under the **Source** tab you can view the schema source.
- Under the **Documentation** tab you can view or modify additional details about the schema, including an optional link to external documentation, the date and time when the schema was created and last updated, and the source of the schema.
- The **Rule Cache** tab enables you to view and configure any validation rules that have been applied in the selected schema. For more information, see [The Rule Cache](#).

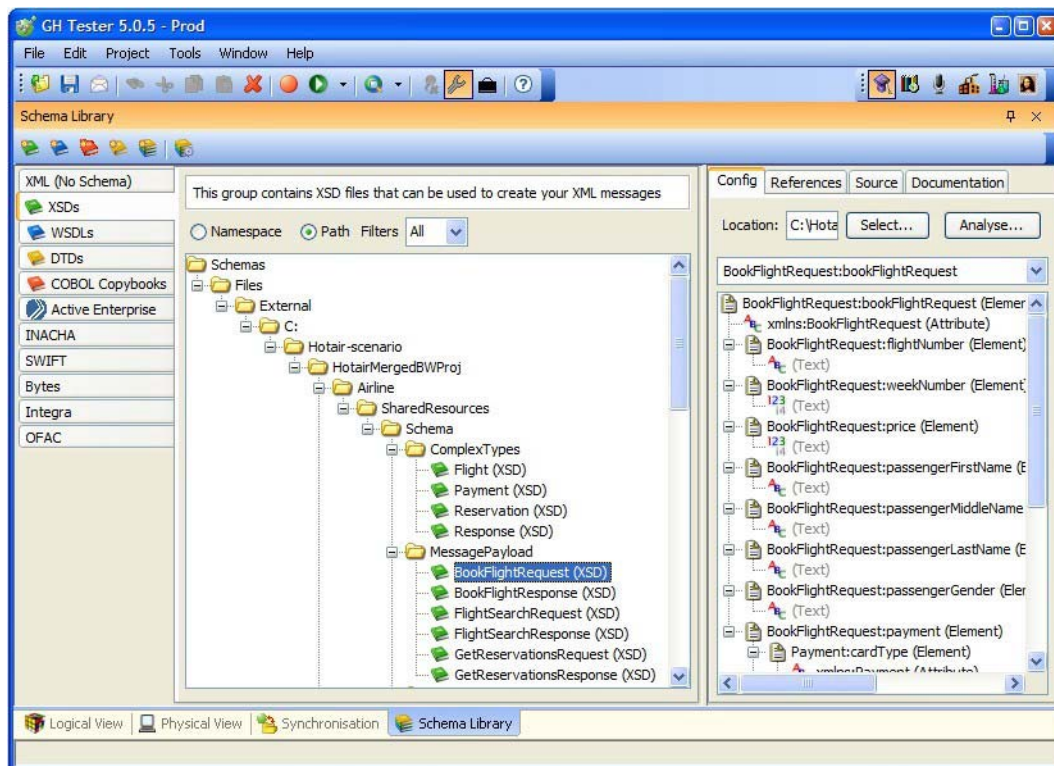
## Format Details



For message formats, there is no tree view in the center frame, and only the **Config** tab is available on the far right side of the view. Here you can preview all of the different message types that are available under the selected format.

## 2.4.5 The Rule Cache

When validation errors are repaired and cached (refer to *IBM Rational Integration Tester Reference Guide*), the repair method is saved as a validation rule. These rules will be applied to the same field from which they are created when the same schema is in use.

Rules are managed under the **Rule Cache** view in Architecture School.



Existing rules are listed on the left side of the view, and they can be sorted by namespace or path by clicking on the appropriate column heading. The star in the left column indicates for each rule whether or not the rule is currently enabled  or disabled .

**NOTE:** The same icons are used on individual fields to indicate where a rule has been applied and whether or not the rule is currently enabled.

Under the **Config** tab on the right side of the view, the selected rule can be enabled or disabled with the **Enable rule** option. Validation actions are listed under the **Validate** tab. Actions can be created, modified, cloned, and deleted, the same way as in a message editor (refer to *IBM Rational Integration Tester Reference Guide*). When the



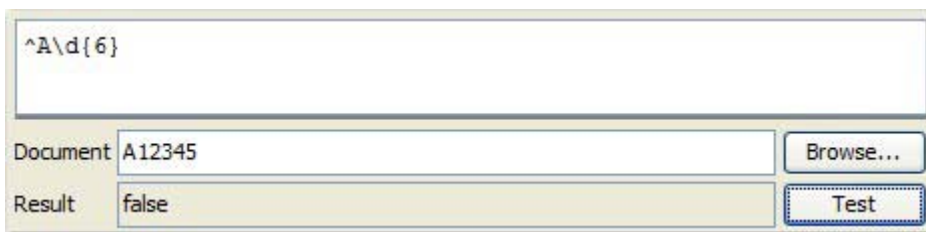
---

selected action includes a rule, it is displayed below. The rule can be edited, and a description can be added if desired.

Click **Delete All** to remove all rules from the selected schema. Click **Delete Orphans** to remove all rules from the selected schema that are associated with fields that no longer exist (due to changes to the schema). This button is only available when “orphaned” rules actually exist for the schema.

## Testing Rules

The current rule can be tested using the contents of a file or by entering a testable value manually.

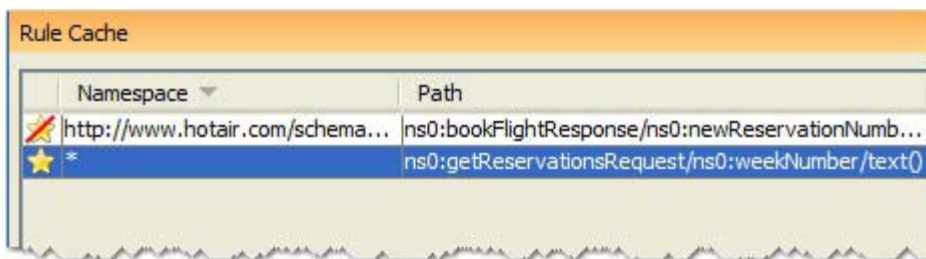


The screenshot shows a dialog box for testing a rule. At the top is a text field containing the regular expression `^A\d{6}`. Below this are two rows of controls. The first row has a label 'Document' followed by a text field containing 'A12345' and a 'Browse...' button. The second row has a label 'Result' followed by a text field containing 'false' and a 'Test' button.

To test with file contents, click **Browse** to locate and select the desired file. Otherwise, simply type the string that you want to use to test the rule in the **Document** field. To test the document or the manual value, click **Test**. If the tested value matches the rule, **true** will be returned. If the value does not match the rule, **false** is returned.

## Generic Rules

You can create XML-based rules (that is, rules that do not depend on a schema) when working in message editors. These rules are displayed in the **Rule Cache** view with a \* for their namespace, since they are applied to matching message fields regardless of schema or target namespace.



The screenshot shows the 'Rule Cache' window with a table of rules. The table has two columns: 'Namespace' and 'Path'. The first row has a red flag icon in the 'Namespace' column, the text 'http://www.hotair.com/schema...' in the 'Namespace' column, and 'ns0:bookFlightResponse/ns0:newReservationNumb...' in the 'Path' column. The second row has a yellow star icon in the 'Namespace' column, an asterisk '\*' in the 'Namespace' column, and 'ns0:getReservationsRequest/ns0:weekNumber/text()' in the 'Path' column. The second row is highlighted in blue.

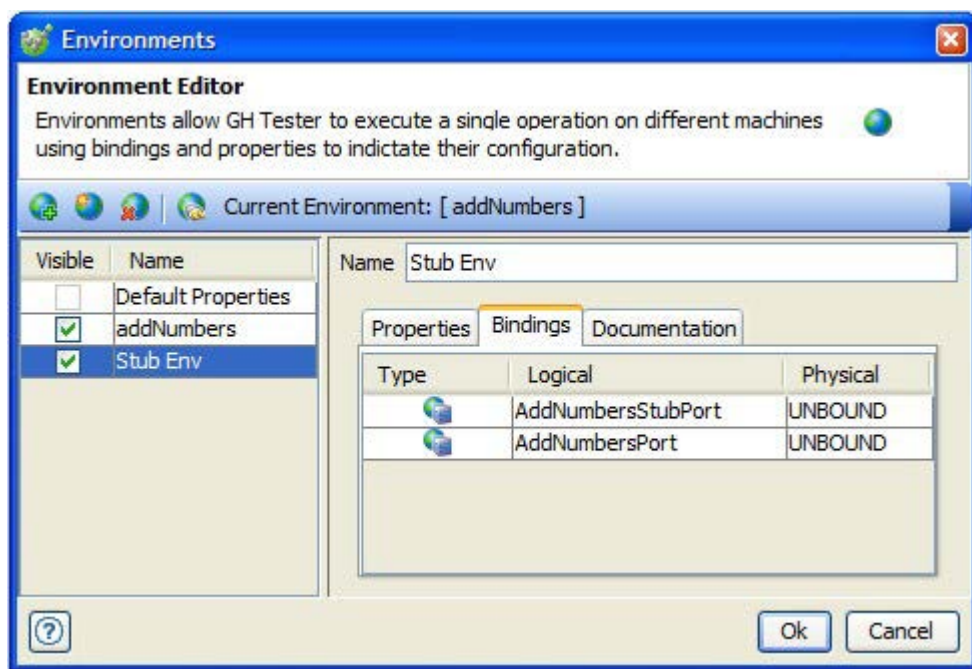
Namespace	Path
http://www.hotair.com/schema...	ns0:bookFlightResponse/ns0:newReservationNumb...
*	ns0:getReservationsRequest/ns0:weekNumber/text()

---

## 2.5 Environments

An environment is a set of variables (tags) and bindings that can be used to provide runtime configuration settings to test resources. By creating multiple environments (for example, UAT, Staging, Production), you can quickly run the same set of test resources against different phases of your product lifecycle. For example, environment tags can be used to modify messaging settings such as port numbers or connection details to JMS/MQ brokers.

A Rational Integration Tester project starts with no environment selected. The environment is accessed through the **Project > Edit Environment** menu option.



Environment variables are referred to as tags, and are accessed by surrounding the name with double percent signs, for example, `%%name%%`. So if you were to create an HTTP transport using a port number, you might configure the connection setting to use `%%HTTP_Port%%`. If you refer to a non-existent tag, it is shown underlined like this: `%%UnknownTag%%`.

The configuration of bindings in an environment, which define the relationship between logical and physical components in the project, is described in more detail later in the document.



# Testing a Web Service

## **Contents**

**Synchronizing With WSDLs**

**Verifying WSDL Resources**

**Creating Tests**

**Running the Tests**

**Creating Test Suites**

**Viewing Test Reports**


This chapter describes how to import a WSDL into Rational Integration Tester, and how to create and run tests based on the operations and input parameters defined in the WSDL.

---

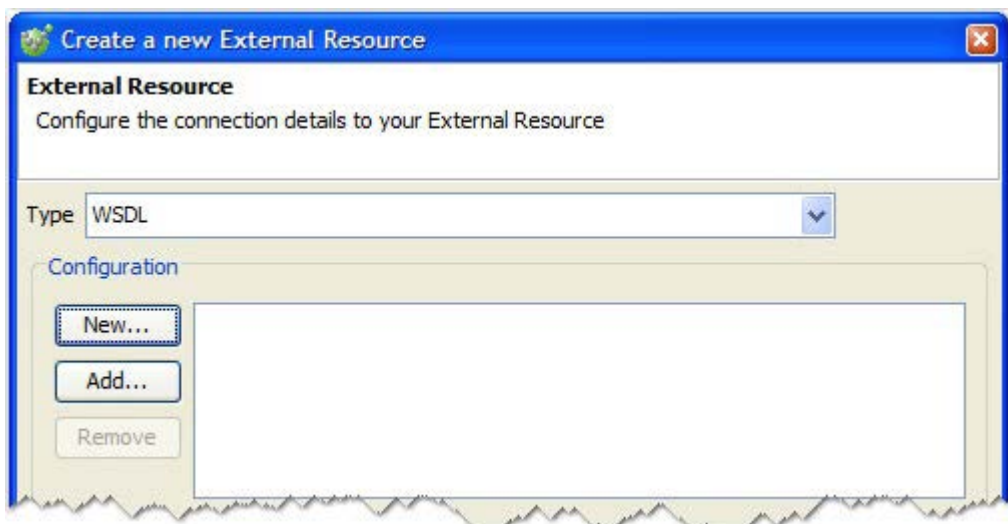
## 3.1 Synchronizing With WSDLs

Synchronisation enables you to add external resources to your project and automatically create all of the artefacts that are needed to test the resource. An example web service (AddNumbers) is hosted at <http://demos.greenhat.com:8088/addNumbers?wsdl> on the Green Hat web site. This web service will be used to illustrate various features in Rational Integration Tester.

**NOTE:** In addition to WSDLs, synchronisation functionality extends to webMethods, TIBCO BusinessWorks projects and Design Time Libraries, SAP, and SCA Domains (Oracle Fusion).

1. Open Rational Integration Tester's Architecture School perspective and select the Logical, Physical, or Synchronisation view.
2. Click the  icon to add a new external resource, or select **WSDL** from the **Web** menu in the Logical or Synchronisation views.

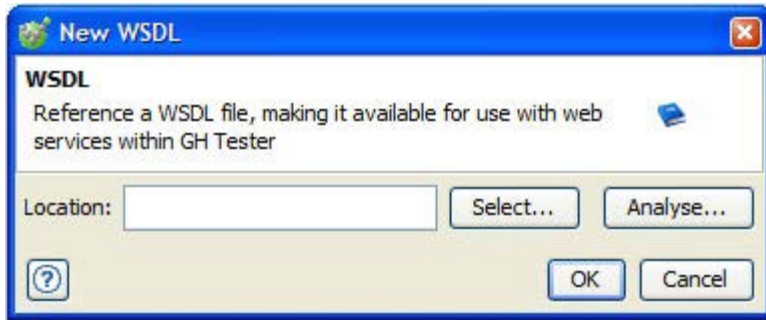
The **External Resource** wizard is displayed.



3. If not already present, select **WSDL** from the **Type** menu.
4. To add a new WSDL, click **New** under **Configuration**.

---

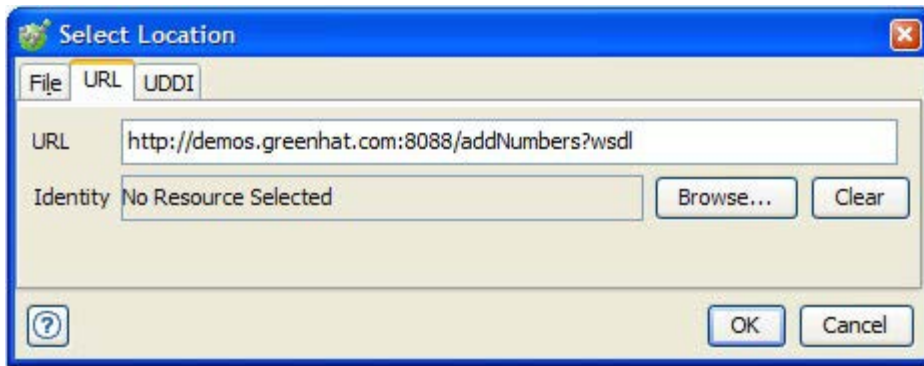
The **New WSDL** dialog is displayed.



