

Rational Test Virtualization Server



Reference Guide

Version 8.0.1



Note

Before using this information and the product it supports, read the information in “Notices” on page 228.

This edition applies to version 8.0.1 of Rational Test Virtualization Server 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

About this Publication	vii
Intended Audience	viii
Scope	viii
Typographical Conventions	viii
Contacting IBM Support	viii
Introduction	1
Functional Overview	2
Definition	2
Components	3
Supported Stub-Types	6
Virtualization Capability	6
Usage Scenarios	7
Before You Begin	9
Installing Required Software	9
Creating a Rational Integration Tester Project Results Database	9
Getting Started	10
Using Rational Integration Tester	11
Starting the Application	11
Creating a New Project	11
Navigating the User Interface	11
Using Rational Test Control Panel	12
Logging In/Out	13
Navigating the User Interface	17
Creating & Modifying Message-Based Stubs	18
Introduction	19

Purpose.....	19
Benefits.....	19
Levels of Richness.....	19
Creating Message-Based Stubs.....	21
Empty Stub Method.....	21
Message Exchange Pattern (MEP) Method.....	24
Recording Studio Method.....	27
Modifying Message-Based Stubs.....	54
Events Tab.....	55
Behaviour Tab.....	77
Properties Tab.....	80
Logging Tab.....	84
Documentation Tab.....	85
Sift-and-Pass-Through Capability.....	86
Creating & Modifying Database Stubs.....	93
Introduction.....	94
Purpose.....	94
Benefits.....	94
Key Concepts.....	94
Before Creating Database Stubs.....	100
Preparation and Planning.....	100
Setting Up the Schema Used for Stubbing a Physical Database.....	100
Creating Database Stubs.....	103
Recording Studio Method.....	103
Proxy Mode Change Method.....	105
Modifying Database Stubs.....	109
Editing Options Toolbar.....	111
Wizards.....	113
Publishing & Running Stubs.....	123
Publishing Stubs.....	124
Verifying Publication of Stubs.....	129
Running Stubs.....	131

Starting Stubs.	131
Stopping Stubs.	141
Modifying Running Stubs	144
Controlling Running Stubs.	144
Locking and Unlocking Environments	146
Viewing Stub Error Messages	151
Viewing Stub Logs	151
Debugging Message-Based Stubs	153
Managing Agents.	155
Viewing All Running Agents Registered with Current RTCP Instance.	156
Viewing Agents For a Specific Domain/Environment	159
Troubleshooting	160
Handling Agent Failures	161
Resolving Stub Log Display Problems	163
Appendix A: Using the Data Model Editor	164
Creating Data Models	165
Editing Data Models.	171
Deleting Data Models	175
Appendix B: Creating Behaviours	176
Introduction	177
Creating a Behaviour.	178
Defining Interfaces.	198
Behaviour Interface	198
Callback Interface	198
Behaviour Factory	199
Behaviour Implementation	200
Appendix C: Using the RTCP Ant Client.	201
Introduction	202
Accessing Rational Test Control Panel Ant Tasks	203

Using Rational Test Control Panel Ant Tasks	204
Selecting Stubs	204
Selecting Scenarios	205
Locking and Unlocking Environments	205
Supported Ant Tasks	206
Start Stub Command	206
Stop Stub Command	207
Start Scenario Command	208
Stop Scenario Command	209
Lock Environment Command	210
Unlock Environment Command	211
Exit Codes	212
Appendix D: Using the RTCP Command-Line Client	214
Introduction	215
Using Commands	216
Supported Commands	217
Start Stub Command	217
Stop Stub Command	219
Start Scenario Command	220
Stop Scenario Command	222
Lock Environment Command	223
Unlock Environment Command	224
Command Help Facility	225
Exit Codes	226
Glossary	227
Notices	228
Trademarks and service marks	231

About this Publication

Contents

Intended Audience

Scope

Typographical Conventions

Contacting IBM Support

This guide describes how to use IBM® Rational® Test Virtualization Server to create and run message-based stubs, database stubs, and rich virtualized applications.

Intended Audience

This document assumes that readers are familiar with software testing, especially functional testing. Ideally, readers should also be familiar with the concepts underlying virtualized applications.

Scope

This document describes how to use Rational Test Control Panel and Rational Integration Tester to create and run message-based stubs, database stubs, and virtualized applications.

For information about installing Rational Test Virtualization Server, refer to the following documents:

- *IBM Rational Integration Tester Agent Installation Guide*
- *IBM Rational Integration Tester Installation Guide*
- *IBM Rational Test Control Panel Installation Guide*
- *IBM Rational Integration Tester Platform Pack Installation Guide* (optional)

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.
Bold	Menu items in graphical user interface windows (such as Microsoft Windows-based or UNIX X Window applications) from which you select options or execute macros and functions. Submenus and options of a menu item are indicated with a “greater than” sign, such as Menu > Submenu or Menu > Option .

Contacting IBM Support

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

Introduction

Contents

Functional Overview

Before You Begin

This chapter provides an overview of Rational Test Virtualization Server, and outlines what you need to do before you can start using Rational Test Control Panel and Rational Integration Tester to create and run stubs.

1.1 Functional Overview

The following sections outline the purpose and the components of Rational Test Virtualization Server, and when, why, and how the application should be used.

1.1.1 Definition

Rational Test Virtualization Server is IBM Rational software that is used for creating, maintaining, publishing, and running message-based stubs, database stubs, and rich virtualized applications.

Stubs and virtualized applications are used to simulate services within an environment for the purposes of software development and testing. Simulating parts of an environment can often be necessary if the real services are not yet available or because they are difficult or expensive to use.

Additionally, from a testing point of view, a tester will often require simple or deterministic responses from services used by the system under test (SUT) and the only way to ensure this is to develop stubs that act in a known way.

Rational Test Virtualization Server offers extremely powerful tools to quickly create and manage stubs in large environments. However, **stubs do not simulate underlying messaging transports**. For example, although Rational Test Virtualization Server does not simulate IBM WebSphere® MQ, it can simulate a service that is accessed over WebSphere MQ and any related stubs will still be accessed by means of WebSphere MQ.

Further, in most situations, **stubs do not remove the need for an Enterprise Service Bus (ESB) or messaging software**. A common exception to this is HTTP(S) or TCP-based Web Services where a Rational Test Virtualization Server stub will act as the HTTP(S)/TCP endpoint for the message.

Rational Test Virtualization Server helps developers by enabling them to create stubs representing services that are needed for the completion of development and testing but which that may not yet exist or which are difficult and time consuming to set up.

Rational Test Virtualization Server helps testers by enabling them to stub out the dependencies of an SUT and to control the external dependencies of that SUT, so they can plan, organize, manage, and control their software testing more effectively and more efficiently.

1.1.2 Components

Rational Test Virtualization Server comprises several IBM Rational software components. The following table outlines how each of those components is used within Rational Test Virtualization Server.

Software component	Mandatory, optional, or conditional	Purpose
Rational Integration Tester	Mandatory	<p>This is the primary application for creating stubs. It can be regarded as the design-time environment for stubs. It can also be used for limited deployment of virtual services.</p> <p>It enables you to:</p> <ul style="list-style-type: none">• Record events from the system.• Create stubs in a variety of ways, including from recorded events, and then executing them.• Create data models based on data in recorded messages for groups of stubs to use, facilitating the creation of richer virtual services. <p>For general information about using Rational Integration Tester, refer to <i>IBM Rational Integration Tester Reference Guide</i>.</p> <p>Information about using Rational Integration Tester as part of Rational Test Virtualization Server is contained elsewhere in this document.</p>
Rational Test Control Panel	Mandatory	<p>This application enables you to manage virtual services, agents, and proxies within an environment.</p> <p>Typically, after a stub has been created, it will be published from Rational Integration Tester to Rational Test Control Panel. The stub is then stored in a repository on Rational Test Control Panel. From this repository, each virtual service can be configured, deployed, and managed.</p> <p>For information about administering Rational Test Control Panel, refer to <i>IBM Rational Test Control Panel System Administration Guide</i>.</p> <p>Information about using Rational Test Control Panel as part of Rational Test Virtualization Server is contained elsewhere in this document.</p>

Software component	Mandatory, optional, or conditional	Purpose
Rational Integration Tester Agents	Mandatory	<p>Agents run stubs in the Rational Test Virtualization Server environment and are deployed on one or more computers within the environment</p> <p>Agents act as hosts for virtual services, enabling them to be deployed to different locations across a network.</p> <p>They are registered with, receive instructions from, and report log data to, Rational Test Control Panel.</p> <p>For general information about using Rational Integration Tester Agents, refer to <i>IBM Rational Integration Tester Reference Guide</i> and <i>IBM Rational Performance Test Server Reference Guide</i>.</p> <p>Information about using agents with Rational Test Virtualization Server is contained elsewhere in this document.</p>

Software component	Mandatory, optional, or conditional	Purpose
Rational Integration Tester Platform Pack (Proxies)	Conditional, depending on your testing requirements	<p>This component is required if you want to record:</p> <ul style="list-style-type: none"> • Live HTTP(S)- or TCP-based messages and route those messages automatically to either the live system or to a stub without having to constantly change the configuration of any client or server applications. • SQL events and/or stub databases that use JDBC connections. <p>Rational Integration Tester proxies are registered with, receive instructions from, and report log data to, Rational Test Control Panel.</p> <p>For general information about setting up a Rational Integration Tester HTTP proxy for use, refer to <i>IBM Rational Integration Tester Reference Guide for HTTP & Web Services</i>. For general information about setting up a Rational Integration Tester HTTP proxy for TCP traffic, refer to <i>IBM Rational Integration Tester Reference Guide for TCP/UDP Sockets</i>.</p> <p>For information about using the Rational Integration Tester proxies to record HTTP(S)/TCP traffic, refer to <i>IBM Rational Integration Tester Reference Guide</i>.</p> <p>Information about using the Rational Integration Tester proxies to route HTTP(S)- and TCP-based traffic to stubs automatically is contained elsewhere in this document.</p> <p>Rational Integration Tester JDBC proxies are registered with, receive instructions from, and report log data to, Rational Test Control Panel.</p> <p>For information about using the Rational Integration Tester JDBC proxy to record SQL events, refer to <i>IBM Rational Integration Tester Reference Guide</i>. Information about using the Rational Integration Tester JDBC proxy to stub databases is contained elsewhere in this document.</p>

NOTE: Depending on whether Rational Test Virtualization Server users are recording, or stubbing (virtualizing), or both, the communications among all these components will be slightly different.

1.1.3 Supported Stub-Types

Stubs and virtualized applications can be simple or rich. Rational Test Virtualization Server provides tools to create stubs and virtualized applications to the level required.

The following table summarizes the various types of stubs that can be created by Rational Test Virtualization Server.

Stub-Type	Description
Basic	There is a hard-coded single response for each specific input.
Non-deterministic	There are “n” hard-coded responses. A message switch is used to “switch” the response based on the input message.
Data-driven (parameterized)	There is input and/or output data specified in external data sources, for example, databases or spreadsheet files.
Data model-driven	There is input and/or output data in a data model that includes relationships among those data items.
Behaviour-driven	Stubs can be extended with “behaviours” that are Java plug-ins that can respond to messages and proactively cause the stub to behave in a particular way. IBM supplies some behaviours with Rational Integration Tester but customers can write their own. For example, a behaviour can be used to make a stub act as a market data feed source.
Deterministic	This comprises a series of Receive Request/Send Response or Subscribe/Publish actions that are based on the message used to create the stub.

1.1.4 Virtualization Capability

Rational Test Virtualization Server’s virtualization functionality is more granular than that of a Virtual Machine (VM) because Rational Test Virtualization Server can virtualize an application or database **or** just part of an application or database.

In contrast, Virtual Machines (VMs):

- Are designed to virtualize an entire machine.
- Require licenses for their applications and are usually not maintained by test teams.
- Are less flexible for testing purposes whereas a virtualized application in Rational Test Virtualization Server can be manipulated easily to fit your testing purposes. For example, Rational Test Virtualization Server enables you to create virtualized

applications that can send erroneous data for negative testing of a system under test.

1.1.5 Usage Scenarios

During software development projects, functional and non-functional requirements can change quickly, and test environments and applications are often in high demand from other teams.

Rational Test Virtualization Server can help because it enables you to:

- Continue testing when test environments are unavailable or not yet built.
- Test earlier and more often, reducing the cost of defects.
- Control the responses from services, enabling you to force the behaviour of the SUT.