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

The **Select Location** dialog is displayed.

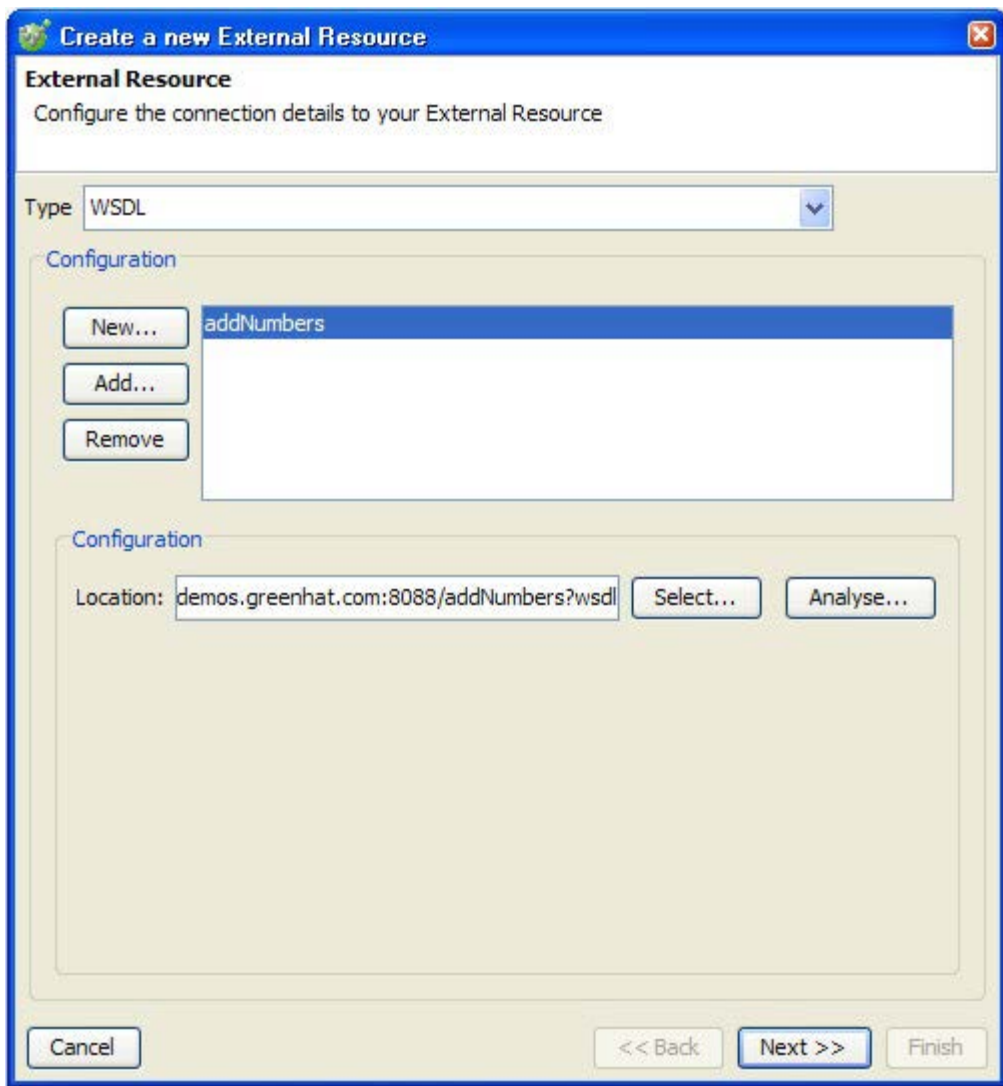
Click the **URL** tab and enter or paste the URL of the WSDL document you want to import (<http://demos.greenhat.com:8088/addNumbers?wsdl>).



**NOTE:** If the WSDL is hosted on a password protected server, you can specify the logon credentials using an existing Identity resource by clicking **Browse** next to the **Identity** field (refer to *IBM Rational Integration Tester Reference Guide*).

6. Click **OK** to save the change and close the dialog.

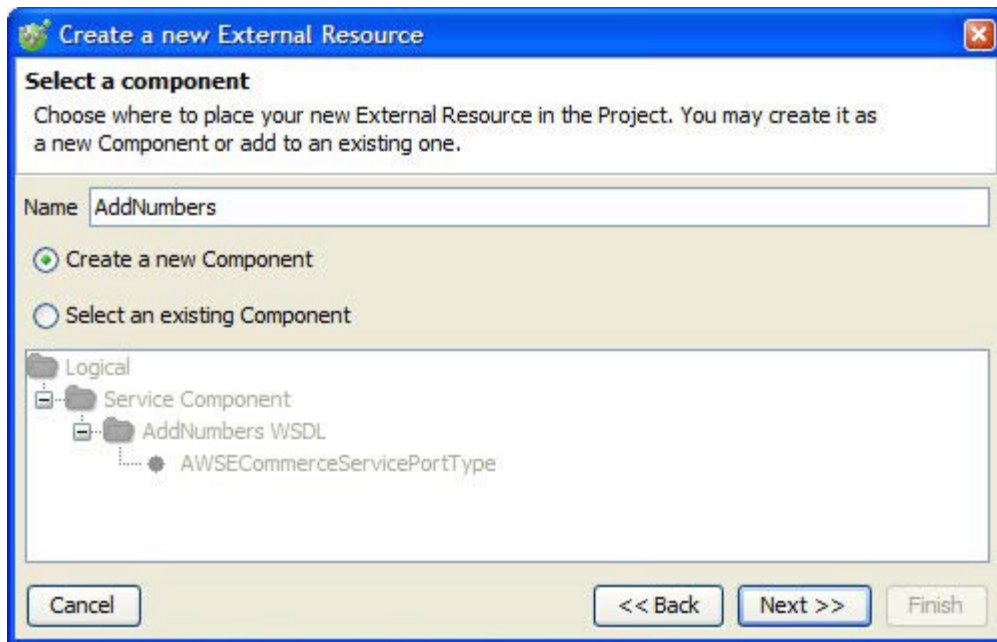
- 
- Click **OK** in the **New WSDL** dialog to return to the **External Resource** wizard.



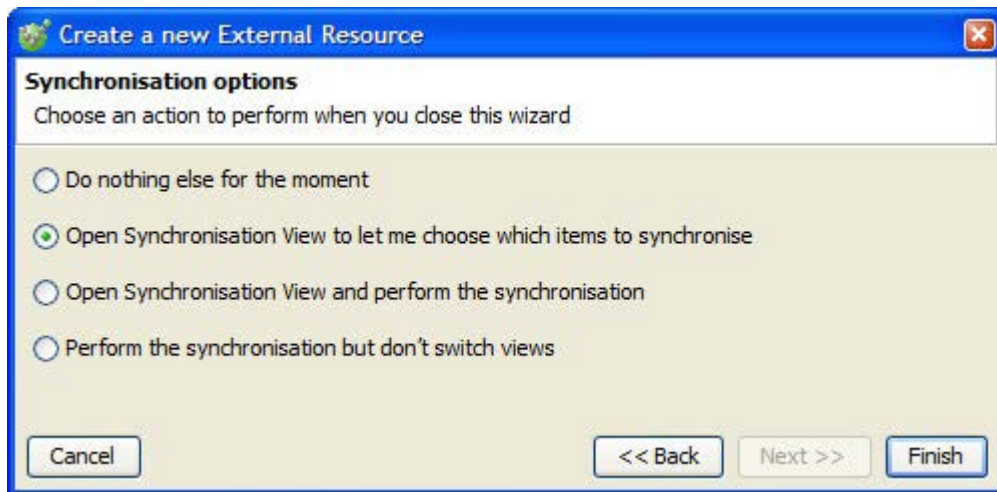
- Click **Next** to proceed.

---

In the **Select a component** dialog, you must select where the WSDL artefact should be created within the project – as part of an existing service component or as a new component. All of the resulting test artifacts (following synchronisation) will be created in this component.



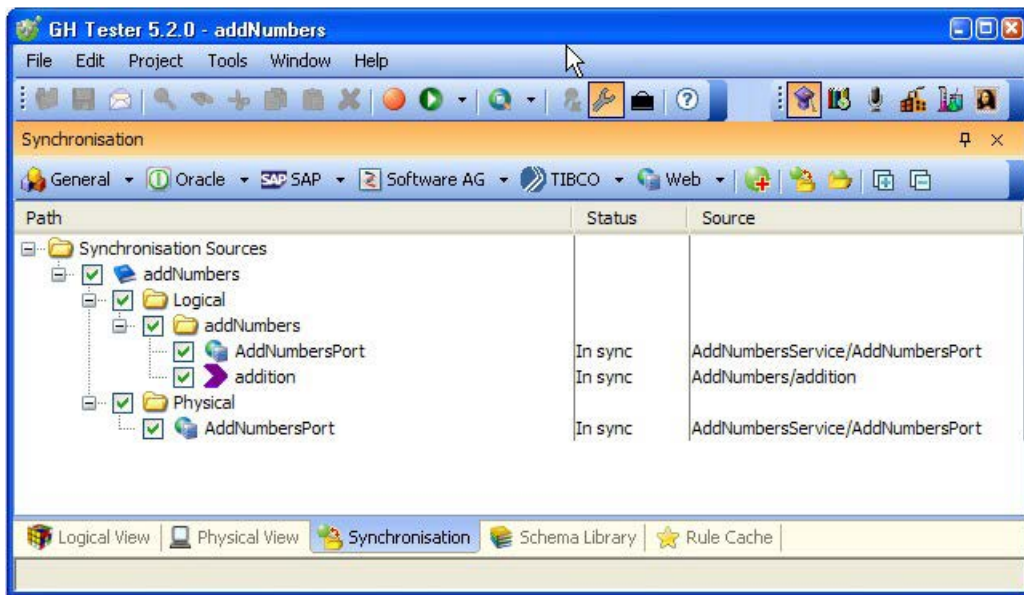
9. Select the **Create a new Component** option and click **Next**.



10. In the **Synchronisation options** dialog, select the second option and click **Finish**.

---

The **Synchronisation** view is displayed in Architecture School.



11. Click on the **Synchronise** button  to create a local copy (in Rational Integration Tester) of all the WSDL artefacts.

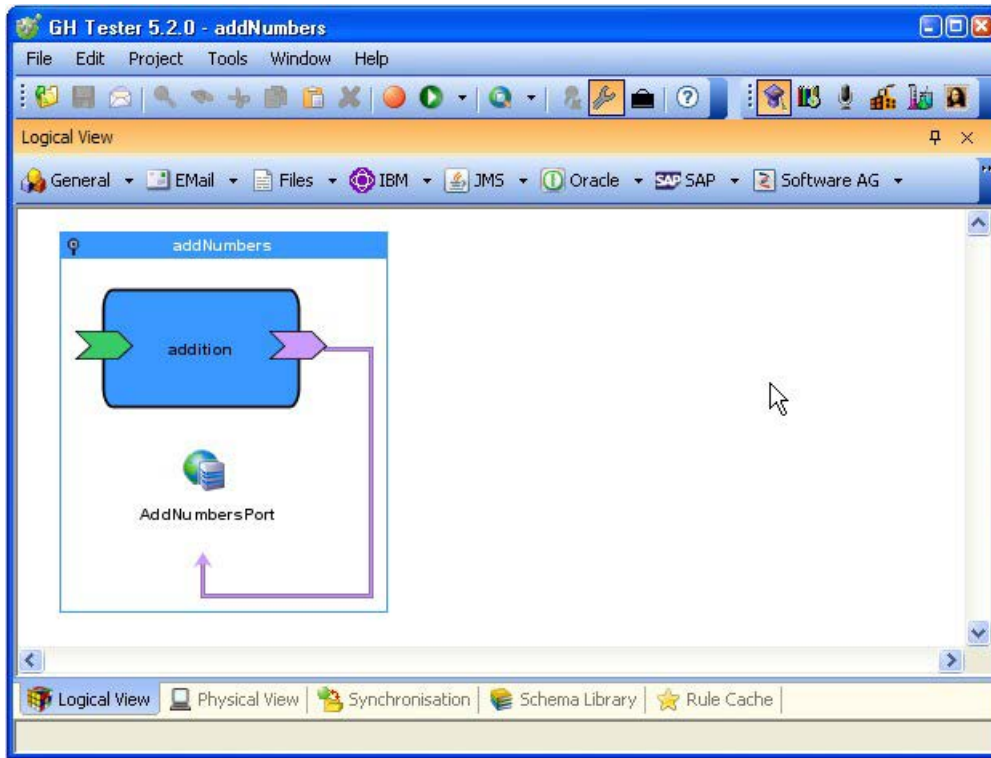
**NOTE:** The status for all of the test artefacts should change from “No local copy exists” to “In sync” once the synchronisation is complete.

12. Proceed to [Verifying WSDL Resources](#) to verify that the required test artefacts have been created.

---

## 3.2 Verifying WSDL Resources

Select the Logical View in Architecture School and you should see the components and operations shown below.

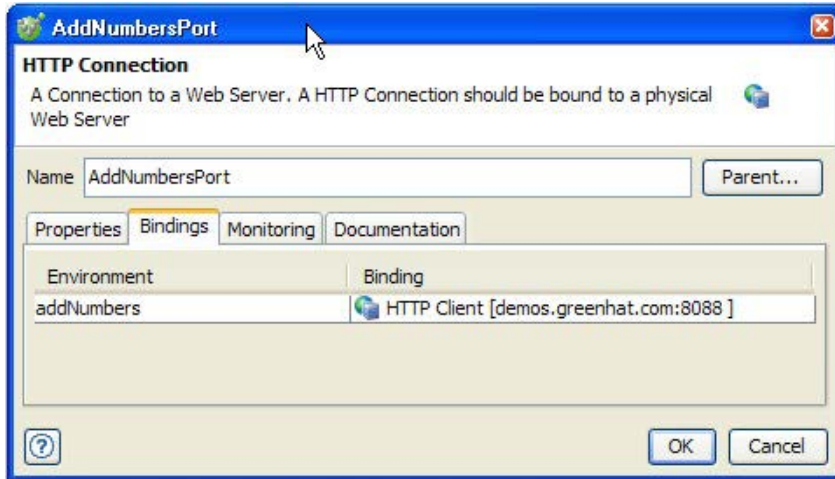


Rational Integration Tester has created a service called **addNumbers**. Within the service, an **AddNumbers** service component has been created based on the port type in the WSDL. Within this service component, an operation has been created based on the name of the operation in the WSDL, which is **addition**.

Clicking on the addition operation shows a dependency on the **AddNumbersPort** HTTP connection, which is highlighted by the purple arrow.

---

The **AddNumbersPort** connection relates to the logical view of the transport. Double-click the connection to display its properties, then click on the **Bindings** tab to see its binding to the physical artefact, HTTP Client.





Double-click the **addition** operation to view its properties (for example, the Message Exchange Pattern, which defines the operation as a request/reply and defines the schemas for the request and reply messages and the transport used to send the data).

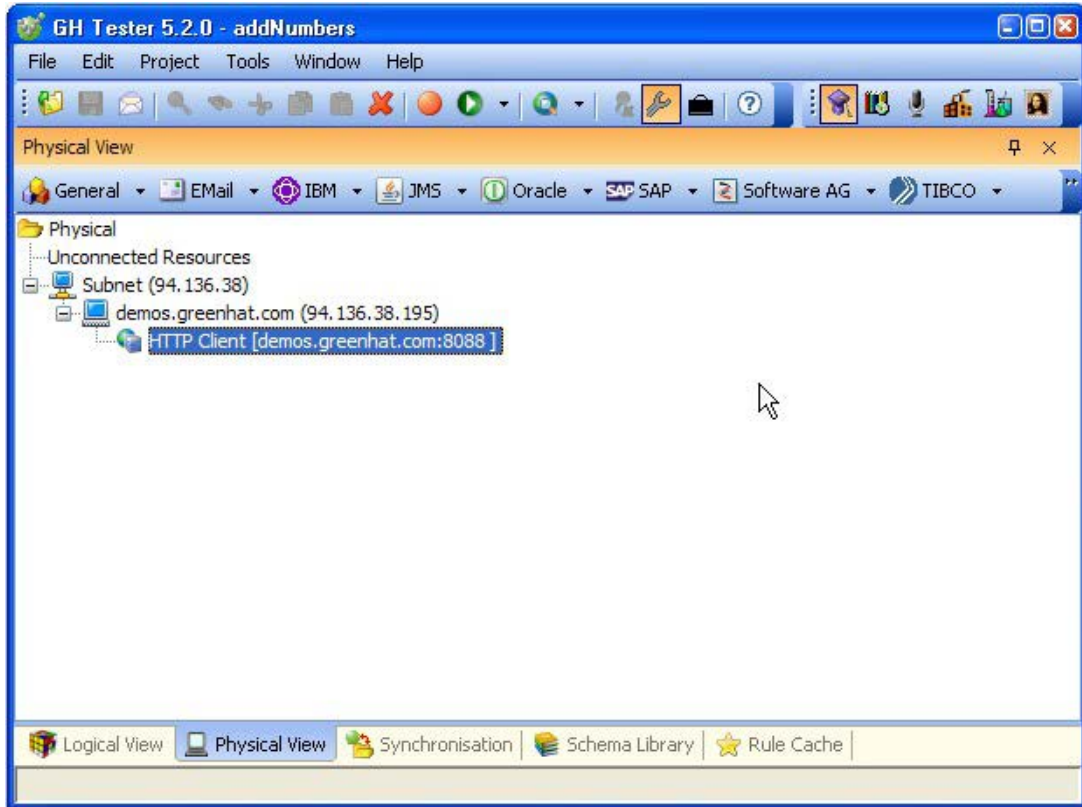
The screenshot shows the 'addition' operation configuration dialog. The 'Operation' tab is active, displaying the following settings:

- Name:** addition
- Message Exchange Pattern:** Request/Reply
- Schema:**
  - Request:** addNumbers (Root: addition\_\_INPUT\_\_addition)
  - Reply:** addNumbers (Root: addition\_\_OUTPUT\_\_additionRe:)
- Binding:**
  - Transport:** AddNumbersPort
  - Message Type:** Text
  - HTTP Properties:**
    - Resource Name:** /addNumbers
    - HTTP Method:** POST
    - Follow Redirects:** ☒

The dialog includes 'Parent...', 'Browse...', and 'Clear' buttons for various fields, and 'OK' and 'Cancel' buttons at the bottom.

---

In Architecture School's Physical View you should see one artefact called **HTTP Client** under `demos.greenhat.com`. This is the HTTP transport that Rational Integration Tester will use to send and receive messages to and from the web service.



At this point you are ready to create and run a few simple tests.

---

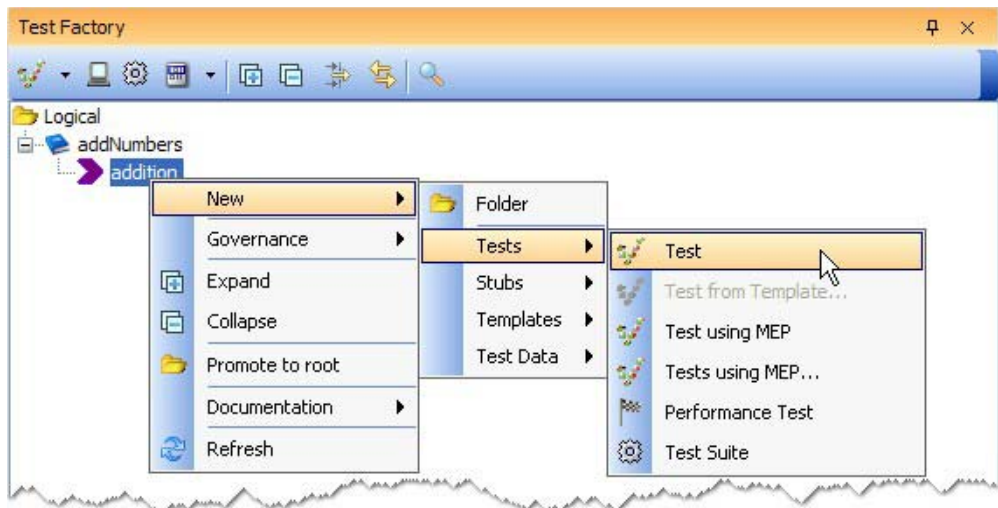
## 3.3 Creating Tests

Tests can be created in two ways – from scratch or with a wizard that automatically creates tests for you based on the containing operation’s message exchange pattern (MEP). Both options will be illustrated in the following sections.

### 3.3.1 Creating Tests from Scratch

Follow the steps below to create (empty) tests from scratch:

1. Open the Test Factory perspective (**F10**).
2. Right-click on the **addition** operation and select **New > Tests > Test**.

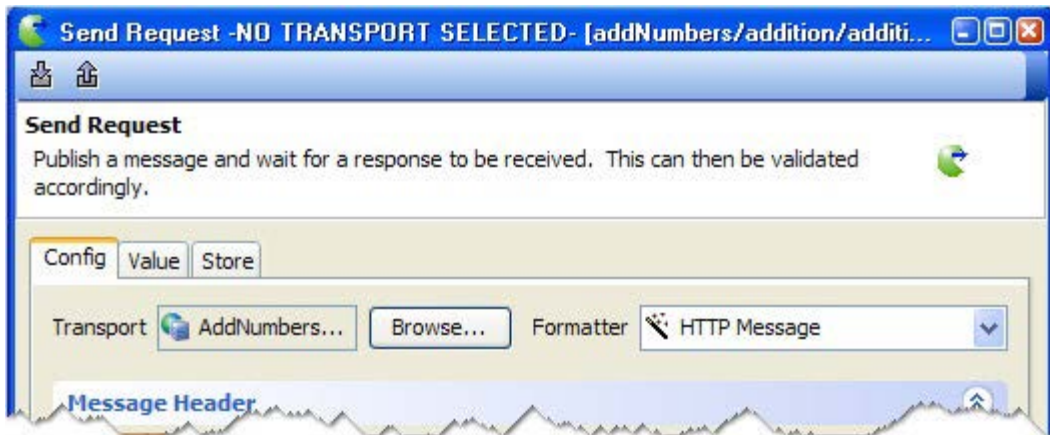


3. When prompted, name the test “additionCheck”, then click **OK**.
4. Double-click the new test in the Test Factory tree to open it for editing.
5. In the additionCheck test editor on the right-hand side of the workspace, right-click the **Test Steps** phase and select **New > Messaging > Send Request**.

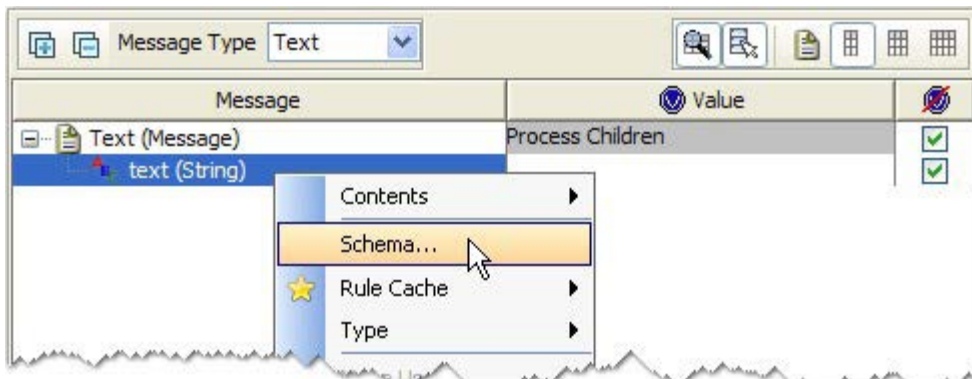
A **Send Request** and **Receive Reply** action should be created.

**NOTE:** Actions placed in the **Initialise** and **Tear-down** sections of a test will run, but these phases are only executed once when running multiple iterations of a test (at test startup and completion). These sections are typically used for one-time test operations, such as clearing database tables or moving files.

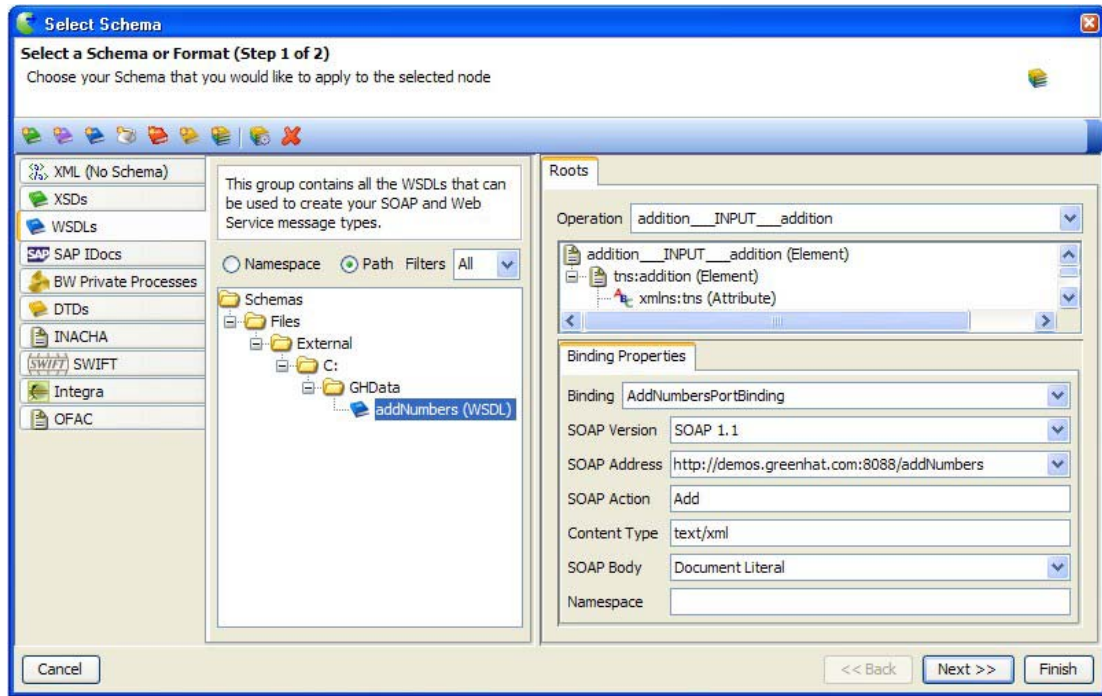
6. Double-click the **Send Request** action to open it.
7. Ensure that the Transport is set to **AddNumbersPort** and the Formatter to **HTTP Message**.



8. In the message configuration area, ensure that the **Message Type** is set to **Text**, then right-click on the **text (String)** node and select the **Schema** option.

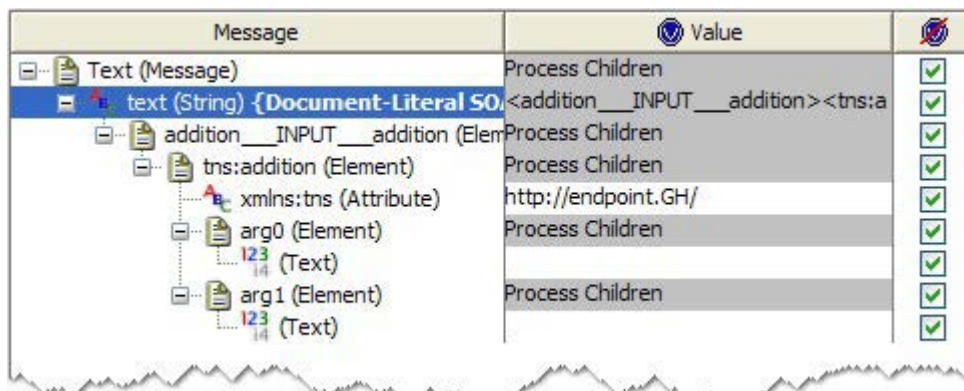


9. In the first dialog of the **Select Schema** wizard, select the **WSDL** tab, select the **AddNumbers** WSDL and select the **addition\_\_INPUT\_\_addition** operation, then click **Next** to proceed.

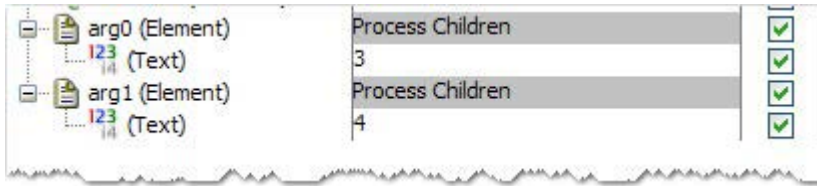


10. In the next window, ensure the **Include Text nodes** option is enabled (ticked) in the **Content** pane, then click **Finish**.