In addition, Rational Test Virtualization Server is designed to be used in all software test phases from unit testing to user acceptance testing:

- During each test phase, you should aim to test in as complete a way as possible. For example, when unit testing individual operations or services, you may not always have the interfacing components available to test against. Rational Test Virtualization Server can be used to virtualize those interfacing components.
- As you proceed from unit testing to integration testing and onwards, you will probably want to introduce more “real” components into the system under test. Virtualization enables you to reduce any risks that may be associated with the introduction of those real components.

The following table outlines example scenarios where Rational Test Virtualization Server could be used.

Scenario	Description
Your testing project may be heavily reliant on integration with third parties or you want to control all systems with which the system under test (SUT) will communicate	Integration with third parties can be immensely frustrating and costly. Rational Test Virtualization Server can virtualize third party interfaces to enable you to test on your own terms according to your schedule.

Scenario	Description
You have integration testing dependencies	<p>Parallel development can sometimes mean that some projects may not be ready to begin integration testing when your project is ready.</p> <p>Rational Test Virtualization Server enables you to virtualize interfaces (even before they have been built) and continue testing.</p>
You want to run training or demonstration instances of applications without access to a large infrastructure	<p>For training or demonstration purposes, you may not require access to a production-size version of the system under test. In addition, you may not require access to any “downstream” applications.</p> <p>Rational Test Virtualization Server can virtualize and simplify interfaces, ensuring that training or demonstration exercises do not impact any production systems.</p>
You want to test a database-dependent application with scrubbed and isolated data	<p>Rational Test Virtualization Server can simulate databases as well applications. This means that you will have full control of all data to be used during testing.</p>
You want to provide a test system where none currently exists	<p>It may be too expensive to build a test environment and it may take too long to build it. Alternatively, there may be a test environment but it is being used by another team for the duration of your project.</p> <p>Rational Test Virtualization Server substitutes for the “absent” test environment by virtualizing applications.</p>

1.2 Before You Begin

Before you can use Rational Test Virtualization Server, you must:

1. Install the Rational Test Control Panel and Rational Integration Tester applications and at least one Rational Integration Tester Agent.
2. Create a Rational Integration Tester project results database (optional depending on requirements).

The following sections outline these tasks.

1.2.1 Installing Required Software

Although certain stubbing-related tasks can be accomplished within Rational Integration Tester, Rational Test Control Panel is required for managing virtual services, agents, and proxies within a large environment.

For information about installing Rational Test Control Panel, refer to *IBM Rational Test Control Panel Installation Guide*. For information about installing Rational Integration Tester, refer to *IBM Rational Integration Tester Installation Guide*.

NOTE: Rational Test Control Panel and Rational Integration Tester do **not** have to be installed on the same computer.

1.2.2 Creating a Rational Integration Tester Project Results Database

All Rational Integration Tester test data is stored in a project results database. If you want to view the output of any stubs that you will create and run, you must create a project results database after Rational Integration Tester is installed. However, if you want only to create and run stubs, creating a project results database is optional.

For information about creating and configuring a project results database, refer to *IBM Rational Integration Tester Installation Guide*.

Getting Started

Contents

[Using Rational Integration Tester](#)

[Using Rational Test Control Panel](#)

This chapter describes how to start using Rational Integration Tester and Rational Test Control Panel, which are the two primary components of Rational Test Virtualization Server.

2.1 Using Rational Integration Tester

NOTE: If you have previous experience of using Rational Performance Test Server and/or Rational Integration Tester, you may skip this section and proceed to [Using Rational Test Control Panel](#).

Rational Integration Tester is used for the following activities:

- Creating a model of the system under test.
- Recording events from the system.
- Creating, designing, and testing stubs before publishing them to Rational Test Control Panel.

The following sections outline how to start using the Rational Integration Tester application.

2.1.1 Starting the Application

For detailed instructions about how to start Rational Integration Tester, refer to *IBM Rational Integration Tester Getting Started Guide* or *IBM Rational Integration Tester Reference Guide*.

2.1.2 Creating a New Project

For detailed instructions about creating projects in Rational Integration Tester, refer to *IBM Rational Integration Tester Getting Started Guide* or *IBM Rational Integration Tester Reference Guide*. You must ensure that the correct Rational Test Control Panel URL is specified on the second screen of the Create New Project wizard.

2.1.3 Navigating the User Interface

Rational Integration Tester's user interface is "dockable", so sub-windows are attached or "docked" to one side of the application's workspace. The workspace comprises six perspectives, which can be selected from the **Perspectives** toolbar.

For more information about Rational Integration Tester's user interface, refer to *IBM Rational Integration Tester Getting Started Guide* or *IBM Rational Integration Tester Reference Guide*.

2.2 Using Rational Test Control Panel

NOTE: This section is a brief overview of Rational Test Control Panel and it is intended primarily for Rational Test Virtualization Server users who are not also Rational Test Control Panel administrators. The use of Rational Test Control Panel is also discussed in [Publishing & Running Stubs](#). For additional information about using Rational Test Control Panel, Rational Test Control Panel administrators should refer to *IBM Rational Test Control Panel System Administration Guide*.

Rational Test Control Panel is used for managing virtual services, agents, and proxies within an environment.

After a stub has been created, it can be published from Rational Integration Tester to Rational Test Control Panel. Therefore, you may not need to use Rational Test Control Panel until you have created at least one stub.

All Rational Test Control Panel users can accomplish the following tasks with Rational Test Control Panel:

- Change their login passwords (depending on how Rational Test Control Panel security is configured).
- View the Rational Test Virtualization Server “landscape” and start and stop stubs.
- View and modify scheduled tests.

Rational Test Control Panel administrators can accomplish additional tasks, such as managing users and domains, removing published stubs, and viewing Rational Test Control Panel logs.

The following sections outline how to start using the Rational Test Control Panel application.

2.2.1 Logging In/Out

The following sections describe how to log into, and log out from, Rational Test Control Panel; and how to change your Rational Test Control Panel login password.

2.2.1.1 Logging In

Depending on the security model that Rational Test Control Panel is using, you may have to **request** a Rational Test Control Panel **address** (host name or IP address, and port number), **user name**, and **password** from a Rational Test Control Panel administrator before you can log into the application.

In addition, there may be more than one Rational Test Control Panel instance in your test environment, so you may need to seek access to more than one Rational Test Control Panel instance.

To log into Rational Test Control Panel:

1. Open a web browser and enter a URL of the following format:

```
http://<Host Name or IP Address Provided by Rational Test  
Control Panel Administrator>:<Port Number Provided by Rational  
Test Control Panel Administrator>/GHServer/
```

Alternatively, if you are using Rational Integration Tester, clicking **Open** next to the **URL** field on the **Server Settings** tab of the Project Settings dialog box (which is opened by clicking **Project > Project Settings** on the menu bar) opens your computer's default web browser with Rational Test Control Panel's default URL displayed in the browser's address bar (you may have to edit the default URL before clicking **Open**).

If Rational Test Control Panel's built-in security functionality is being used, Rational Test Control Panel's Login screen is displayed.



Otherwise, Rational Test Control Panel's application window is displayed and there is no need to enter any login details.

2. In the **Username** and **Password** fields, enter your user name and password (provided by a Rational Test Control Panel administrator).
3. Click **Log in**.

NOTE: If Rational Test Control Panel's built-in security functionality is being used, you will be prompted to change your password after you have logged into Rational Test Control Panel for the first time. (For information about this, refer to [Changing Your Login Password](#).)

Rational Test Control Panel's application window is displayed.



NOTE: If you are a Rational Test Virtualization Server user, the **VIE** (Virtual Integration Environment) icon and **VIE** navigation link should be displayed on the application window. If they are not displayed, contact a Rational Test Control Panel administrator.

2.2.1.2 Logging Out

To log out from Rational Test Control Panel:

1. Click **LOGOUT** on the upper right corner of Rational Test Control Panel's application window.

A confirmation prompt is displayed.



2. Click **OK**.

You have now quitted the application.

NOTE: If Rational Test Control Panel's security functionality is disabled, the **LOGOUT** button is not displayed, so quitting your web browser will enable you to quit the Rational Test Control Panel application.

2.2.1.3 Changing Your Login Password

If Rational Test Control Panel's built-in security functionality is being used, you can change your Rational Test Control Panel login password at any time irrespective of your Rational Test Control Panel user privileges.

To change your login password:

1. In the **Password** and **Re-enter password** fields on the **Change Password** tab on the **Administration** page, enter and re-enter a new password (your user name should be displayed on the tab).

NOTE: A password must be unique but it can contain spaces and there is no limit on the number of characters that can be used.

2. Click **Change Password**.

NOTE: The **Change Password** button becomes unavailable if the passwords entered in the **Password** and **Re-enter password** fields are different.

Your password is changed. A status message is displayed to confirm this.

2.2.2 Navigating the User Interface

After logging into Rational Test Control Panel successfully, Rational Test Control Panel's **Home** page is displayed (for information about this, refer to [Using Rational Test Control Panel](#)).

The following table describes how to use the screen controls on the **Home** page.

Clicking the...	Displays...	For more information, refer to...
Logo or Home navigation link	The Home page.	(Not applicable)
Scheduling icon or navigation link	The Scheduling page.	<i>IBM Rational Integration Tester Reference Guide</i>
Agents icon or navigation link	The Agents page.	<i>IBM Rational Test Control Panel System Administration Guide</i>
VIE icon or navigation link	The VIE page.	Publishing & Running Stubs
Administration icon or navigation link	The Administration page.	Changing Your Login Password NOTE: The Administration page provides access to Rational Test Control Panel to various Rational Test Control Panel features, including activity and audit logs. For information about these logs, Rational Test Control Panel administrators should refer to <i>IBM Rational Test Control Panel System Administration Guide</i> .
LOGOUT button NOTE: This button is not displayed if Rational Test Control Panel's security functionality is disabled.	A logout confirmation prompt.	Logging Out
ABOUT button	Software version information and IBM contact information.	(Not applicable)

Creating & Modifying Message-Based Stubs

Contents

Introduction

Creating Message-Based Stubs

Modifying Message-Based Stubs

This chapter describes how to create and maintain message-based stubs.

3.1 Introduction

The following sections provide an introduction to message-based stubs.

3.1.1 Purpose

A message-based stub is essentially a Rational Integration Tester resource that listens for incoming messages on a particular transport. For example, a stub may be subscribed to an IBM WebSphere MQ message queue waiting for messages to arrive or it could be a Web Service waiting for a SOAP message to arrive over an HTTP connection.

NOTE: If you are using Rational Integration Tester 8.0.1 (or later), you can create REST-based stubs. Specifically, you can extract information from URLs and make the information available as tags in stubs, and Rational Integration Tester will generate a REST schema.

A stub can reply to specific messages or message-types, matched according to incoming filters, by using Message Case actions (for information about those actions, refer to *IBM Rational Integration Tester Reference Guide*).

3.1.2 Benefits

Message-based stubs enable you to simulate services that you are unable to use or that may otherwise be unavailable.

For example, if you are testing a TIBCO BusinessWorks process but you do not always have access to TIBCO Designer, you can record the process events, create a stub from them, and then run the stub in place of TIBCO Designer.

Stubs can use most Rational Integration Tester transports and schemas, including CHIPS, COBOL Copybook, Fedwire, FIX, SWIFT, and XML. (Some Rational Integration Tester transports cannot be used in stubs.)

3.1.3 Levels of Richness

A message-based stub can be simple or rich. For example, you might want to simulate a single service that utilizes a single message case and the default case. Alternatively, you might want to simulate a Web Service or some other point-to-point operation that includes multiple message-types or operations.

The basic principles for simple and rich message-based stubs are the same because a specific reply is sent when a specific message-type is received, but there are some important differences:

-
- A simple message-based stub will receive a message and (optionally) validate its contents. Based on the validation results or the fact that the message was received, the stub can return some static response, for example, a simple log action, a reply message, and so on.
 - In contrast, a rich message-based stub can receive incoming messages and, based on the specific contents of those messages, execute out a more extensive set of actions. For example, it might look up data in a database or spreadsheet, update one or more records in a database, utilize failure paths, and so on. Through the use of message cases and other actions, a stub can be configured to generate responses in an intelligent manner.