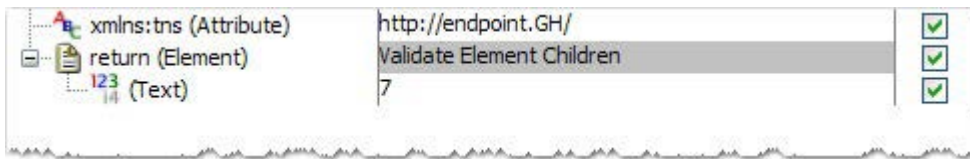
The structure and content of the message body will be updated according to the selected schema.




- 
11. Set the values of the **arg0** and **arg1** text nodes to 3 and 4 (double-click the empty field under the **Value** column and type in the values).



12. Click **OK** to save your changes and close the editor.
13. Double-click the **Receive Reply** action to open it and ensure **Reply to** is set to **Send Request 1**.
14. Apply the same schema to the message (as described in steps 8 through 10), except this time choose **addition\_\_output\_\_additionResponse** operation.
15. Enter “7” in the **return** element text node, which is the sum of **arg0** and **arg1**.



16. Click **OK** to save your changes and close the editor.
17. Save the project (click  or press **Ctrl + S**) and the test is now ready to run.



---

### 3.3.2 Creating Tests using MEP

An operation's Message Exchange Pattern (MEP) defines the pattern, schema, and binding for messages that are exchanged by the operation. This information is used by the MEP wizard to quickly create tests.

You can create a single test using an operation's MEP, or multiple tests. The singular option quickly creates one test that you can populate and run, while the multiple option creates a number of structural tests according to content combinations that are based on the schemas defined in the WSDL. The MEP wizard can populate the tests with a variety of test data (for example, random data values, constants, regular expressions, and so on) that is defined by the schema.

The following example illustrates how to use the MEP wizard.

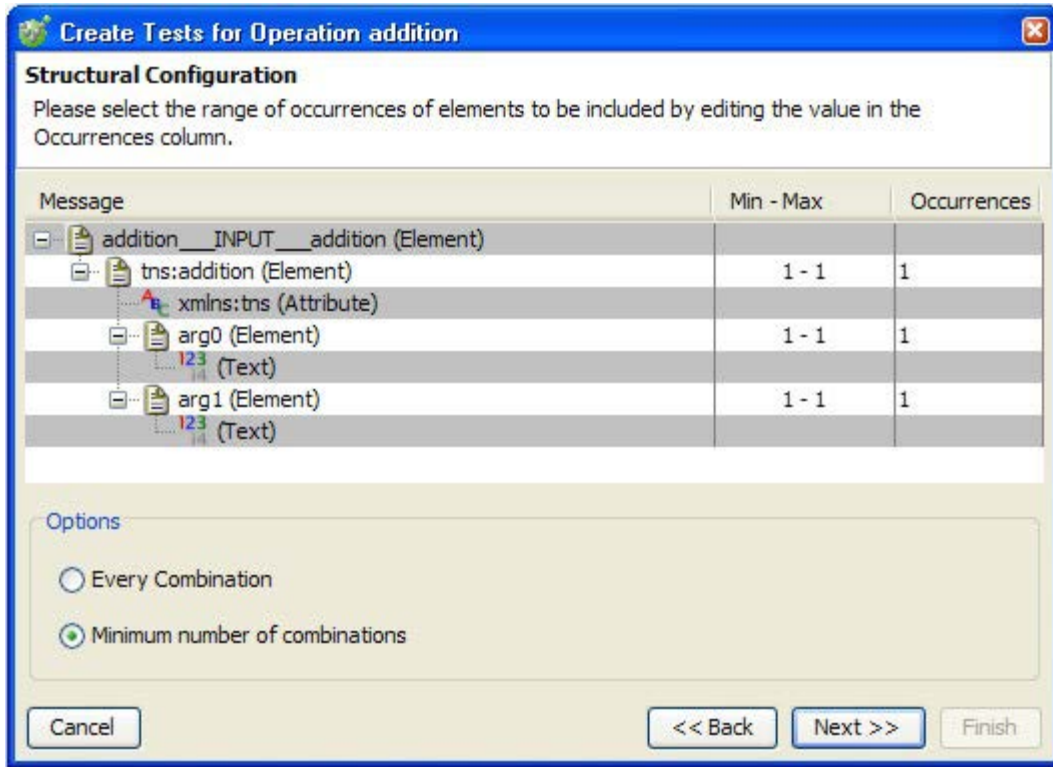
1. Open the Test Factory perspective (**F10**).
2. Right-click on the **addition** operation and select **New > Tests > Tests using MEP**.

The **Create Tests** wizard is displayed. The first window of the wizard enables you to configure the binding properties for the request and reply messages.

The screenshot shows the 'Create Tests for Operation addition' wizard window. The title bar reads 'Create Tests for Operation addition'. The main heading is 'Binding Configuration' with the instruction 'Please select the bindings for the message.' Below this, there are two sections: 'Request Binding Properties' and 'Reply Binding Properties'. Each section contains the following fields: Binding (AddNumbersPortBinding), SOAP Version (SOAP 1.1), SOAP Address (http://demos.greenhat.com:8088/addNumbers), SOAP Action (Add), Content Type (text/xml), SOAP Body (Document Literal), and Namespace (empty). At the bottom of the window, there are four buttons: 'Cancel', '<< Back', 'Next >>', and 'Finish'.

- 
- Click **Next** to proceed as we do not need to change the message bindings.

The **Structural Configuration** screen enables you to set the number of occurrences for each message element (0 through  $n$ ) and select the desired combination option for generating tests.



- If **Minimum number of combinations** is selected, at least one unique element will be generated in a message for each occurrence, but only enough tests will be generated to satisfy the highest single occurrence value (that is, if the most occurrences for any element is three, then three tests would be generated).
  - If **Every Combination** is selected, a test will be created so that every unique combination of occurrences is satisfied (that is, if one element specifies two occurrences and another specifies three, then six tests would be generated).
- Leave the default number of occurrences and combination options, then click **Next**.



The **Contents Configuration** screen allows you to select the type of content to be populated in the text nodes of the message.

**Create Tests for Operation addition**

**Contents Configuration**

Please select the type of contents to be used by selecting one of the options in the Type column. Enter a value in the Value column for relevant types.

Message	Type	Value
addition__INPUT__addition (Element)		
tns:addition (Element)		
xmlns:tns (Attribute)		
arg0 (Element)		
123 (Text)	Constant	4
arg1 (Element)		
123 (Text)	Constant	5

**Options**

☐ Every Combination

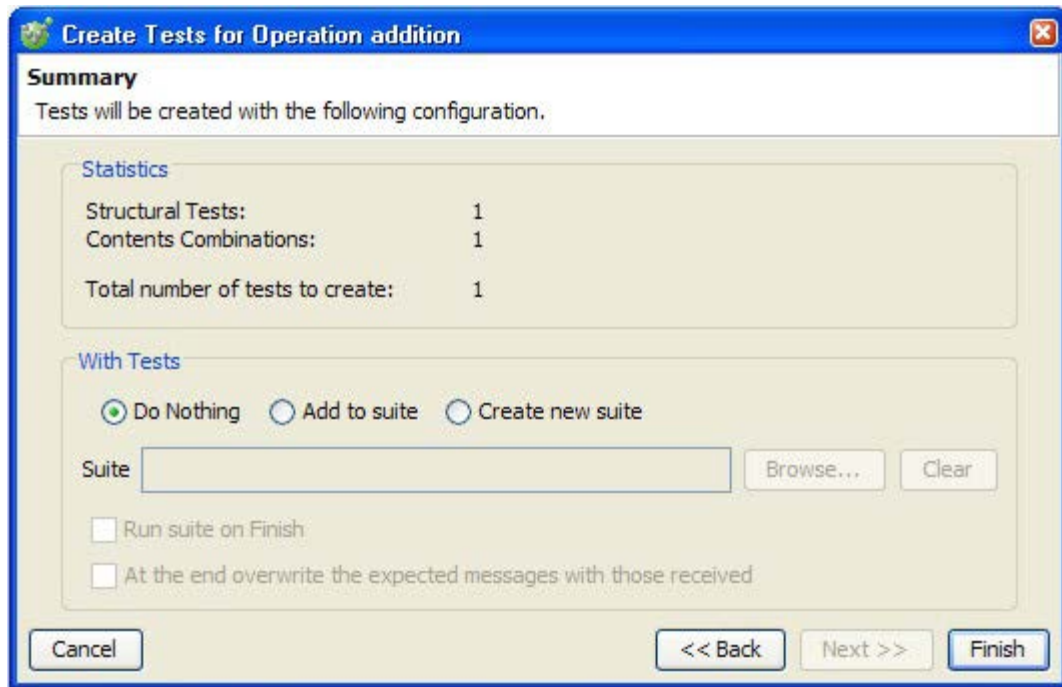
☒ Minimum number of combinations

Cancel    << Back    Next >>    Finish

- For the **arg0** and **arg1** text fields, change the type to **Constant** and enter values of “4” and “5”, respectively.
- Click **Next** to proceed.

---

A summary is displayed, showing the number of tests that the wizard will create.



7. For this example, select **Do Nothing** under **With Tests** to simply create the tests.

**NOTE:** The MEP test can be added to an existing test suite or added to a new test suite, as described in [Create a Test Suite with the MEP Test Wizard](#).

8. Click **Finish** to close the wizard and create the tests as specified. You should see a new test called "Test" with pre-populated test data in the messages.
9. Rename the test "additionCheckMEP" (right-click and select **Rename**, or select the test and press **F2**). You can double-click the new test to open it for editing.

**NOTE:** The response message has not been populated with an expected value. The test repair wizard will be demonstrated in the next section to remedy this.

---

## 3.4 Running the Tests

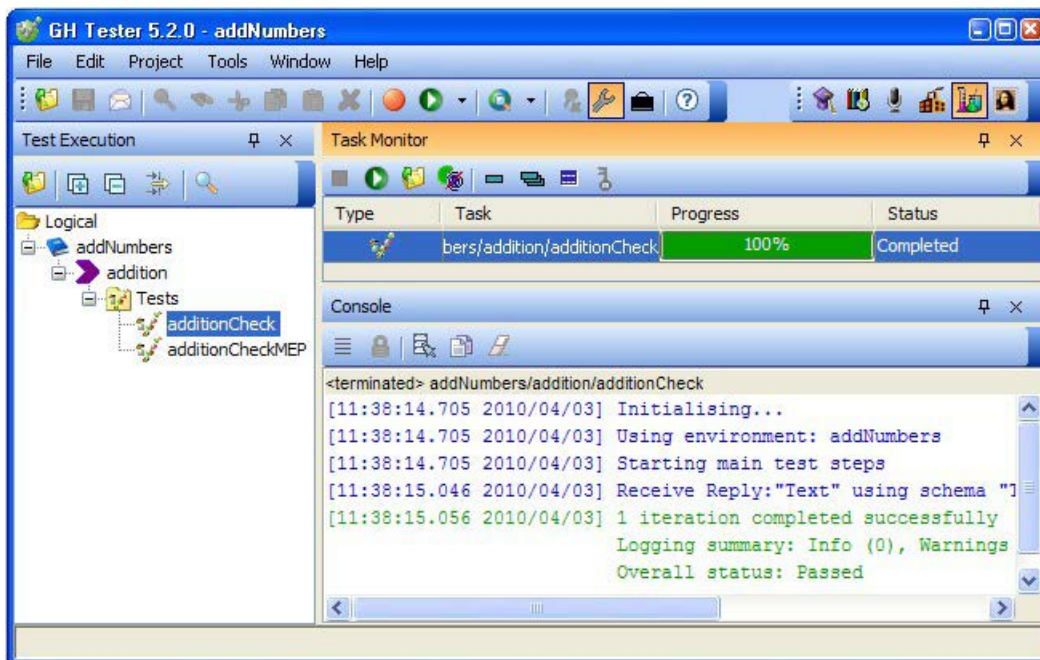
Tests – regardless of how they are created – are executed in the Test Lab perspective. This section describes how to run the two tests that were created previously.



### 3.4.1 Running the additionCheck Test

Follow the steps below to run the test made from scratch, **additionCheck**:

1. Navigate to the original WSDL URL and ensure that the sample web service is still running.
2. Open the Test Lab perspective (**F11**) and select the **additionCheck** test.
3. Right-click the test and select **Run**, or press **F5**.

While the test is running, the Task Monitor and Console will provide execution details. Once finished, the **Progress** bar should turn green and show “100%”, and the **Status** of the test should be “Complete”. In the Console, you should see a message indicating that the test has passed.



The Task Monitor shows the status of all current and previous tests, and the Console displays the detailed execution results for the currently selected run. If desired, you can remove selected test runs from the Task Monitor with the Delete icon , or clear all runs with the Delete All icon .

### 3.4.2 Running the additionCheckMEP Test

Follow the steps below to run the test made with the wizard, **additionCheckMEP**:

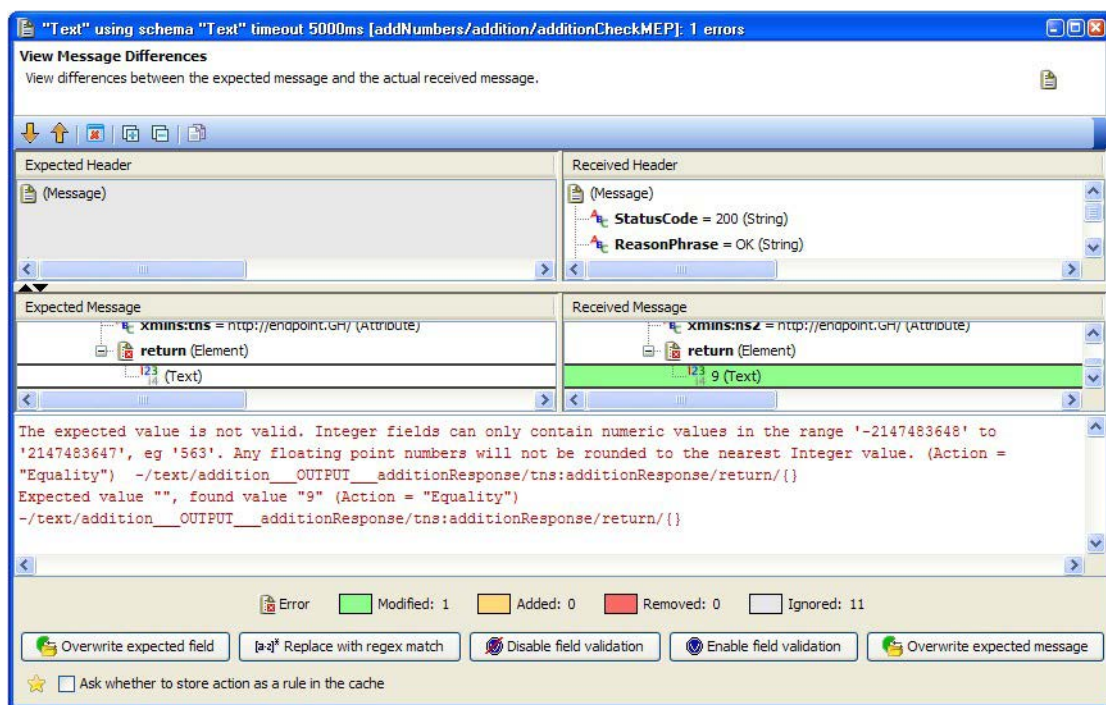
1. Run the test in the same way as described in [Running the additionCheck Test](#).