3.2 Creating Message-Based Stubs

In Rational Test Virtualization Server, message-based stubs can be created by using any of the following methods:

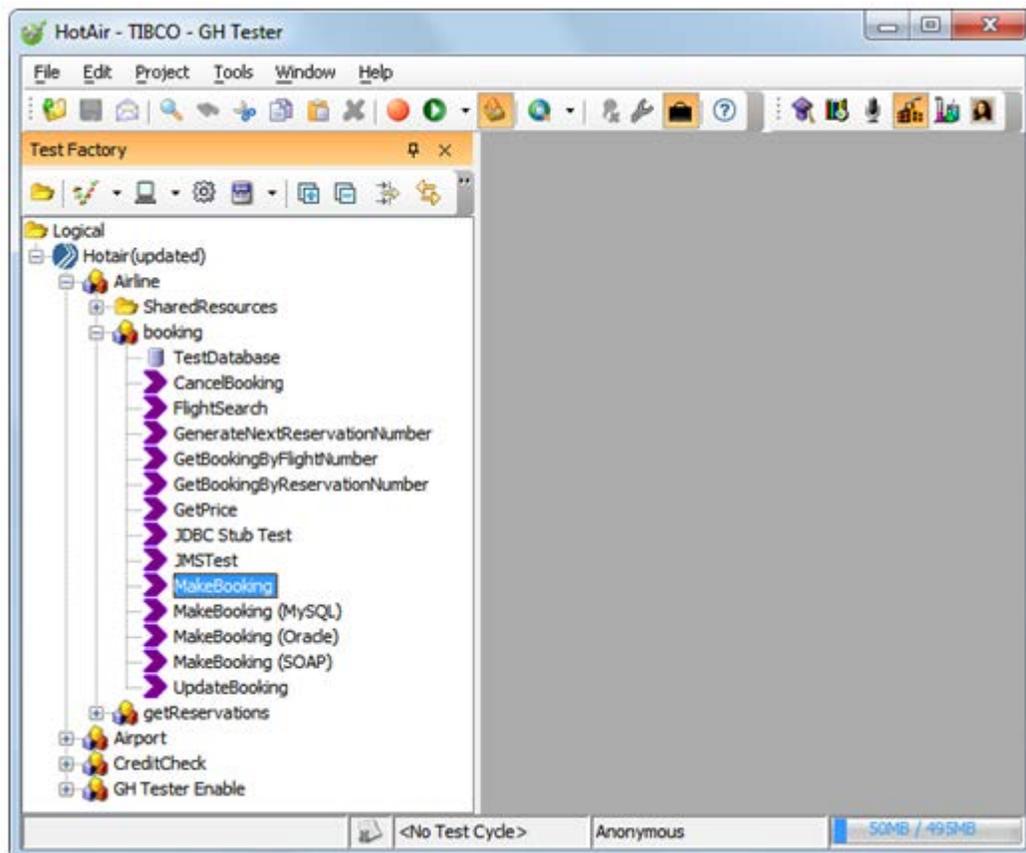
- Create an empty stub and configure it manually.
- Create a stub using Rational Integration Tester's Message Exchange Pattern (MEP) wizard.
- Create stubs from recorded events by using the Recorded Events wizard.

The following sections describe how to use each of these methods.

3.2.1 Empty Stub Method

To create a new empty stub:

1. Open Rational Integration Tester's Test Factory perspective.
2. Select the operation for which you want to create the stub.



-
3. Click the **Create New Stubs** button (📄) on the Test Factory window's toolbar.
Alternatively, right-click the operation and click **New > Stubs > Stub** on the shortcut menu.

NOTE: If at least one stub has already been created, you can right-click the **Stubs** folder or an existing stub in that folder and select **New > Stub**.

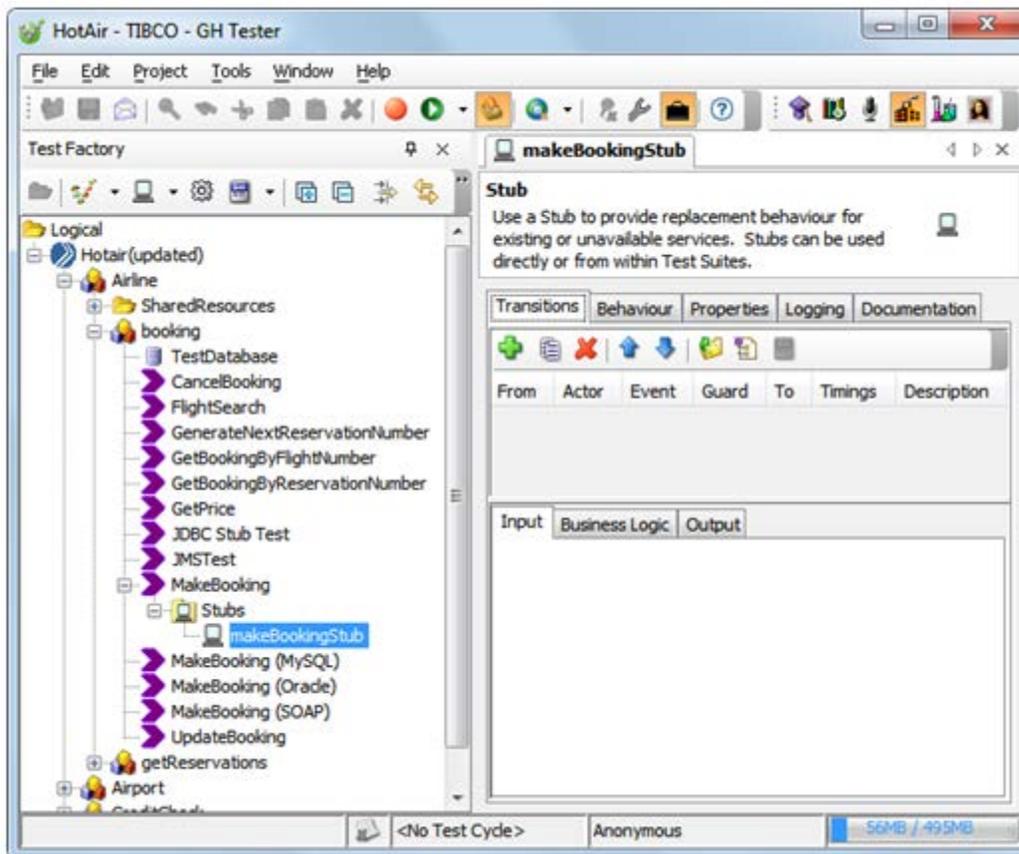
The Create new Stub dialog box is displayed.

4. In the **Name** field, enter a name for the stub.



5. Click **OK**.

The stub is opened for editing.



For information about using the Stub Editor to modify or enhance an empty stub, refer to [Modifying Message-Based Stubs](#).

6. After making any desired changes to your stub, you can save it by clicking the **Save** button (📁) on Rational Integration Tester's toolbar.

Alternatively, click **File > Save** on the menu bar or press CTRL+S.

The stub is now ready to run.

For information about running message-based stubs, refer to [Publishing & Running Stubs](#).

3.2.2 Message Exchange Pattern (MEP) Method

To create a stub by using a selected operation's MEP properties:

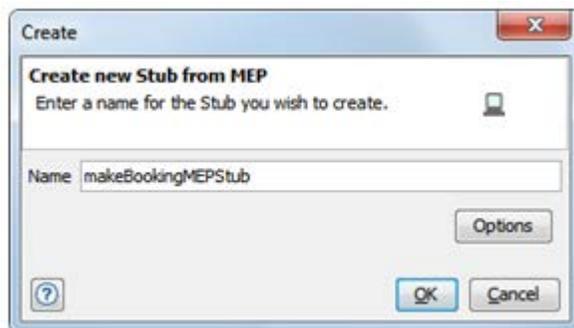
1. Open Rational Integration Tester's Test Factory perspective.
2. Click the **arrow** button (▼) next to the **Create New Stubs** button (📄) on the Test Factory window's toolbar.

Alternatively, right-click the operation where you want to create the stub and select **New > Stubs > Stub using MEP** on the shortcut menu.

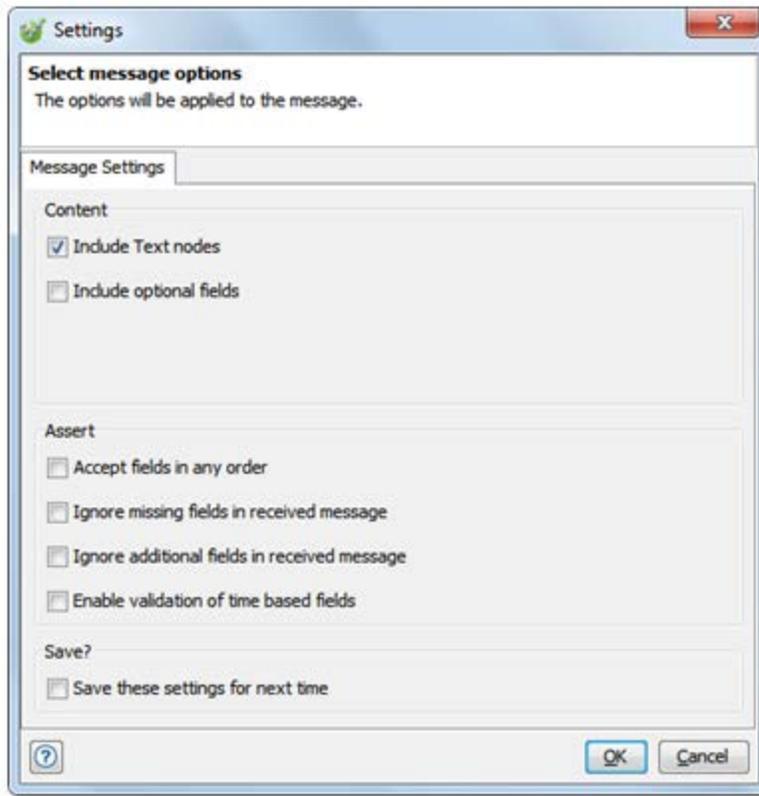
NOTE: If at least one stub has already been created, you can right-click the **Stubs** folder or one of the existing stubs and select **New > Stub using MEP**.

The Create new Stub from MEP dialog box is displayed.

3. In the **Name** field, enter a name for the stub.

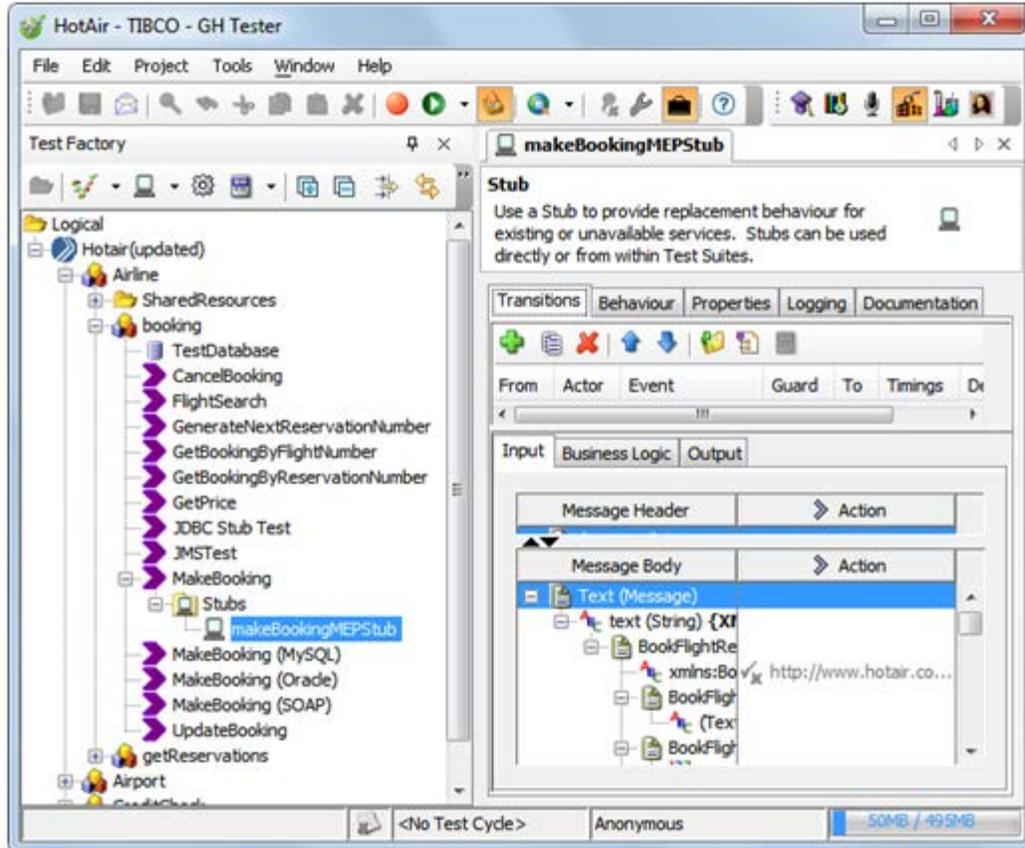


4. **Optional:** Click **Options** to open the Select message options dialog box, which enables you to enter additional settings.