The test should fail (that is, not pass validation) because the value returned in the response does not match the expected result – remember this value was left blank. The following errors should be displayed in the Console for the Receive Reply action:

The expected value is not valid. Integer fields can only contain...  
Expected value "", found value "9" (Action = "Equality")


The test, however, can be repaired so that it passes the next time it is run.

2. Click the *"Expected value..."* line to display the **View Message Difference** window.



3. Select the highlighted line in the expected or received message to see error details.
4. From the available options, select **Overwrite expected field** and close the window.

**NOTE:** For information about the available repair options, refer to *IBM Rational Integration Tester Reference Guide*.

5. Run the test again by clicking the **Launch again** button  in the Task Monitor and the test should pass.

---

## 3.5 Creating Test Suites

Test suites are used to run sets of tests and test suites. For example, if you have a set of regression tests to run, you can group and run them together in a single test suite.

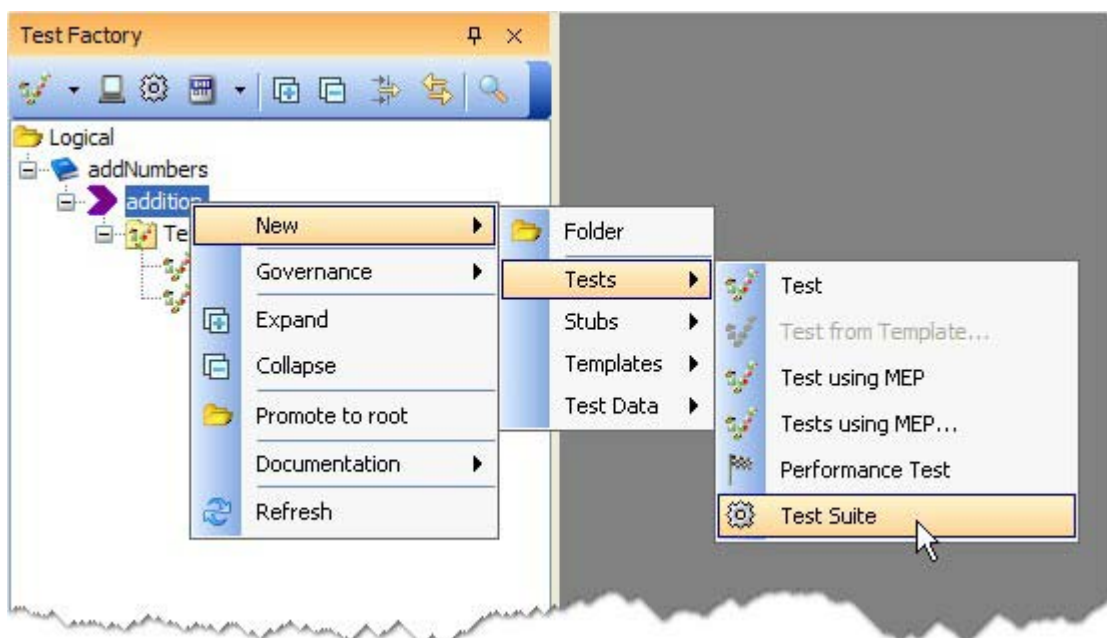
The following sections illustrate two different ways to create test suites:

- [Create a Test Suite Manually](#)
- [Create a Test Suite with the MEP Test Wizard](#)

### 3.5.1 Create a Test Suite Manually

The steps below illustrate how to create a very simple test suite that will run both of the tests that were created earlier.

1. Open the Test Factory perspective (**F10**).
2. Right-click on the **addition** operation and select **New > Tests > Test Suite**.

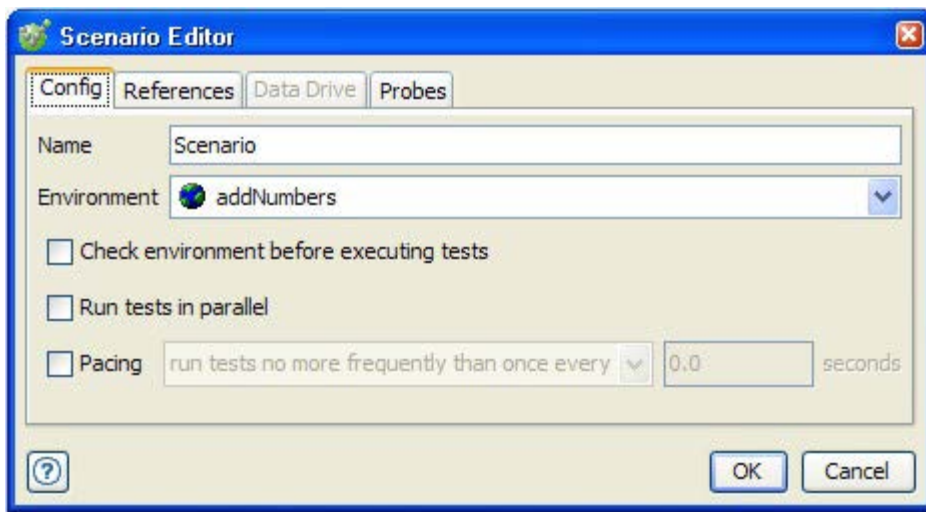



3. When prompted, name the suite “additionCheckTestSuite”, then click **OK**.
4. Double-click the new test suite to open it for editing.
5. In the suite editing window to the right, double-click **Scenario**.

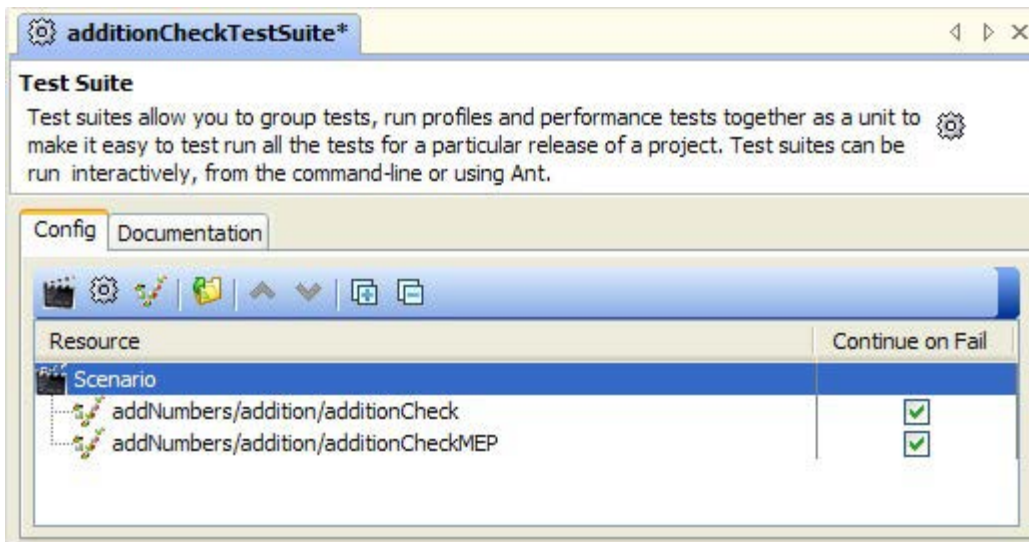



---

The **Scenario Editor** is displayed.



6. Under the **Config** tab, select the **addNumbers** environment and click **OK**.
7. Add the **additionCheck** and **additionCheckMEP** tests to the suite by dragging them from the **Tests** folder, or click the **Add Tests** icon  and select the tests from the project resource dialog.



8. Save the project (click  or press **Ctrl + S**) and run the suite (right-click and select **Run**, or select and press **F5**).

The Test Lab is opened and the suite is executed. The suite will run both tests, and you should see all three items (the suite and the two tests) pass.

---

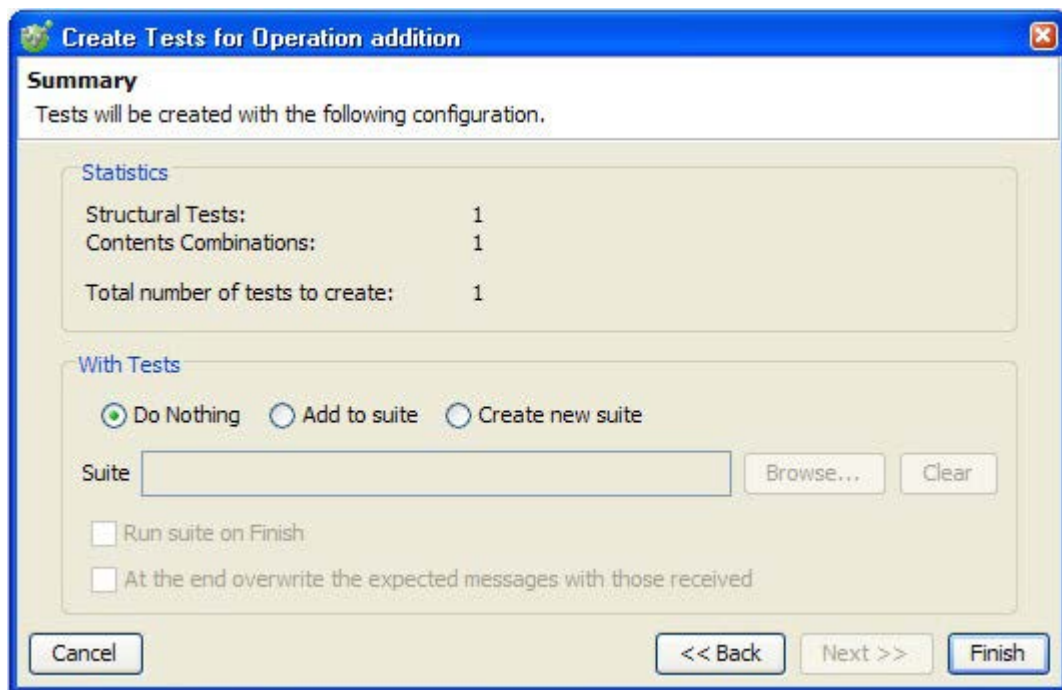
### 3.5.2 Create a Test Suite with the MEP Test Wizard

Earlier (in [Creating Tests using MEP](#)) we created a test based on an operation's message exchange pattern using the MEP wizard. In the last step of the wizard, you have the option to simply create the tests, add the tests to an existing test suite, or create a new test suite that contains the created tests.

The following steps illustrate how to create a new test suite using one or more tests created from MEP:

1. Return to [Creating Tests using MEP](#) and carry out steps 1 through 6, then return to this section and proceed.

The MEP wizard summary should be displayed, showing the number of tests that the wizard will create.



2. Select **Create new suite** under **With Tests** to create a new test suite that contains the test being created (**Do Nothing** simply creates the tests from the MEP wizard and **Add to suite** will add the new tests to an existing test suite, selected by clicking **Browse** and selecting the desired suite).

- 
3. Provide a name for the new suite in the **Suite** field.
  4. Enable any of the desired execution options under the suite name, as follows.
    - The **Run suite on Finish** option will execute a new or existing test suite when the MEP wizard closes.
    - The **At the end...** option will overwrite expected messages in any executed tests with the messages that are received.
  5. Click **Finish** to close the wizard and create the tests and test suite as specified. You should see a new test called “Test” and a new suite (with the name that you specified) containing the new test.



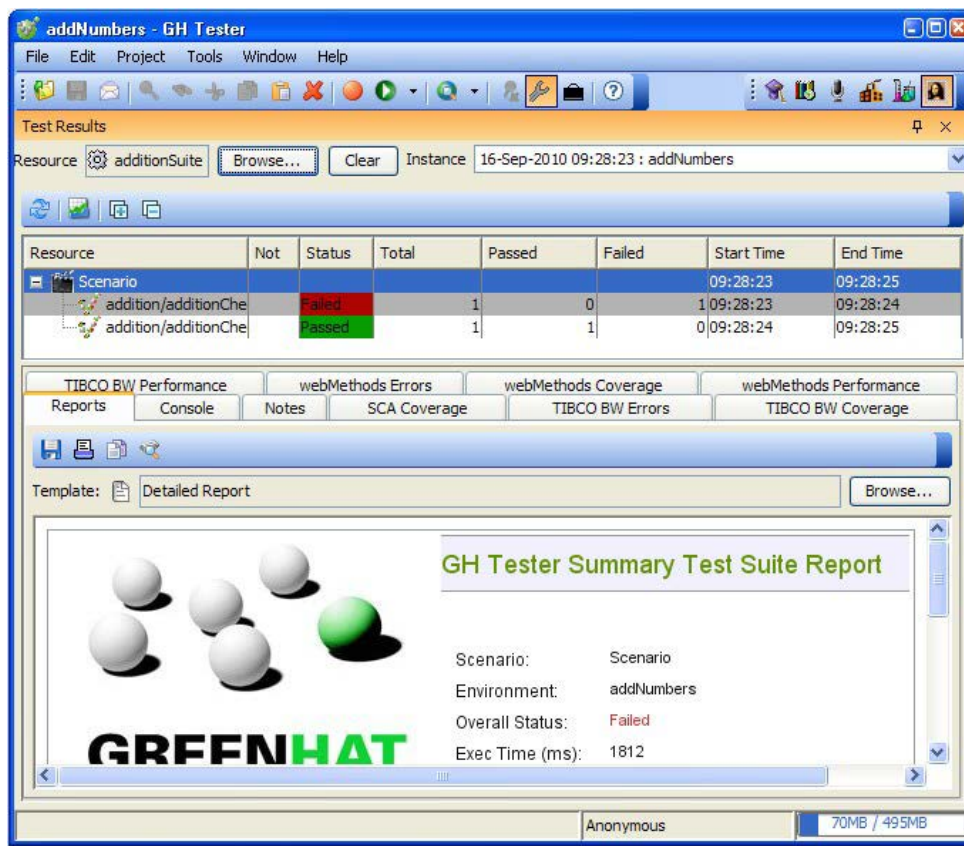
## 3.6 Viewing Test Reports

The Results Gallery enables you to view summary and detailed coverage, error, and performance reports for tests and suites that have been executed in the project.

**NOTE:** Test suite results can only be stored and viewed if the connection to the project database has been established properly. For more information, refer to *IBM Rational Integration Tester Installation Guide*.

Follow the steps below to view reports for the suite that was executed earlier.