5. Click **OK**.

The stub is opened for editing.



For information about using the Stub Editor to modify or enhance an “MEP stub”, refer to [Modifying Message-Based Stubs](#).

6. After making any desired changes to your stub, you can save it by clicking the **Save** button (📁) on Rational Integration Tester’s toolbar.

Alternatively, click **File > Save** on the menu bar or press CTRL+S.

The stub is now ready to run.

For information about running message-based stubs, refer to [Publishing & Running Stubs](#).

3.2.3 Recording Studio Method

This is the most effective method of creating a message-based stub if you have access to “live” systems from which you can record events, and it enables you to create very quickly stubs that behave like real services.

Rational Integration Tester’s Recording Studio perspective provides Event Monitors, which enable you to determine which parts of the environment you wish to record. For example, you might want to record events relating to specific parts of the system’s infrastructure, or you might want to record events relating to specific services that use the system’s infrastructure. After you have decided what you want to record, you can start recording events.

NOTE: For general information about using Rational Integration Tester’s Recording Studio, including the use of Rational Integration Tester proxies for recording events, refer to *IBM Rational Integration Tester Reference Guide*.

The environment must be modelled in Rational Integration Tester’s Architecture School perspective (Logical and Physical Views) before it can be recorded. However, for many technologies, you need to model only the transports (for example, a web server or an IBM WebSphere MQ Queue Manager), so there is no need to define any operations.

While events are being recorded, they are displayed on the Recording Studio perspective’s Events View window; and you can add or remove Event Monitors, and filter which events are displayed on the Events View window by selecting different monitors on the Event Monitors window.

NOTE: Recording events does not usually interfere with the operation of the system under test. For many (but not all) transports, events will still be dealt with in the same way that they would have been if Rational Integration Tester was not being used. The only difference is that the events will be accessible through Rational Integration Tester.

After you have completed recording events, you can use them to create any of the resource-types outlined in the following table by saving the events and using the Save wizard provided.

Resource-Type	Description
Unit tests	The data within recorded events may be hard-coded into a unit test.
Integration tests	The data within recorded events may be hard-coded into an integration test.
Stubs	The data within recorded events may be hard-coded into a basic stub that always returns the same response.
Triggers	<p>Events may be reused in the form of triggers, which enable you to stimulate the system under test directly from Rational Integration Tester.</p> <p>You can then record what happens in response but you must bear in mind that this will not necessarily be the same as what happened when you created the triggers, and you will not validating any events recorded in response to the triggers.</p> <p>Thus, you can send events to the system under test and bypass the GUI layer (and any other layers of the system that you might want to bypass) and determine how the system reacts to various inputs.</p>
Requirements	<p>You may want to save a message for subsequent use but not specify how it will be used.</p> <p>You could choose to save the message as an example message that can be viewed in Rational Integration Tester's Requirements Library and (optionally) imported into other Rational Integration Tester resources.</p>
Operations	If you have an incomplete model of the system under test, events recorded from the transports within the system can enable you to create new operations within your system model.
Test data sets	The data within recorded events may be entered into a data set, such as a comma-separated value (CSV) file, or a data model, which maps the relationships among data items in the system under test.

The following sections describe how to use the Recording Studio perspective to create the following message-based stub-types:

- Basic (hard-coded)
- Data-driven (parameterized)
- Data model-driven
- Deterministic

NOTE: For more information about using the Recording Studio perspective to create tests, triggers, requirements, and operations, refer to *IBM Rational Integration Tester Reference Guide*. For general information about stub-types, refer to [Supported Stub-Types](#).

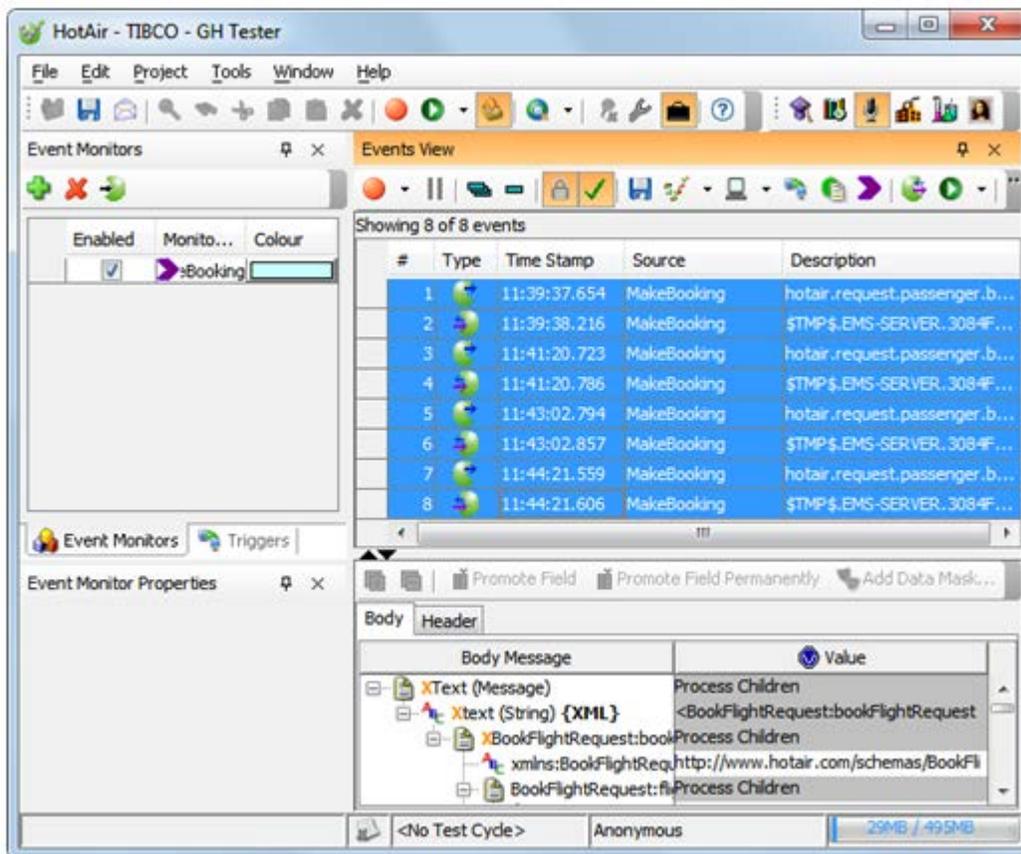
3.2.3.1 Creating Basic Stubs

A basic stub is a stub that always returns the same response. It does not execute any other operations, such as performing calculations, looking up any data, or making any decisions.

To create a basic (hard-coded) stub from recorded events:

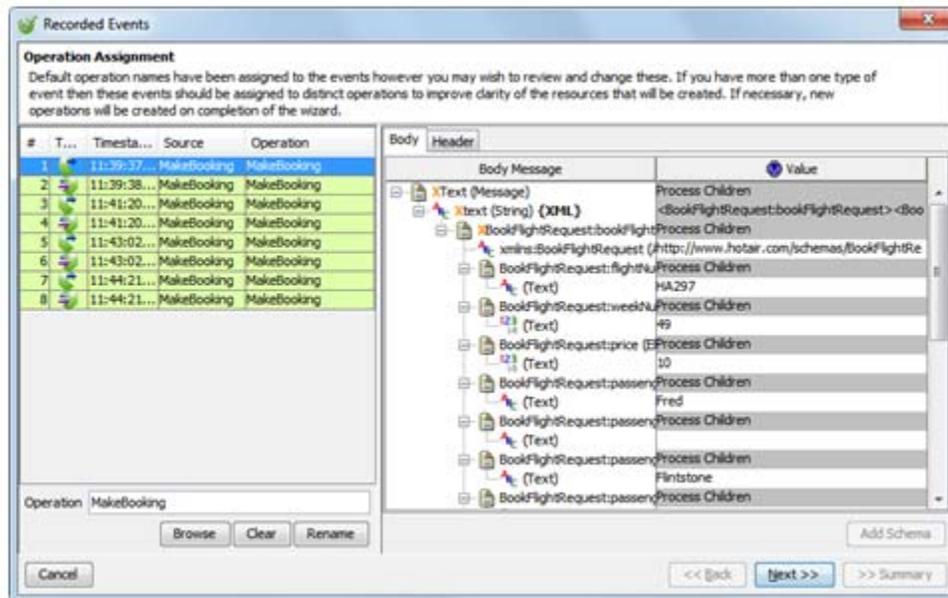
1. In Rational Integration Tester's Recording Studio perspective, select one or more recorded events on the Events View window.

NOTE: You should select events that represent the message exchanges that you wish to simulate. You can choose multiple message exchanges because Rational Integration Tester will distinguish among them and create appropriate operations. Thus, a stub created in Rational Integration Tester can simulate more than one operation.



NOTE: For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.

-
2. Click the **Save Stub from selected events** button () on the Events View toolbar. Alternatively, right-click the messages and click **Save Stub** on the shortcut menu. The Operation Assignment screen of the Recorded Events wizard is displayed.



The Operation Assignment screen, which is the third screen of the wizard, enables you to modify (if you wish) the operation and the events associated with the stub.

NOTE: The first and second screens of the Recorded Events wizard are the Resource Type and Data Storage screens. The screens are displayed only if you click the **Save** button () or press CTRL+S. Clicking the **Save Stub from selected events** button () bypasses the Resource Type and Data Storage screens.

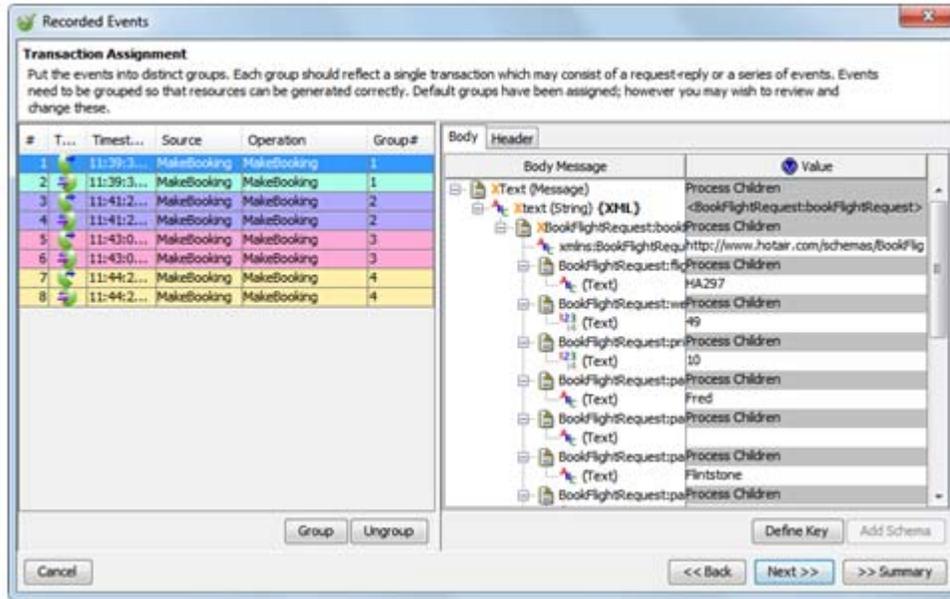
The following table describes how to use the Operation Assignment screen.

To...	Do this...
Define names of operations from recorded message payloads	<ol style="list-style-type: none">1. Select a message that contains a field with the value that must be used as the name of the operation.2. On the Body tab, select the relevant field.3. Click Define Operation. (This button is available only in Rational Integration Tester 8.0.1 (or later).) <p>The location of the field within the message is used to define operation names for all selected messages, or all messages if only a single message is selected.</p>
Select a different operation in the current Rational Integration Tester project	<ol style="list-style-type: none">1. Click Browse to open the Select a Resource dialog box.2. Select a different operation.3. Click OK to close Select a Resource dialog box.
Select a different operation that is not in the current Rational Integration Tester project	<ol style="list-style-type: none">1. Click Clear. Alternatively, type over text in the Operation field.2. In the Operation field, enter the name of a new operation. (The operation will be created after you have completed using the Recorded Events wizard.)3. Click Rename.4. If you have not selected all the events displayed on the Operation Assignment screen for assignment to the new operation, you are prompted to confirm whether you want any non-selected events on the screen to be assigned to the new operation.
Apply a different schema to a recorded event	<ol style="list-style-type: none">1. On the left side of the screen, select the event that you want to modify.2. On the right side of the screen, click the Body and/or Header tab (as appropriate).3. Select a field.4. Click Add Schema to display the Select Schema dialog box. (For information about selecting message schemas, refer to <i>IBM Rational Integration Tester Reference Guide</i>.)5. Clicking Finish on the Select Schema dialog box prompts you to confirm that you want to apply the selected schema to the selected event.
Close the wizard without making any changes	Click Cancel .