1. Open to the Results Gallery perspective (**F12**).
2. The **additionCheckTestSuite** suite should already be selected in the **Resource** field. If not, click **Browse** and select it from the project resource dialog.
3. If the suite has been executed more than once, you can select the desired execution instance from the **Instance** field.
4. Ensure that the **Reports** tab is selected to view a summary report of the selected suite and execution instance.



5. Under the **Resource** column in the suite summary table, select the **additionCheck** test to view a detailed report of its results.



## GREENHAT

### GH Tester Detailed Test Report

Version: 5.1.1  
Environment: addNumbers  
Test: addNumbers/addition/additionCheck  
Overall Status: **Passed**  
Exec Time (ms): 2719

Start Time: 2009-04-08 15:12:29.614    Errors: 0  
End Time: 2009-04-08 15:12:33.398    Warnings: 0

#### Summary

Executed	Start Time	End Time	Exec Time (ms)	Status
<a href="#">Send Request: "addition__INPUT__addition" using sch</a>	15:12:29.634	15:12:29.644	10	<b>Passed</b>
<a href="#">Receive Reply: "addition__OUTPUT__additionRespons</a>	15:12:33.380	15:12:33.380	0	<b>Passed</b>

#### Environment

Name: addNumbers

#### Test Steps

Start Time: 15:12:29.634    End Time: 15:12:33.392  
Exec Time (ms): 3758    Host Id: JDGebickilap

Send Request: "addition\_\_INPUT\_\_addition" using schema "addNumbers" via "AddNumbersPort"

Start Time: 15:12:29.634    End Time: 15:12:29.644  
Exec Time (ms): 10    Error: n/a  
Transport: AddNumbersPort    Formatter: GH Text

Header	Message
httpDetails	

The detailed test report is displayed, including the overall status of the test, any errors and warnings that were generated, execution times, and the detailed results of the Send Request and Receive Reply actions in the test.

© Copyright IBM Corp. 2001, 2012

44

# Data Validation

## **Contents**

### **Overview**


### **Exercise**

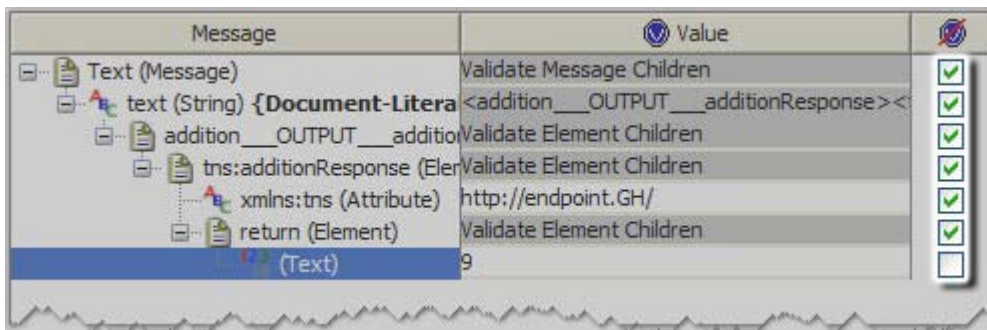
This chapter provides an overview of data validation in Rational Integration Tester, how it works, and how to configure it.

---

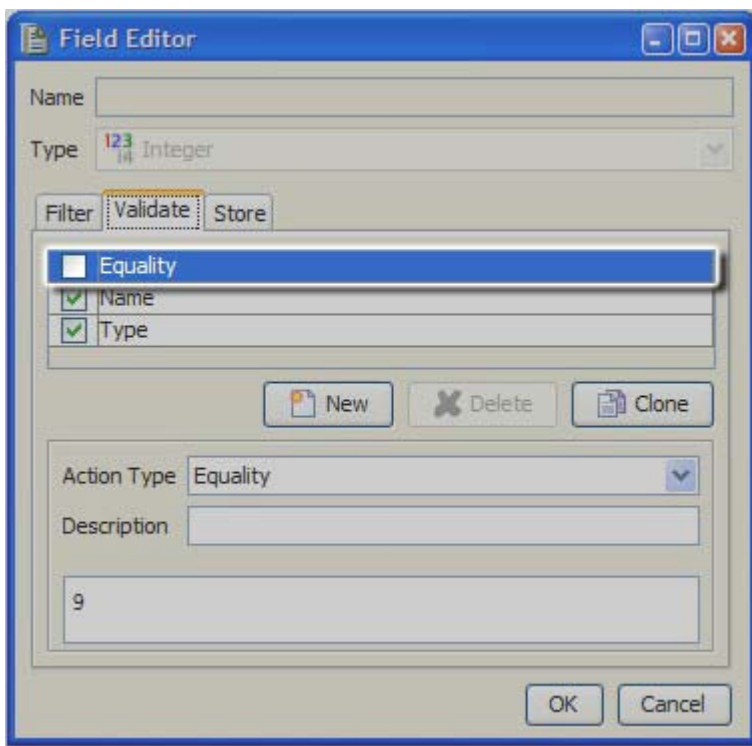
## 4.1 Overview

Data validation has been used throughout the exercises that have been illustrated in this guide. When validation is enabled for a selected field, the returned value is matched against the expected value.

Validation can be enabled or disabled for a message field by selecting or clearing the check box in the  column for that field, under the **Config** or **Assert** tab of the message editor.



Validation can also be controlled by enabling or disabling the **Equality** option, under the **Validate** tab, of the field editor (available by double-clicking a message element).

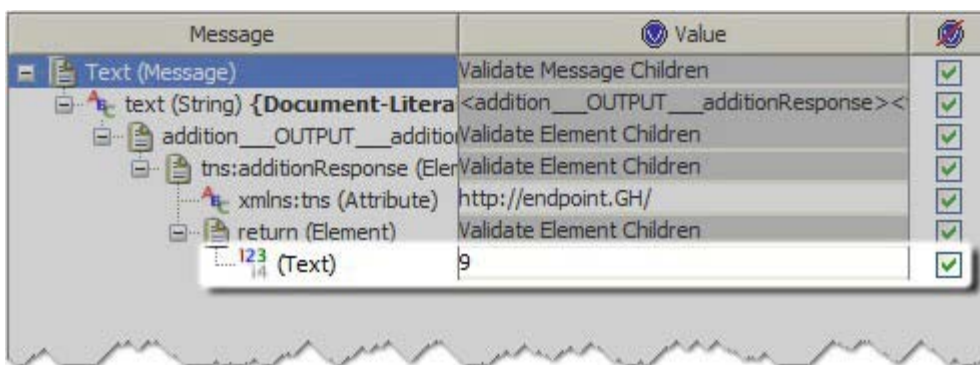



---

## 4.2 Exercise

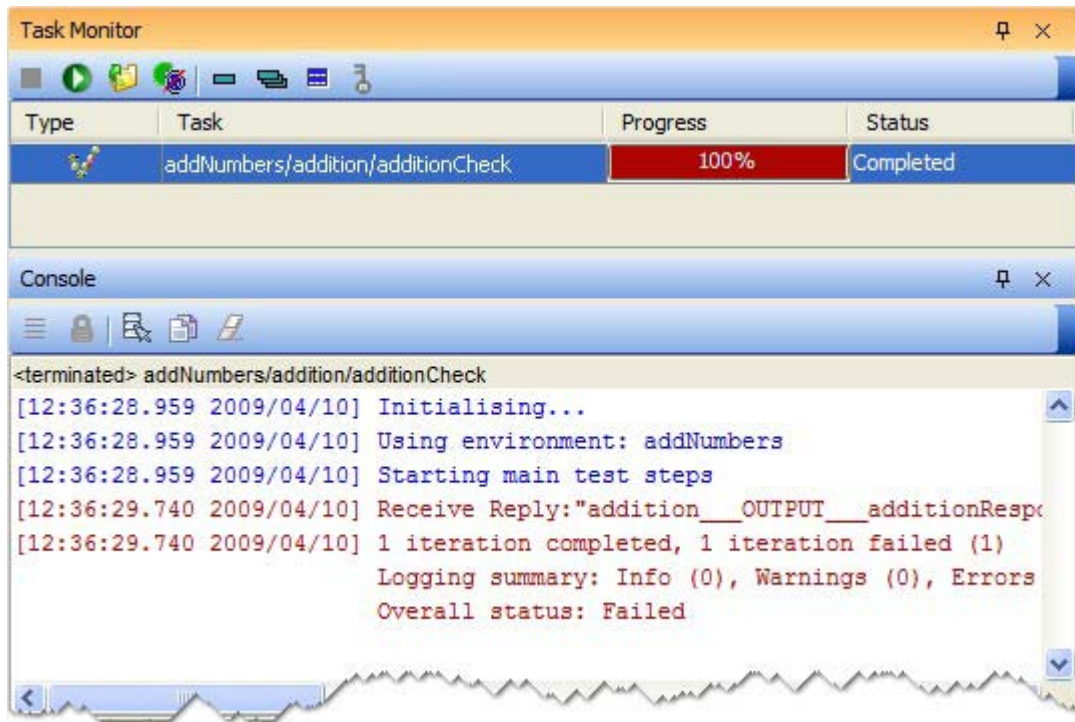
The following exercises illustrates how validation can be enabled or disabled for individual message fields, and the effect it has on whether or not the test passes.

1. Open the Test Factory perspective (**F10**)
2. Double-click the **additionCheck** test to open it for editing in the Test Factory perspective (**F10**).
3. Open the Receive Reply action and note the checked box in the validation column for the **return** text node.



4. Change the value of the node to any other number (that is, something that does not equal the sum of **arg0** and **arg1** in the Send Request action), then click **OK**.
5. Save the project (click  or press **Ctrl + S**).

- Run the **additionCheck** test in the **addNumbers** environment and the test should fail due to a validation error, as shown below.



The specific error for the Receive Reply action would be something like the following:

Expected value "99", found value "7" (Action = "Equality")

- Open the Receive Reply action again and clear the check box next to the same field to disable validation, or double-click the **(Text)** node and disable the **Equality** action in the field editor.

**NOTE:** If desired, as illustrated previously, you can disable validation in the **View Message Differences** dialog (see [Running the additionCheckMEP Test](#)).

- Click **OK** in the Receive Reply editor to close it and save your changes.
- Run the test again in the same environment and it should pass.

**NOTE:** In this example, the **Equality** action was used on a message field to validate the response, but numerous actions are available (for example, Xpath, Regex, Schema, Length, and custom validation). For more information about other validation options, refer to *IBM Rational Integration Tester Reference Guide*.

# Test Data Sets

## **Contents**

### **Overview**

### **Creating Data Sources**

### **Adding Log Actions to Tests**

### **Configuring Tests**

This chapter provides an overview of how to create data sets that can be used by Rational Integration Tester test resources to load test data dynamically.

---

## 5.1 Overview

So far, all of the data used in our example tests has been hard-coded. A more practical approach to testing, however, would be to decouple the tests from the data. One of the more common requirements of functional testing is to run a test through several iterations with different data being used each time. To do this, you will need to use test data sets. Data sets, along with the Iterate Test Data action, can be used to map test data into tags that are used to populate data fields in a test.

Test data can be supplied to Rational Integration Tester from four types of sources:

- Files (comma separated and fixed-width)
- Microsoft Excel spreadsheets
- Database queries
- Directory (a set of files containing XML or other data)



---

## 5.2 Creating Data Sources

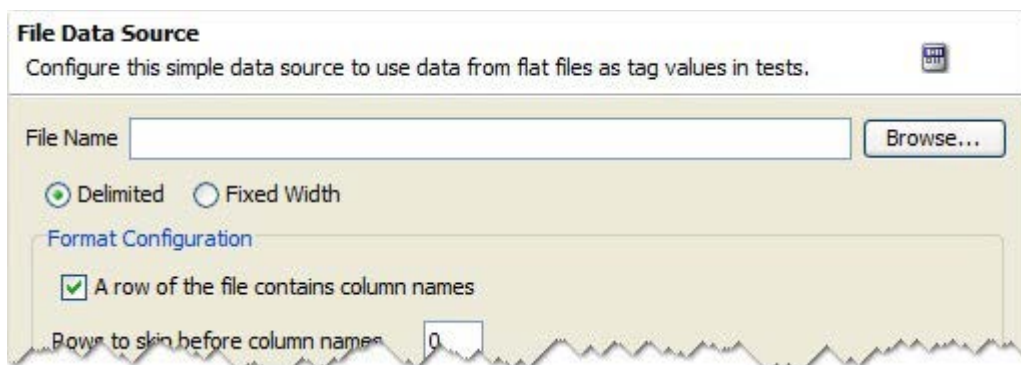
For our example, we will create a comma separated file that contains the data we will utilize in the **additionCheck** test.

1. Create and open a new text file and enter the following data in it:

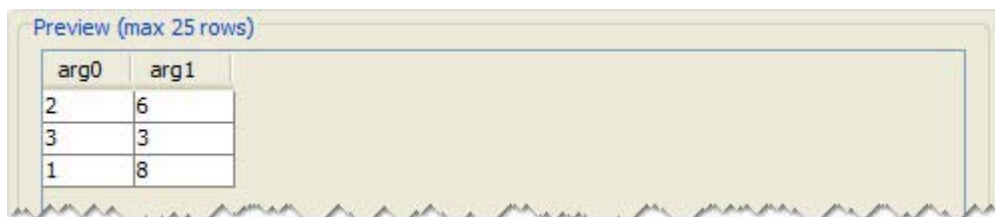
```
arg0,arg1
2,6
3,3
1,8
```


The fields in the first line – *arg0* and *arg1* – represent the column names.

2. Save the file as **data.txt** to your local drive.
3. In the Test Factory perspective (**F10**), right-click the **addition** operation and select **New > Test Data > File Data Source** from the context menu.
4. Enter “addNumbers” for the data source name and click **OK**. The properties of the new data source will be opened for editing to the right.



5. Click **Browse** next to the **File Name** field and select the data file you just created.
6. Leave all of the other settings as they are and click **Refresh** at the bottom of the data source editor – preview of the data in the selected file is displayed.




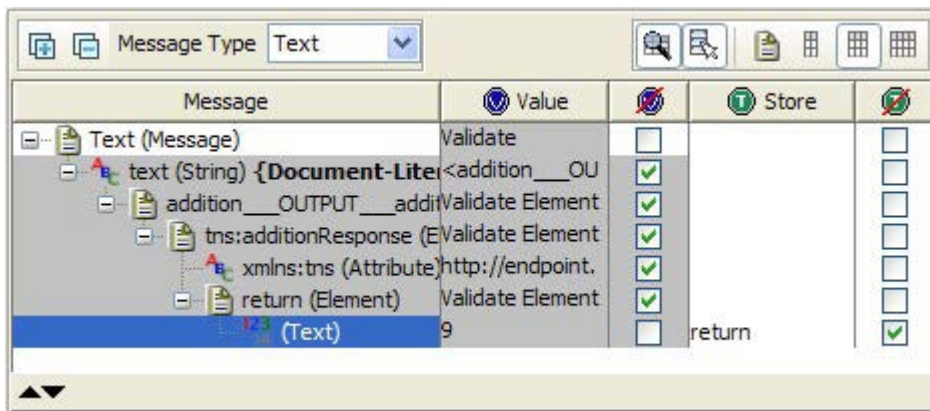
7. Click  or press **Ctrl + S** to save the new data source.