To...	Do this...
Move to the (next) Transaction Assignment screen of the wizard	Click Next .

NOTE: If you are creating a stub for multiple operations, Rational Integration Tester will attempt to verify that all selected recorded events are associated with the correct operations.

The Transaction Assignment screen, which is the fourth screen of the Recorded Events, enables you to group events into transactions. A transaction may consist of request-reply or a series of events.

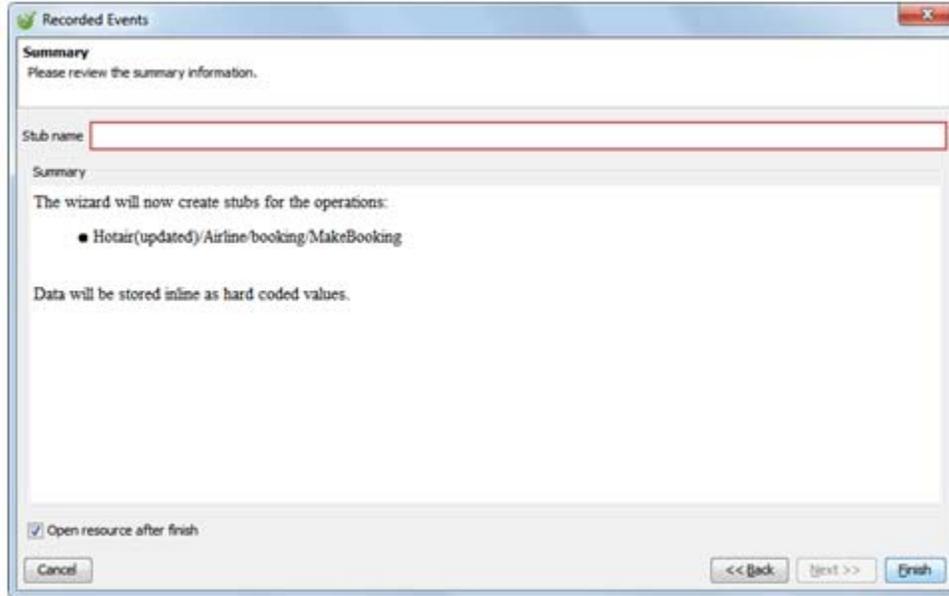


The following table describes how to use the Transaction Assignment screen.

To...	Do this...
Group two or more recorded events into a single transaction	<ol style="list-style-type: none"> 1. On the left side of the screen, select the events that you want to group into a transaction. 2. Click Group. The Group# field of the selected events changes.
“Ungroup” a transaction	<ol style="list-style-type: none"> 1. On the left side of the screen, select the events that you want to ungroup. 2. Click Ungroup. The Group# field of the selected events changes.

To...	Do this...
<p>Add one or more key fields to one or more selected events displayed on the screen</p> <p>Defining a key field enables you any events on the screen that contain that field but that grouping does not supersede or overwrite any groups of transactions created by selecting events and clicking Group</p>	<ol style="list-style-type: none"> 1. On the left side of the screen, select the event that you want to modify. 2. On the right side of the screen, click the Body and/or Header tab (as appropriate). 3. Select a field. 4. Click Define Key. If the field's value is also in any other fields of other events displayed on the screen, the Define Key dialog box is displayed. Otherwise, an error message is displayed. 5. On the Define Key dialog box, clear the check boxes of any other events that do not have the same logical value as the field you selected. 6. Optional: In the Save as a Field Type called field, enter a comment for key field. Any comment you enter will be displayed in the Architecture School perspective's Rule Cache. 7. Click OK to save your changes and to close the Define Key dialog box. <p>Any messages on the screen that are not already grouped into transactions and that contain the key field that you have defined are grouped.</p>
<p>Apply a different schema to a recorded event</p>	<ol style="list-style-type: none"> 1. On the left side of the screen, select the event that you want to modify. 2. On the right side of the screen, click the Body and/or Header tab (as appropriate). 3. Select a field. 4. Click Add Schema to display the Select Schema dialog box. (For information about selecting message schemas, refer to <i>IBM Rational Integration Tester Reference Guide</i>.) 5. Clicking Finish on the Select Schema dialog box prompts you to confirm that you want to apply the selected schema to the selected event.
<p>Close the wizard without making any changes</p>	<p>Click Cancel.</p>
<p>Move to the (previous) Operation Assignment screen</p>	<p>Click Back.</p>
<p>Move to the (final) Summary screen of the wizard</p>	<p>Click Next or Summary.</p>

The Summary screen, which is the final screen of the Recorded Events wizard, enables you to review the configuration of the stub and to save the stub.

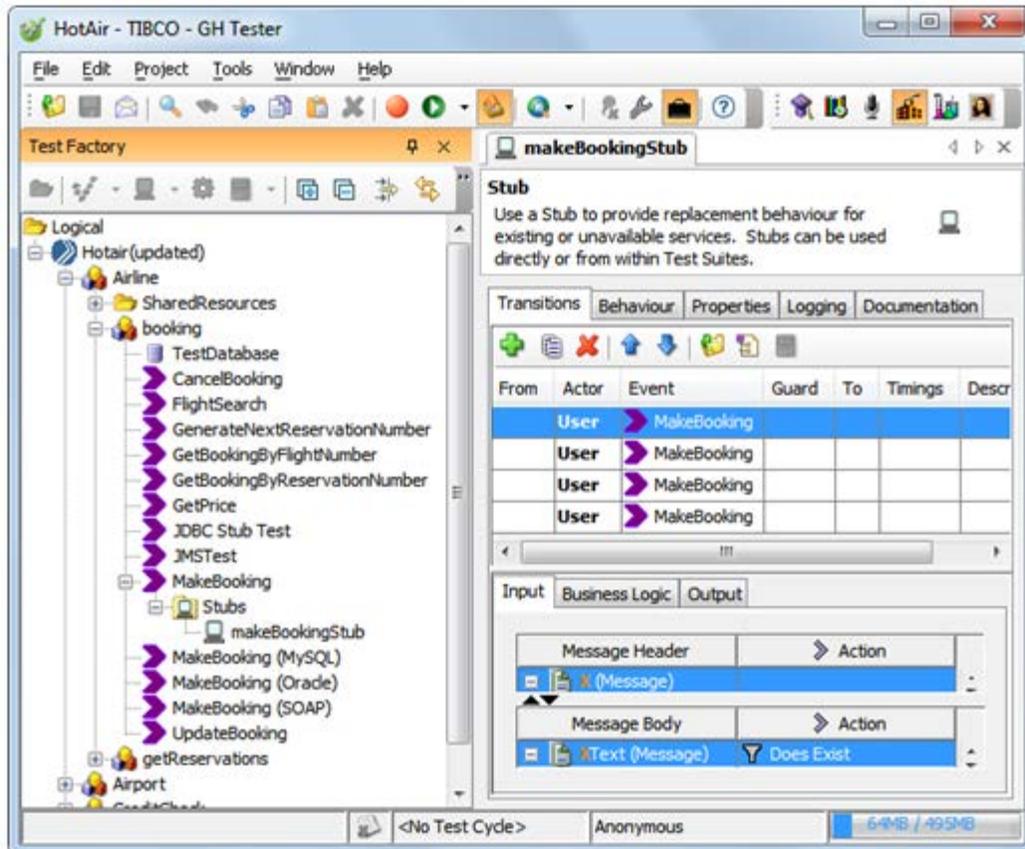


The following table describes how to use the Summary screen.

To...	Do this...
Open the stub in the Stub Editor after you save it and quit the Recorded Events wizard	Select the Open resource after finish check box.
Close the wizard without making any changes	Click Cancel .
Move to the (previous) Transaction Assignment screen	Click Back .
Save your stub	1. In the Stub name field, enter a name for the stub. 2. Click Finish .

After clicking **Finish** on the Summary screen:

- Rational Integration Tester's Test Factory perspective is displayed and the newly created stub is displayed under the relevant logical resource on the Test Factory perspective's component tree. If you created a stub for multiple operations, the stub is displayed under each applicable operation.
- If you selected the **Open resource after finish** check box on the Summary screen, the stub is also opened in the Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)).



3.2.3.2 Creating Data-Driven Stubs

A data-driven stub is a stub that is driven by the contents of a data source. The data fields that have been promoted to the events table (for information about this, refer to *IBM Rational Integration Tester Reference Guide*) are used to create the data source.

The following table outlines the different types of data sources that Rational Integration Tester supports. (For more information about these data source-types, refer to *IBM Rational Integration Tester Reference Guide*.)

Data Source-Type	Produced by Recorded Events Wizard?	Description
File data source	Yes	This reads data from a file, for example, a comma-separated value (CSV) file, a fixed width file, or some other delimited file-type. This data can be supplied to any tests or stubs in Rational Integration Tester.
Excel data source	No	This reads data from a worksheet in a Microsoft Excel workbook file.
Database data source	No	This reads data from a table in a database or the results of a query on a database. NOTE: The database must be set up in Rational Integration Tester's Architecture School perspective before this data source can be created.
Directory data source	No	This reads in a set of files, for example, a set of XML documents.

You can create a data-driven stub from recorded events or you can create a data source for a driven stub without using recorded events.

For example, you could create a stub by using the associated operation's Message Exchange Pattern properties (for information about this, refer to [Message Exchange Pattern \(MEP\) Method](#)) and then modifying the stub to read information from an Excel workbook file (with or without repeating elements), look up specified data, and then send any matching data in reply messages (for information about this, refer to [Modifying Message-Based Stubs](#)).

To create a data-driven stub from recorded events:

1. In Rational Integration Tester's Recording Studio perspective, select multiple recorded events on the Events View window.

NOTE: For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.

2. Click the **arrow** button (▼) next to the **Save Stub from selected events** button (📄) on the Events View toolbar and then click **Save Parameterized Stub** on the shortcut menu.

Alternatively, right-click the messages and click **Save Parameterized Stub** on the shortcut menu.

The Operation Assignment screen of the Recorded Events wizard is displayed.

The Operation Assignment screen, which is the third screen of the wizard, enables you to modify (if you wish) the operation and the events associated with the stub.

NOTE: The first and second screens of the Recorded Events wizard are the Resource Type and Data Storage screens. The screens are displayed only if you click the **Save** button (📄) or press CTRL+S. Clicking the **Save Stub from selected events** button (📄) bypasses the Resource Type and Data Storage screens.

For information about using Operation Assignment screen, refer to [Creating Basic Stubs](#).

NOTE: If you are creating a stub for multiple operations, Rational Integration Tester will attempt to verify that all selected recorded events are associated with the correct operations.

3. Clicking **Next** on the Operation Assignment screen displays the Transaction Assignment screen.

The Transaction Assignment screen, which is the fourth screen of the Recorded Events, enables you to group events into transactions.

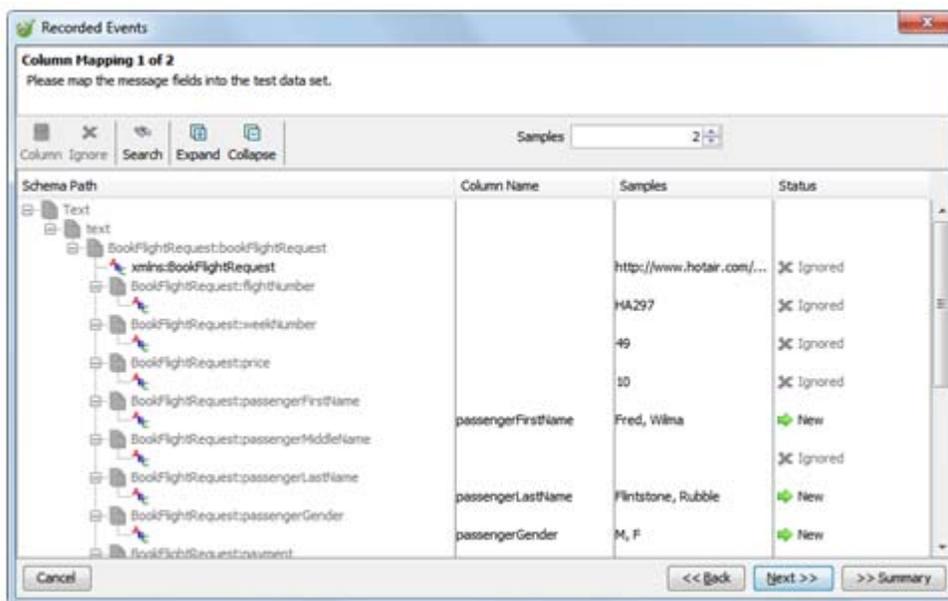
NOTE: The groups that you choose are important because they will determine the number of merged messages and thus the number of remaining screens in the Recorded Events wizard. For example, one recorded event may have a single transaction, which means that there is only one mapping. Alternatively, two recorded events may be

grouped into two groups, which means that there are two transactions and thus only one mapping. Equally, two recorded events may be put into a single group, which means that there are two mappings.