---

## 5.3 Adding Log Actions to Tests

Before configuring the test to use the new data source, we will add a Log action that will display the results of adding the arguments in the console.

1. In the Test Factory perspective (**F10**), double-click the **additionCheck** test to open it.
2. Right-click the **Receive Reply** action and select **New > Flow > Log** to add a log action.
3. Double-click the **Receive Reply** action and click the **Simple Editing View**  icon in the message editor.
4. Double-click the **Store** field next to the **Text** field (under the **return** element) and enter “return.”
5. Press **Enter** when finished.

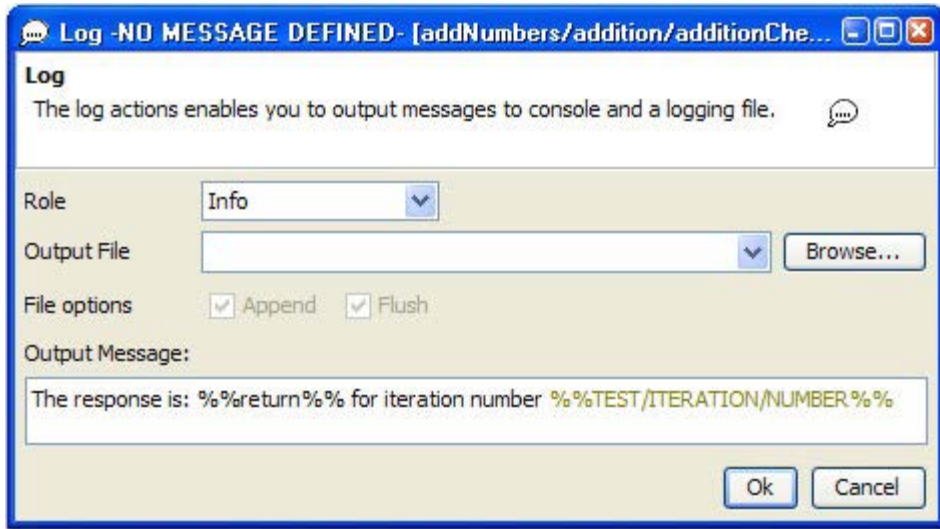


This will create a store action that will store the results of the field in a tag named `%%return%%`.

6. Click **OK** to apply the change and close the **Receive Reply** action.
7. To log the iterations as they are processed, open the Log step.
8. Enter “The response is: ”, then right-click at the end of the line and select **Insert Tag > Test Scope> return**.

- 
9. Now add “for iteration step” to the end of the line, then right-click at the end of the line and select **Insert Tag > Built-In > TEST > ITERATION > NUMBER**.

The Log step should now contain the following text: The response is %%return%%  
for iteration step %%TEST/ITERATION/NUMBER%%



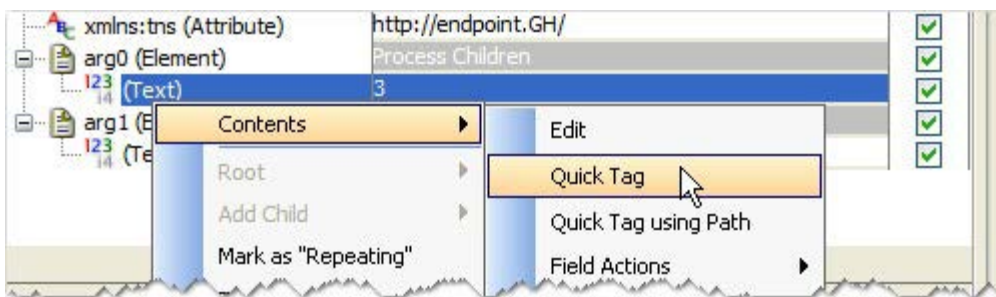
10. Click **OK** to save the test action and return to the Test Factory.

---

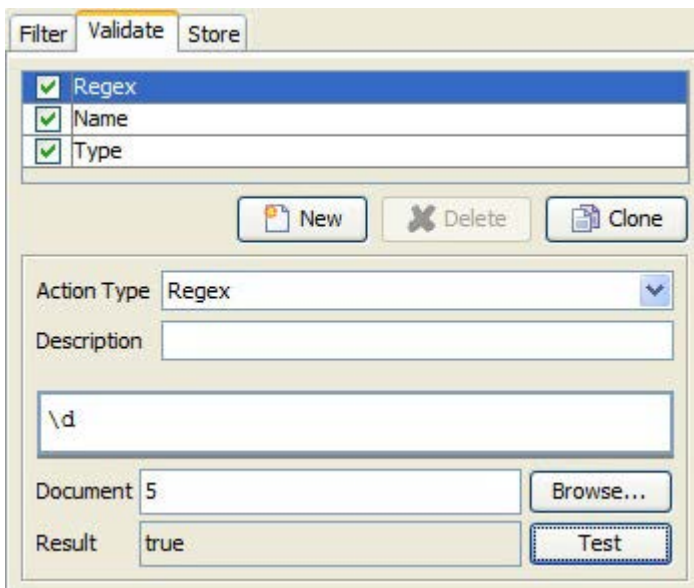
## 5.4 Configuring Tests

Now that the test data set is pointing to the new data source (our simple text file), we can configure a test to utilize the data that the source contains.

1. In the Test Factory perspective (**F10**), double-click the **additionCheckMEP** test to open it.
2. Double-click the **Send Request** action to open it.
3. Right-click on each of the text nodes of the **arg0** and **arg1** elements and select **Contents > Quick Tag** – click **Yes** when prompted to overwrite data.



4. Click **OK** when finished.
5. Open the **Receive Reply** action.
6. On the return text node, either disable validation or replace the **Equality** action with the **Regex** action and enter “\d” to match the expected result, then click **OK**.

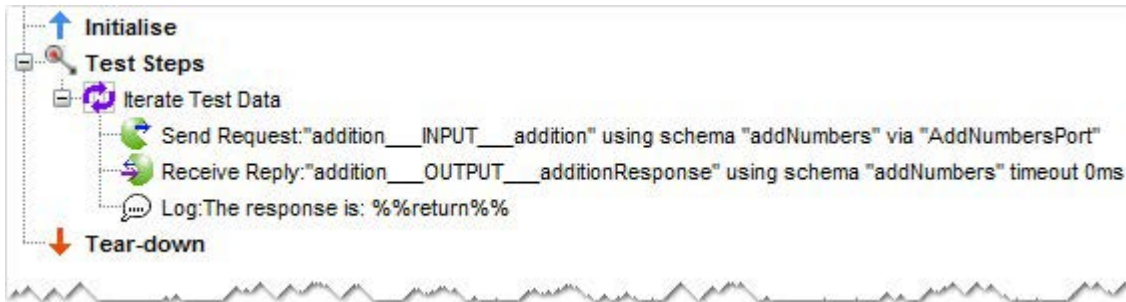


- 
7. Right-click the **Test Steps** phase and select **New > Flow > Iterate Test Data**.

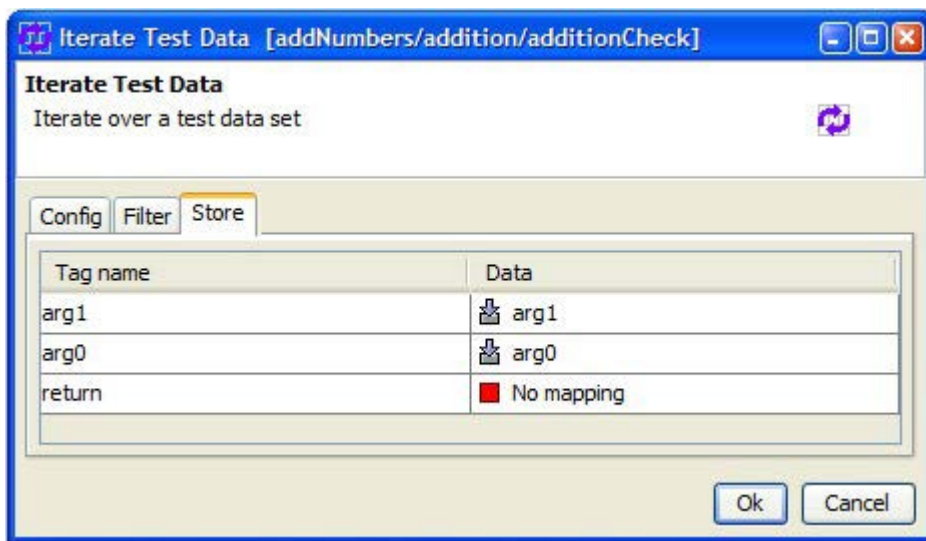
The new action should be created below the existing actions.

8. Select the Send Request, Receive Reply, and Log actions (using the **Ctrl** or **Shift** key) and drag them on to the new Iterate Test Data action.

The actions should now be inside the Iterate Test Data action, as shown below:



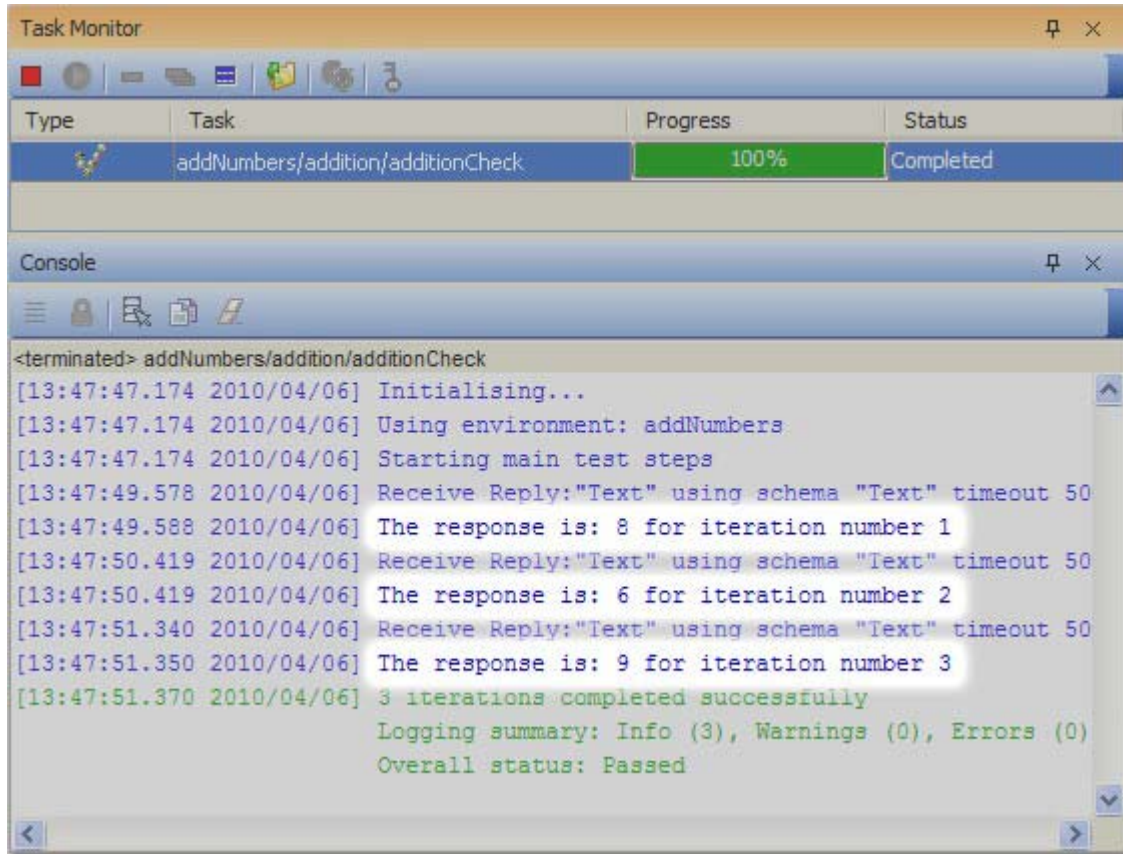
9. Double-click the Iterate Test Data action to open it for editing.
10. Under the **Config** tab, click **Browse** next to the **Test data set** field and select the **addNumbers** data set from the project resource tree.
11. Select the **Store** tab to verify the mappings from the data set to the test tags.



**NOTE:** The return node is unmapped since there is no such column in the test data set.

12. Run the **additionCheck** test.

The Iterate Test Data action iterates through the data in the data set and substitutes the tags according to the defined mappings. The test should pass and the results should be written to the console by the Log action.



# Functions and Decisions

## **Contents**

### **Overview**

### **Exercise**

This chapter provides an overview of how to use functions and decisions in Rational Integration Tester.

---

## 6.1 Overview

A fundamental requirement for some tests is to apply logic that tests something and takes alternative actions depending on the outcome. One of several built in or custom functions in Rational Integration Tester can be used to apply the logic, and a *Decision* action allows you to create one or more expressions that evaluate the outcome of the function. Two possible execution paths can be taken from the decision.

In this chapter we will create a trivial example that will test the value contained in a tag and generate an output a message to the user if the value is greater than 100. The first part of the test will prompt the user for a value and store it in a tag. This test will use the *User Interaction* action to solicit a response from the user. User interactions can be useful for debugging, as you can use them to display tag values at any point during the execution of a test.

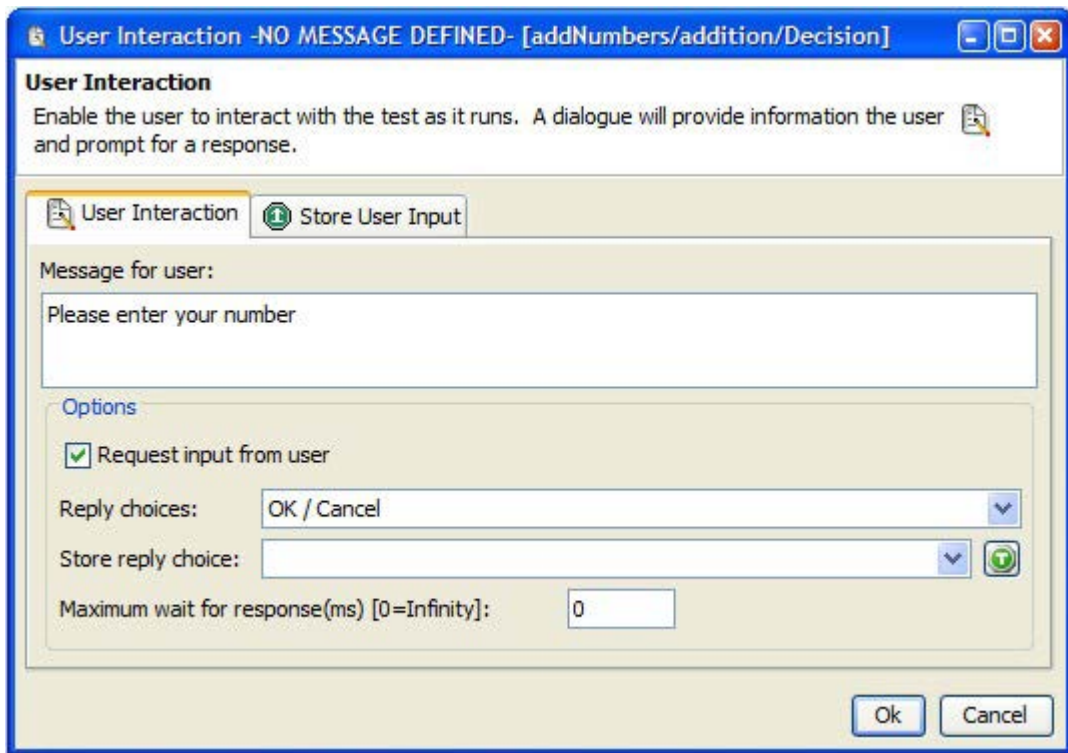


---

## 6.2 Exercise

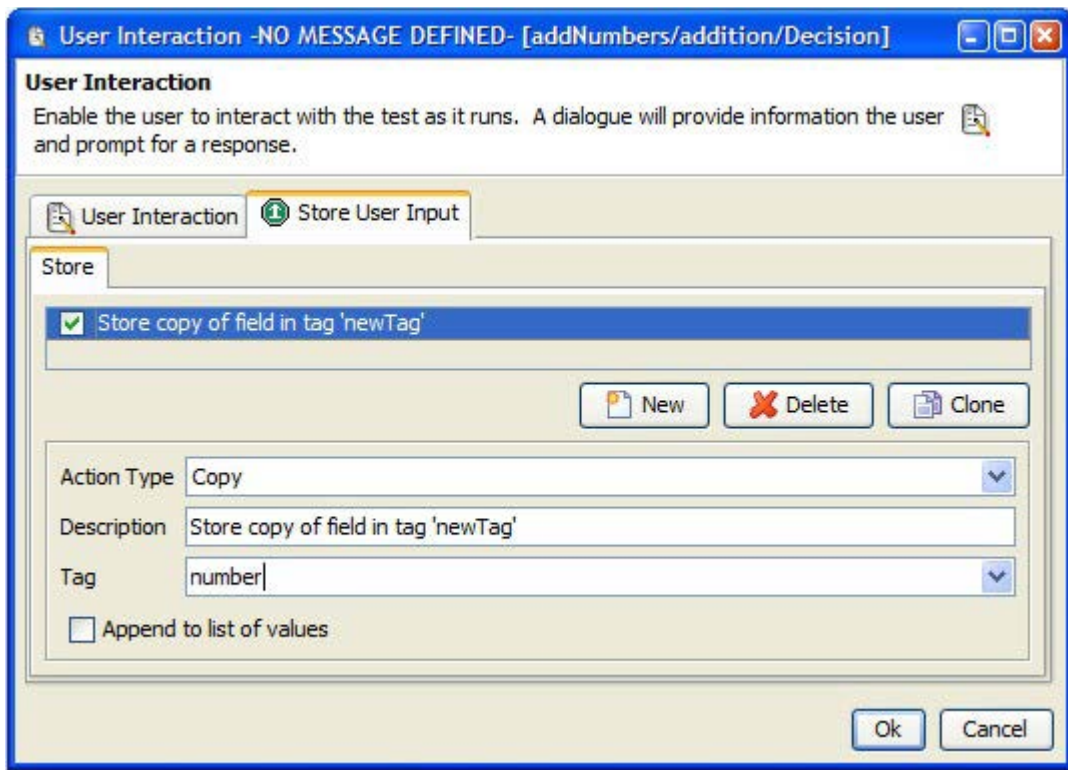
The following exercise creates a new test and uses a function and the *User Interaction* action to evaluate the results of user input.

1. Open the Test Factory perspective (**F10**).
2. Click on the **addition** operation and create a new test called “Decision”.
3. Double-click the new test to open it for editing, then right-click the Test Steps phase and add a User Interaction action (**New > General > User Interaction**).
4. Double-click the new action and add the following message in the **Message for user** field: “Please enter your number”
5. Tick the box next to the **Request input from user** option.

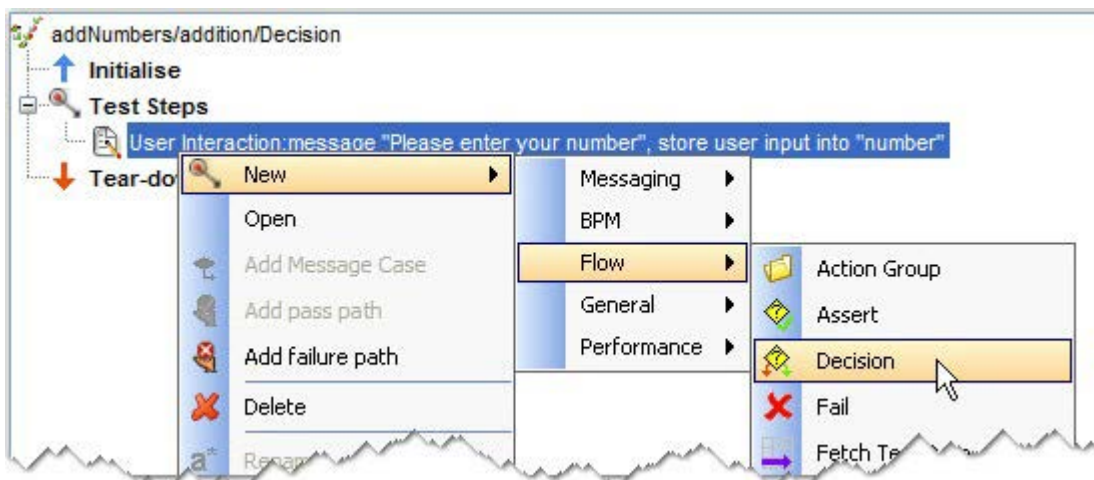


6. Select the **Store User Input** tab and click **New** to add a store action.

7. Rename the default tag (newTag) to “number”.

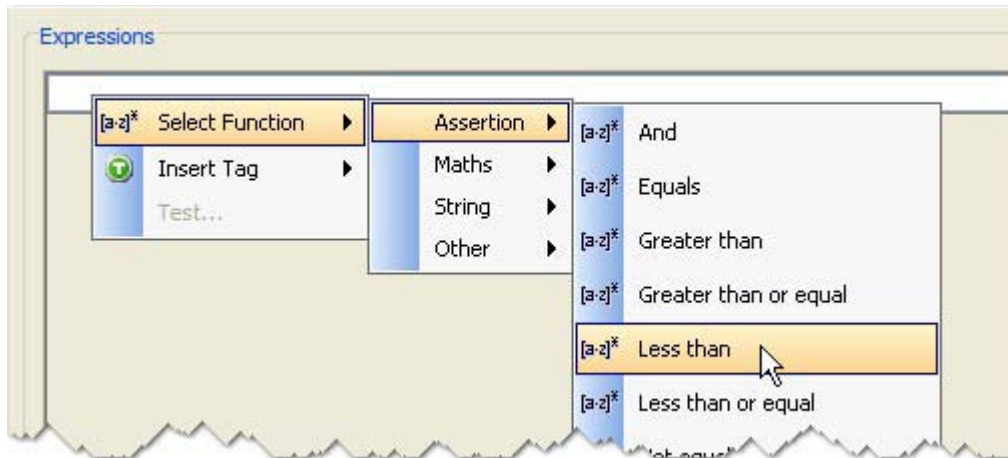


8. Click **OK** to save the changes and return to the test editor.
9. Right-click the *User Interaction* step and add a *Decision* step to it (**New > Flow > Decision**).

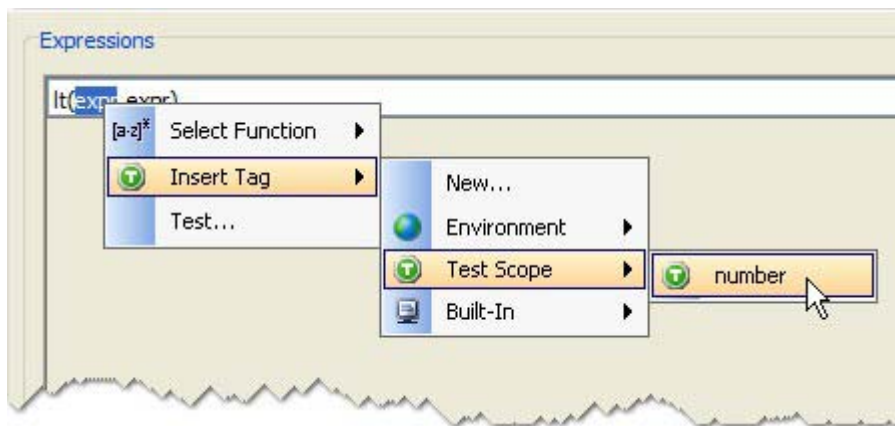


10. Double-click the *Decision* action to edit it.

- 
11. Click **Add** to add an expression.
  12. Right-click in the empty expression field and select the **Select Function > Assertion > Less than** option.



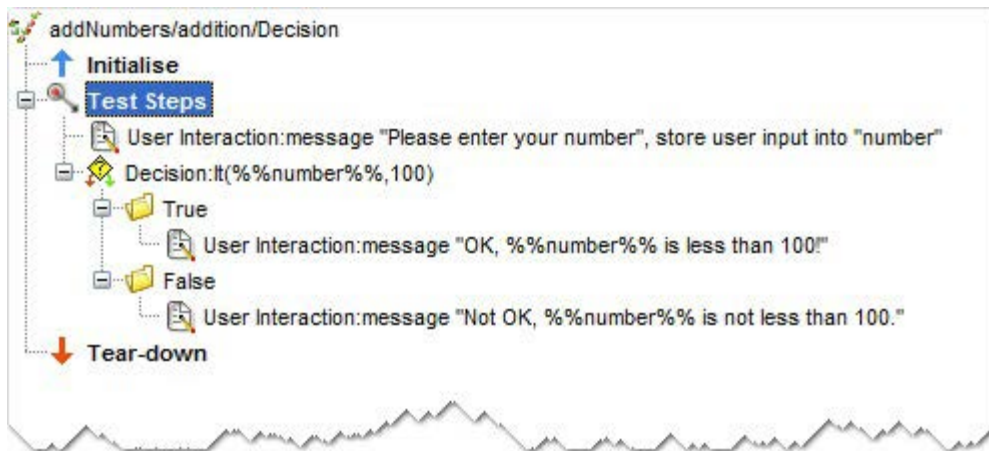
13. Select the first expression and right-click to insert the number tag (**Insert Tag < Test Scope > number**).




14. Replace the second expression with “100” and the expression should read as follows:  
`lt(%%number%%,100)`
15. Add *User Interaction* actions to the **True** and **False** paths to display the correct message.

---

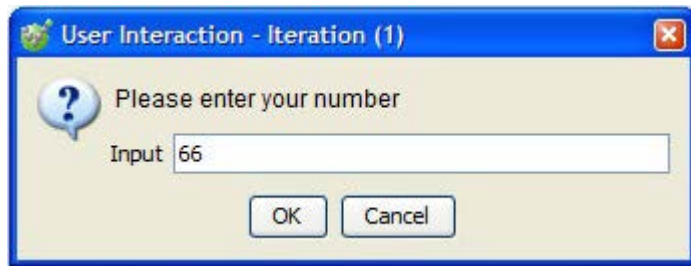
The completed test sequence should look as follows.



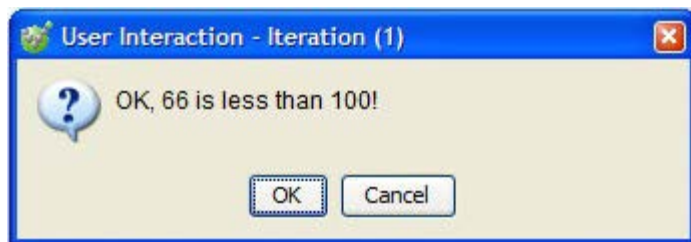
16. Save the project (click  or press **Ctrl + S**) and run the test.

You will be prompted for input.

17. Enter a number less than 100 (for example, 66) and click **OK**.



The number will be evaluated and the result is displayed.



---

# Glossary

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

Term	Description
Field	A bit of data constituent to a message. Most fields are scalar and therefore unitary, equivalent to data attributes. Vector fields are an aggregation of fields both scalar and vector, and are usually referred to as Messages. See also Message.
Message	A unit of information made up of a header consisting of meta-information and a body consisting of the message data.
Host	The computer on which a software process runs.
Publish-Subscribe	A messaging paradigm for efficient one-to-many communication in which one process (the publisher) sends information to zero or more other processes (subscribers).
Transport	Informally, the messaging software in use. For instance, TIBCO EMS, TIBCO ActiveEnterprise, IBM WebSphere® MQ (JMS).
Publishing	Making a message (data) available on a message channel.
Subscribing	Receiving a stream of messages (data) on a given message channel.
Server	A host computer on a network shared by more than one user.
JMS	Java Message Service, a J2EE technology. Several implementations of JMS exist, for instance IBM MQ and TIBCO EMS.
JMS Topic	The JMS equivalent of a subject, used by JMS providers to implement Publish-Subscribe messaging paradigms.
JMS Queue	A JMS Queue is normally used for one-to-one messaging.

---

---

Term	Description
TIBCO Rendezvous	A software toolkit for creating distributed applications that can inter-operate with TIBCO servers and applications on a TIBCO network. The product includes a communications daemon and APIs that define protocols for publish/subscribe and request/reply data distribution and exchange. It uses the subject-based addressing™ messaging technique for data delivery, and defines rules for supported subject naming formats.
TIBCO AE Message	A TIBCO (Active Enterprise) proprietary format for messages contained within the TIBCO Repository
Subject	A user-defined, meaningful name for identifying messages on transports. For example, the subject EQ.IBM might identify all pricing data about IBM stocks, while EQ.IBM.N might identify price data from the New York Stock Exchange only.

---

---

# Notices

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

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

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

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

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

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

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT,

---

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

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

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

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

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



---

capabilities of non-IBM products should be addressed to the suppliers of those products.

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

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

#### COPYRIGHT LICENSE:

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

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

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

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

---

## Trademarks and service marks

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

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