For information about using Transaction Assignment screen, refer to [Creating Basic Stubs](#).

4. Clicking **Next** on the Transaction Assignment screen displays the Column Mapping screen.

The Column Mapping screen enables you to map the fields in the selected recorded events to the data source that will be used for data-driven stubbing.

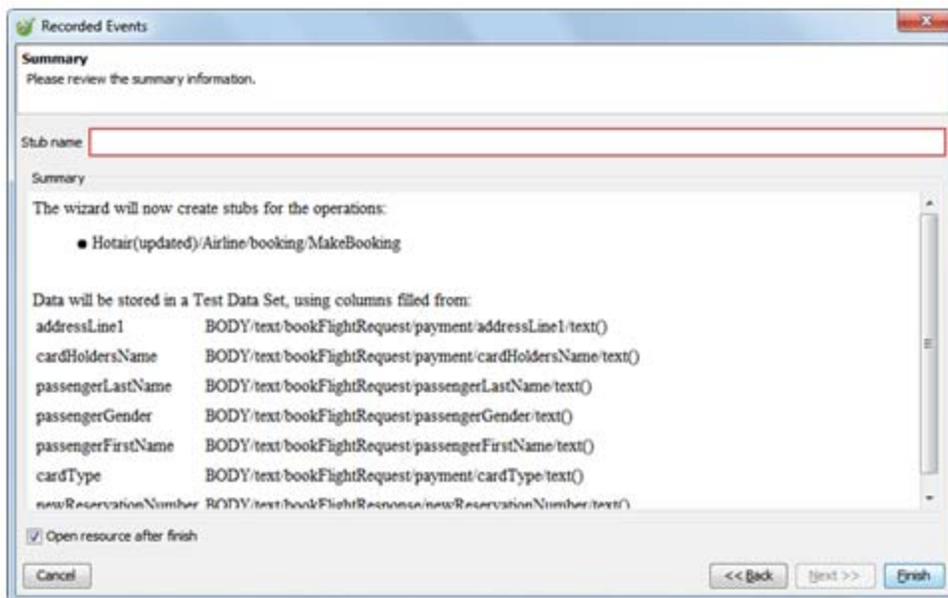


The following table describes how to use the Column Mapping screen.

To...	Do this...
Specify that a field in a recorded event will be a column in the data source	<ol style="list-style-type: none">1. Select the field.2. Click Column. The Map to column dialog box is displayed.3. Click OK. For the selected field, New is displayed under Status.

To...	Do this...
Map a field in a recorded event to an existing column	<ol style="list-style-type: none"> 1. Select the field. 2. Click Column. The Map to column dialog box is displayed. 3. In the list, click the column name to which you want to map the field. 4. Click OK.
Specify that a field in a recorded event is not mapped	<ol style="list-style-type: none"> 1. Select the field. 2. Click Ignore.
Search for a specific data item in a recorded event	<ol style="list-style-type: none"> 1. Click Search. 2. In the Find field (displayed on the lower half of the screen), enter your search query. 3. Press ENTER.
Increase or decrease the number of example values displayed for each field	In the Samples box, enter or select the number of examples you want to display (default value: 2).
Expand or collapse a node	<ol style="list-style-type: none"> 1. Click the node. 2. Click Expand or Collapse (as applicable).
Close the wizard without making any changes	Click Cancel .
Move to the (previous) Transaction Assignment screen	Click Back .
Move to the next screen	<p>Click Next.</p> <p>NOTE: The next screen may be another Data Mapping screen because the number of Column Mapping screens displayed by the Recorded Events wizard depends on the number of merged messages (which determined by the using the Transaction Assignment screen).</p>
Move to the (final) Summary screen of the wizard	Click Summary .

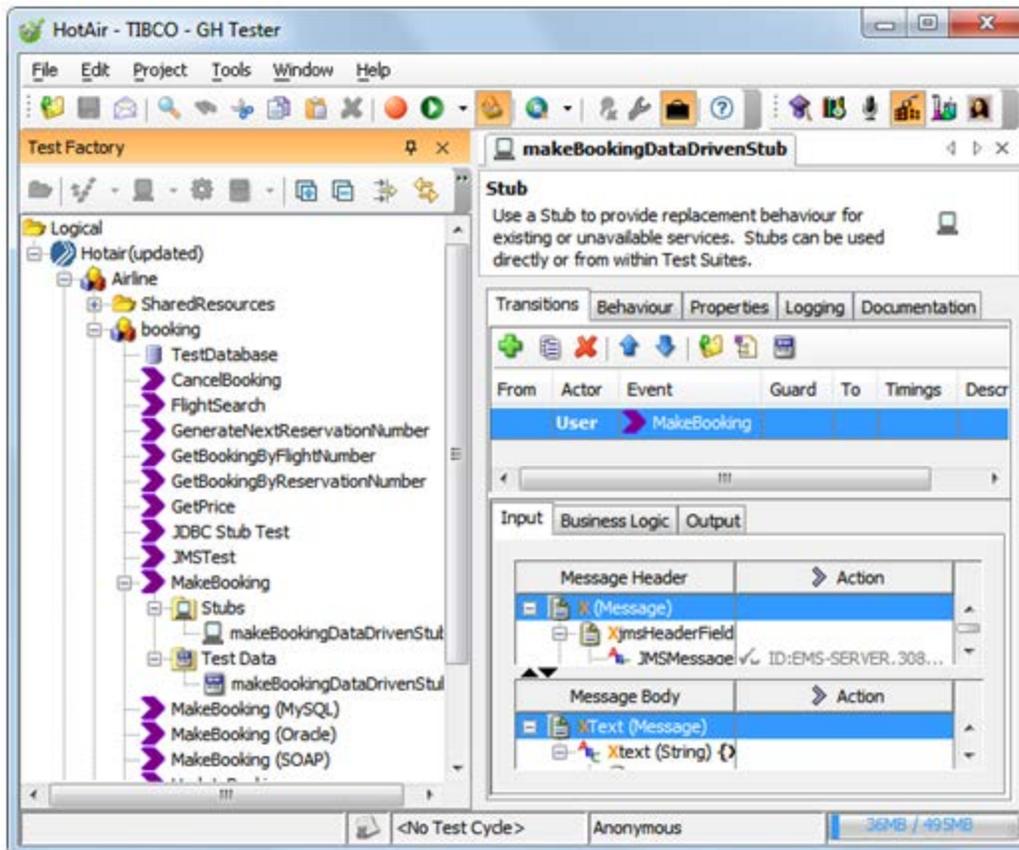
The Summary screen, which is the final screen of the Recorded Events wizard, enables you to review the configuration of the stub and to save the stub.



For information about using the Summary screen, refer to [Creating Basic Stubs](#).

After clicking **Finish** on the Summary screen:

- Rational Integration Tester's Test Factory perspective is displayed, and the newly created stub and data source are displayed under the relevant logical resource on the Test Factory perspective's component tree. If you created a stub for multiple operations, the stub is displayed under each applicable operation.
- If you selected the **Open resource after finish** check box on the Summary screen, the stub is also opened in the Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)). A Lookup Test Data action that uses the new data source is displayed on the **Business Logic** tab on the **Events** tab of the Stub Editor.



NOTE: The **Output** tab may not detect a Send Reply action on the **Business Logic** tab if the Send Reply action is nested under a Lookup Test Data action.

3.2.3.3 Creating Data Model-Driven Stubs

Basic (hard-coded) and parameterized (data-driven) stubs are examples of simple stub-types because the data that is provided when the stub is run is either hard-coded in the stub itself or is taken from a simple data source (whichever is applicable). In each case, the same data is used each time when the stub is started.

However, the testing of complex systems may require persistent storage of data so that a stub can save information across multiple operations. This enables you to virtualize services that create data, for example, a customer, and then use that data in a “GetCustomer” service without having to know a design time the data will be used. It is this capability that distinguishes simple stubs from those that are coordinated to provide a virtualized application.

In cases where you require multiple operations within a stub, or multiple stubs to share data to provide a virtualized application, you should use a data model to hold such data. Data models can be shared across stubs. In addition, data models can hold data about multiple entities and their relationships, such as customers and orders.

Therefore, a data model is a simple view of entities and their relationships that can be used by stubs to persist information. Data models exist as assets within a project. A stub’s properties define which data model it is using (if any). Stubs can read and modify data held in a data model. The Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)) enables you to indicate whether an operation is trying to create, update, or delete data based on the received message, reducing the time and effort required to create rich stubs.

In Rational Integration Tester, you can create a data model by using the Data Model Editor (for information about this, refer to [Appendix A: Using the Data Model Editor](#)). Alternatively, you can create a data model while creating a data model-driven stub from recorded events. This second method analyzes the message in recorded events and creates a data model and entities within that data model based on the messages. Operations that are created will be connected to the data model automatically.

To create a data model-driven stub from recorded events:

1. In Rational Integration Tester’s Recording Studio perspective, select multiple recorded events on the Events View window.

NOTE: For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.

2. Click the **arrow** button (↘) next to the **Save Stub from selected events** button (📄) on the Events View toolbar and then click **Save Data Model Stub** on the shortcut menu.

Alternatively, right-click the messages and click **Save Data Model Stub** on the shortcut menu.

The Operation Assignment screen of the Recorded Events wizard is displayed.

The Operation Assignment screen, which is the third screen of the wizard, enables you to modify (if you wish) the operation and the events associated with the stub.

NOTE: The first and second screens of the Recorded Events wizard are the Resource Type and Data Storage screens. The screens are displayed only if you click the **Save** button () or press CTRL+S. Clicking the **Save Stub from selected events** button () bypasses the Resource Type and Data Storage screens.

For information about using Operation Assignment screen, refer to [Creating Basic Stubs](#).

NOTE: If you are creating a stub for multiple operations, Rational Integration Tester will attempt to verify that all selected recorded events are associated with the correct operations.

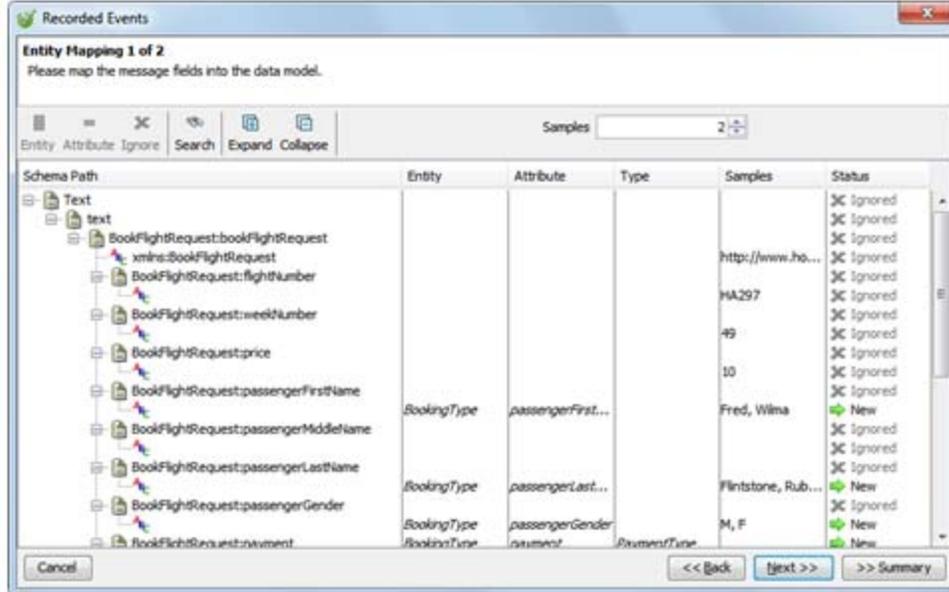
3. Clicking **Next** on the Operation Assignment screen displays the Transaction Assignment screen.

The Transaction Assignment screen, which is the fourth screen of the Recorded Events, enables you to group events into transactions.

NOTE: The groups that you choose are important because they will determine the number of merged messages and thus the number of remaining screens in the Recorded Events wizard. For example, one recorded event may have a single transaction, which means that there is only one mapping. Alternatively, two recorded events may be grouped into two groups, which means that there are two transactions and thus only one mapping. Equally, two recorded events may be put into a single group, which means that there are two mappings.

For information about using Transaction Assignment screen, refer to [Creating Basic Stubs](#).

4. Clicking **Next** on the Transaction Assignment screen displays the Entity Mapping screen.



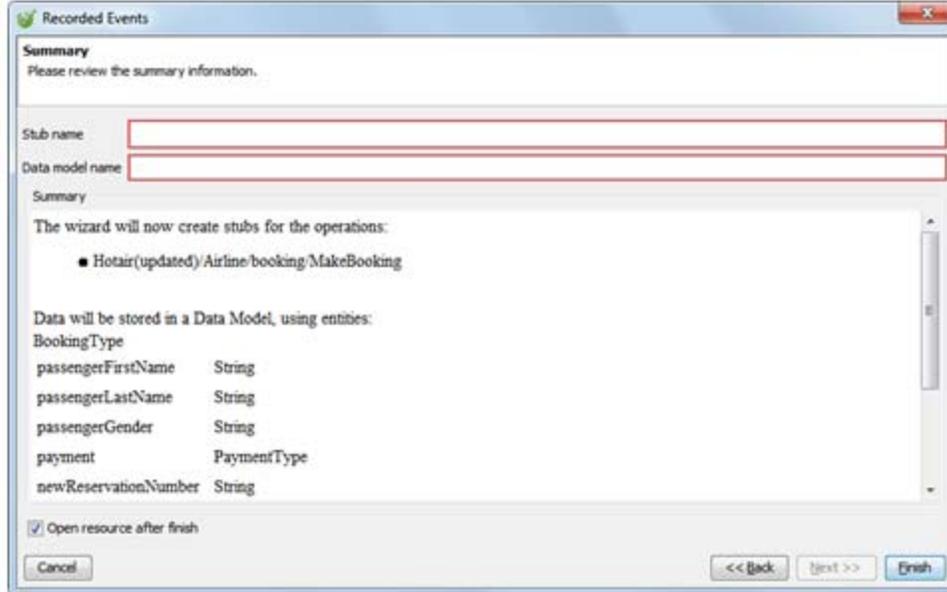
The Entity Mapping screen enables you to map the fields in the selected recorded events to the data source that will be used for data model-driven stubbing.

The following table describes how to use the Entity Mapping screen.

To...	Do this...
Specify that a field in a recorded event will be an entity in the data model	<ol style="list-style-type: none"> 1. Select the field. 2. Click Entity. The Map to entity dialog box is displayed. 3. Click OK. For the selected field, New is displayed under Status.
Specify that a field in a recorded event will be an attribute of a particular entity in the data model	<ol style="list-style-type: none"> 1. Select the field. 2. Click Attribute. The Map to Attribute dialog box is displayed. 3. Click OK. For the selected field, New is displayed under Status.
Specify that a field in a recorded event is not included in the data model	<ol style="list-style-type: none"> 1. Select the field. 2. Click Ignore.
Search for a specific data item in a recorded event	<ol style="list-style-type: none"> 1. Click Search. 2. In the Find field (displayed on the lower half of the screen), enter your search query. 3. Press ENTER.

To...	Do this...
Increase or decrease the number of example values displayed for each field	In the Samples box, enter or select the number of examples you want to display (default value: 2).
Specify that a field is a “key” field so that it can be used to match data between different messages	Under the Key column, select the check box of the field.
Expand or collapse a node	<ol style="list-style-type: none"> 1. Click the node. 2. Click Expand or Collapse (as applicable).
Close the wizard without making any changes	Click Cancel .
Move to the (previous) Transaction Assignment screen	Click Back .
Move to the next screen	Click Next . NOTE: The next screen may be another Entity Mapping screen because the number of Entity Mapping screens displayed by the Recorded Events wizard depends on the number of merged messages (which determined by the using the Transaction Assignment screen).
Move to the (final) Summary screen of the wizard	Click Summary .

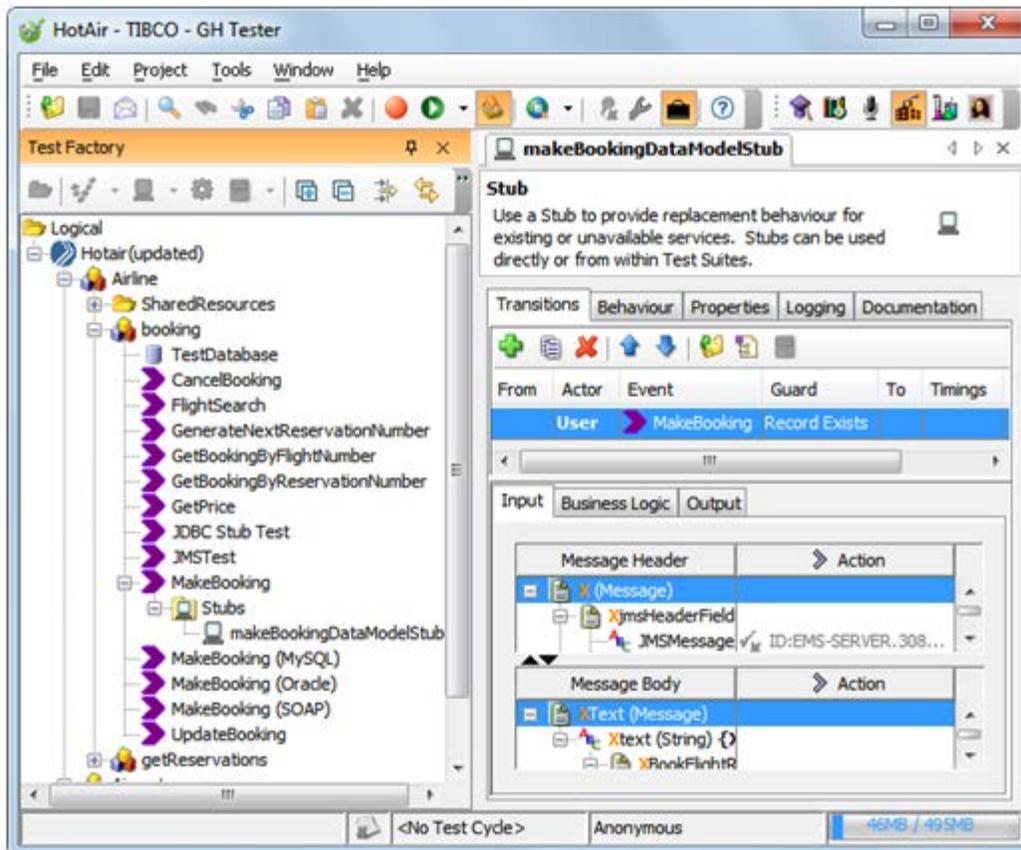
The Summary screen, which is the final screen of the Recorded Events wizard, enables you to review the configuration of the stub and to save the stub and the data model.



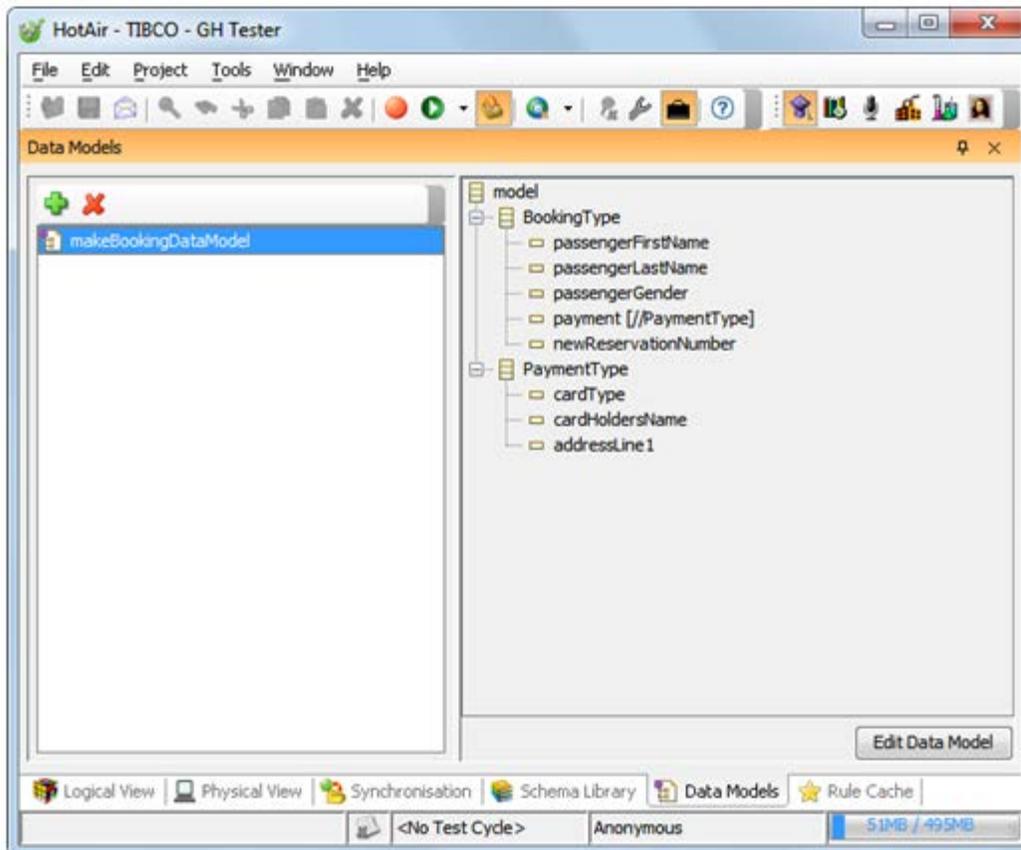
For information about using the Summary screen, refer to [Creating Basic Stubs](#).

After clicking **Finish** on the Summary screen:

- Rational Integration Tester's Test Factory perspective is displayed, and the newly created stub is displayed under the relevant logical resource on the Test Factory perspective's component tree. If you created a stub for multiple operations, the stub is displayed under each applicable operation.
- If you selected the **Open resource after finish** check box on the Summary screen, the stub is also opened in the Stub Editor (for information about this, refer to [Modifying Message-Based Stubs](#)).



- If you view the Architecture School perspective's Data Models window, the newly created data model is listed on the upper left of the window and the data model's details can be displayed on Data Models window by selecting it. For information about editing a data model, refer to [Editing Data Models](#).



3.2.3.4 Creating Deterministic Stubs

A deterministic stub comprises a series of Receive Request/Send Response or Subscribe/Publish actions that are based on the message used to create the stub.

To create a deterministic stub from recorded events:

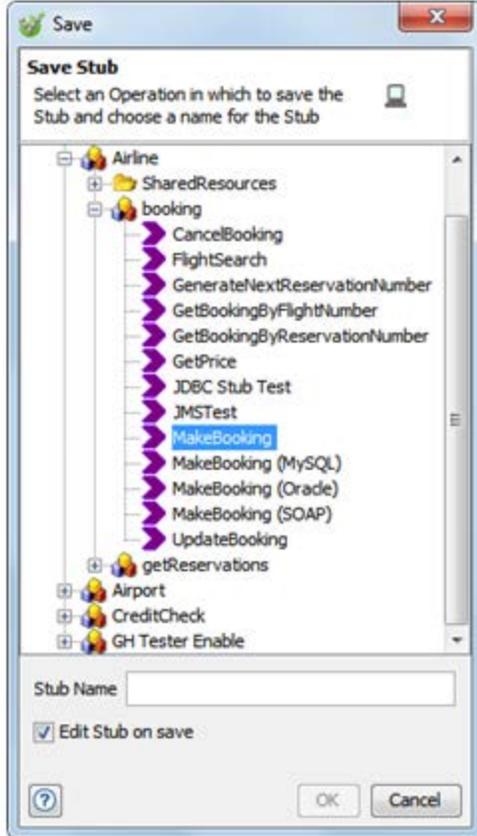
1. In Rational Integration Tester's Recording Studio perspective, select multiple recorded events on the Events View window.

NOTE: For information about using the Recording Studio perspective to record events, refer to *IBM Rational Integration Tester Reference Guide*.

2. Click the **arrow** button (↘) next to the **Save Stub from selected events** button (📁) on the Events View toolbar and then click **Save Deterministic Stub** on the shortcut menu.

Alternatively, right-click the messages and click **Save Deterministic Stub** on the shortcut menu.

The Save Stub dialog box is displayed.

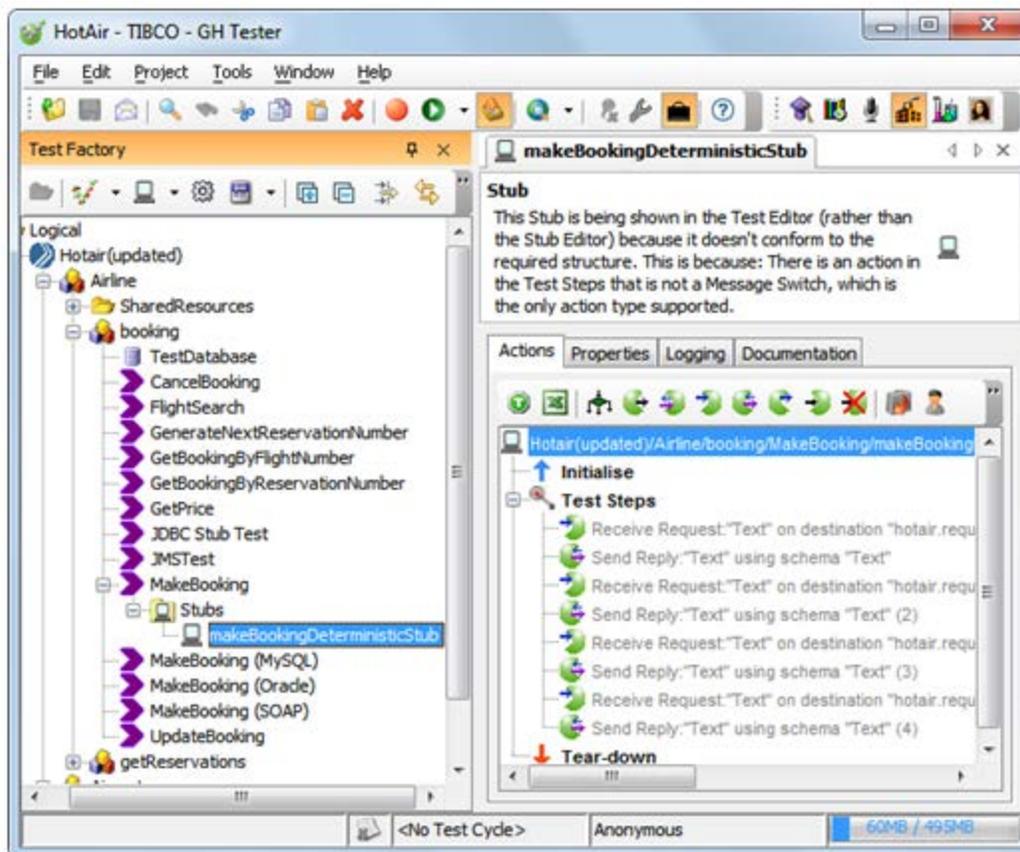


3. Select the operation with which you want to associate the stub.

NOTE: If there is only one operation associated with the selected recorded events, it is selected automatically when the Save Stub dialog box is displayed.

4. In the **Stub Name** field, enter the name of the stub.
5. **Optional:** Clear the **Edit Stub on save** check box if you do not want to edit the stub after creating it.
6. Click **OK**.

If the **Edit Stub on save** check box on the Save Stub dialog box was selected, Rational Integration Tester's Test Factory perspective is displayed, and the newly created stub is displayed under the relevant logical resource on the Test Factory perspective's component tree and the stub is displayed on a basic Stub Editor screen that does not support events and behaviours (for information about modifying stub, refer to [Modifying Message-Based Stubs](#)).



If the **Edit Stub on save** check box is cleared and you want to verify that the stub has been created, you must open Rational Integration Tester's Test Factory's perspective and view the component tree.

3.3 Modifying Message-Based Stubs

In Rational Integration Tester 5.4.0 (or later), you can use either the Test Editor or the Stub Editor to create, modify, and enhance any legacy or new stubs created by Rational Integration Tester.

For example, you may want to create an advanced message-based stub that can record and report its internal state so that you can create a virtualized application that replicates a complex live transaction processing system.

The following table outlines how to open the Test Editor and the Stub Editor.

If you want to...	Do this...	Then...
Modify a stub created by Rational Integration Tester 5.2.11 (or earlier).	In Rational Integration Tester's Test Factory perspective: <ul style="list-style-type: none">• Double-click the stub.• Alternatively, right-click the stub and click Open on the shortcut menu.	The selected stub is displayed in the Stub Editor if the stub does not contain any features that cannot be handled by the Stub Editor. Otherwise, the selected stub is displayed in the Test Editor.
Modify a stub created by Rational Integration Tester 5.4.0 (or later).	In Rational Integration Tester's Test Factory perspective: <ul style="list-style-type: none">• Double-click the stub.• Alternatively, right-click the stub and click Open on the shortcut menu.	The selected stub is displayed in the Stub Editor. NOTE: You can also modify a stub while it is running. For information about this, refer to Modifying Running Stubs .
Use Test Editor to modify a stub created by any release of Rational Integration Tester.	In Rational Integration Tester's Test Factory perspective, right-click the stub and click Open With > Test Editor on the shortcut menu.	The selected stub is displayed in the Test Editor.

The Stub Editor comprises the following tabs:

- **Events**
 - **Behaviour**
 - **Properties**
 - **Logging**
 - **Documentation**
-

The following sections describe how to use these tabs.

NOTE: For information about using the Test Editor, refer to *IBM Rational Integration Tester Reference Guide*.

NOTE: For information about debugging message-based stubs, refer to [Debugging Message-Based Stubs](#).

3.3.1 Events Tab

This tab enables you to define how the stub will handle internally and/or externally received events. Strictly defined, an *event* refers to a transition between two states. However, quite often, stubs will not use states, so an event can be regarded as the action that a stub will take when a message arrives on the operation for which you want to create a stub.

NOTE: This tab is not displayed when editing a deterministic stub.

The following sections describe how to use the buttons, lists table, and three tabs on the **Events** tab.

3.3.1.1 Buttons and List Table

The following table describes each button on the upper half of the **Events** tab.

Button	Description
Add Event	Clicking this button creates a new event for the currently selected operation.
Clone Event	Clicking this button creates a copy of the currently selected event in the list table.
Delete Event	Clicking this button deletes the currently selected event in the list table.
Move Event Up/Down	Clicking these buttons changes the list order of the currently selected event in the list table and consequently the order in which any guard conditions of any events listed in the table are evaluated while the stub is being executed.
Open operation referenced by event	Clicking this button opens the associated operation's Operation dialog box and changes the perspective from Test Factory to Architecture School.

Button	Description
Open data model referenced by stub	Clicking this button opens the data model (a comma-separated value (CSV) file), if any, associated with the stub. NOTE: You must use the Properties tab to associate a data model with the stub. (For information about using the Properties tab, refer to Properties Tab .)
Open data used by selection	Clicking this button opens any data associated with the currently selected event in the list table.
Advanced	Clicking this button toggles the display of fields on the list table between basic view and advanced view. NOTE: This button is available only in Rational Integration Tester 8.0.1 (or later). NOTE: A stub will be displayed in advanced view by default if data has been added to any of the columns that are displayed only in advanced view.

The list table on the upper half of the **Events** tab lists all the events contained in the stub.

When a stub receives an event, it searches the list of events from top to bottom to find an event that has been set up to handle the event received, based on the stub's current state. You can use message filters and guards if you want to ensure that the contents of a message determines whether and how an event is processed.

The following table describes the fields in the list table.

Field	View	Entry	Description
From State	Advanced view only	Optional	<p>This field enables you to specify an initial state for an event.</p> <p>For example, if you are building a stub that can track user sessions, you might want to have an initial state called <code>LoggedOut</code> for an operation <code>Login</code> operation. (The final state for the corresponding <code>Logout</code> operation might be <code>LoggedIn</code>.)</p> <p>Clicking the field displays a drop-down list that lists any states already defined for the stub and an Add new... option that enables you to create a new state.</p> <p>Clicking Add new... in the From State list opens the New State dialog box. In the New State dialog box, you must enter a name for the new state in the Name field. You can enter a description for the new state in the Description field but that is optional. Clicking OK closes the New State dialog box.</p> <p>If this field and the To State field are left blank, Rational Integration Tester will assume that this event will work for any state of the currently selected stub, so the stub will be stateless.</p> <p>You can also create a new state by clicking New next to the States field on the Properties tab. (The Edit and Delete buttons next to the States field enable you to edit and delete states as required.)</p> <p>NOTE: The From State field becomes editable only after an event has been selected for the event.</p>

Field	View	Entry	Description
Event	Basic and Advanced views	Mandatory	<p>In computing generally, an “event” is something that happens that affects the system. In Rational Test Virtualization Server, an event refers to an operation.</p> <p>This field enables you to:</p> <ul style="list-style-type: none"> • Select one of the behaviours displayed on the Behaviour tab (for information about this, refer to Behaviour Tab). • Specify the operation associated with the currently selected transition event. <p>When creating a new event, clicking this field displays a drop-down list that includes any operations already associated with the stub and a Browse... option that enables you to select a different operation.</p> <p>Clicking Browse... in the Event list opens the Select an Operation dialog box. In the Select an Operation dialog box, select an operation from your project’s logical resources and click OK.</p> <p>NOTE: This field is not editable until after an event has been selected for the event.</p>
Guard	Basic and Advanced views	Optional	<p>This field enables you to specify conditions that will determine if an event will be handled. For example, you could create a guard based on the data held in a data model associated with the currently selected stub.</p> <p>For information about the options in this drop-down list, refer to Guard Condition Options.</p> <p>NOTE: You must use the Input tab on the lower half of the Events tab to configure the guard condition of an event. (For information about using the Input tab, refer to Input Tab.)</p> <p>NOTE: This field is not available until after an event has been specified for the currently selected event.</p>

Field	View	Entry	Description
To State	Advanced view only	Optional	<p>This field enables you to specify a final state for an event.</p> <p>For example, if you are building a stub that can track user sessions, you might want to have a final state called <code>LoggedIn</code> for an operation called <code>Login</code>. (The final state for the corresponding <code>Logout</code> operation might be <code>LoggedOut</code>.)</p> <p>Clicking the field displays a drop-down list that lists any states already defined for the stub and an Add new... option that enables you to create a new state.</p> <p>Clicking Add new... in the To list opens the New State dialog box. In the New State dialog box, you must enter a name for the new state in the Name field. You can enter a description for the new state in the Description field but that is optional. Clicking OK closes the New State dialog box.</p> <p>If this field and the From State field are left blank, Rational Integration Tester will assume that this event will work for any state of the currently selected stub, so the stub will be stateless.</p> <p>You can also create a new state by clicking New next to the States field on the Properties tab. (The Edit and Delete buttons next to the States field enable you to edit and delete states as required.)</p> <p>NOTE: The To State field becomes editable only after an event has been selected for the event.</p>

Field	View	Entry	Description
Timings	Advanced views only	Optional	<p>This field enables you to specify a timing override for the currently selected event. If a timing override is set for a specific event, it will override any response times set for the stub. (For information about setting the response times of a stub, refer to Behaviour Tab.)</p> <p>Double-clicking this field opens the Timing Override dialog box. In the Timing Override dialog box, select an option in the Delay Distribution list, enter minimum and maximum delays (in milliseconds) in the fields provided, and click OK. Clicking Cancel or Clear closes the dialog box without saving any changes.</p> <p>NOTE: This field is not available until after an event has been specified for the event.</p>
Description	Basic and Advanced views only	Optional	<p>This field enables you to enter a narrative description for the currently selected event.</p> <p>To enter a description for an event, select it and double-click this field.</p> <p>NOTE: This field is not editable until after an event has been specified for the event.</p>

Guard Condition Options

Guard conditions are Boolean expressions that affect the behaviour of a stub by enabling actions or events only when they evaluate to “TRUE” and disabling them when they evaluate to “FALSE”.

The following table describes the **Guard** drop-down list options in the list table on the **Events** tab.

Option	Description
None	<p>Click this option (it is also the default) if you do not want to specify any guard conditions for the currently selected event.</p> <p>The Guard field on the list table will be blank if this option is clicked or if none of the Guard drop-down list options is clicked.</p>

Option	Description
Record Exists	<p>NOTE: This option is displayed only if there is a data model associated with the currently selected stub.</p> <p>Click this option if a data model exists and you want to run the business logic/event if the record referenced by the incoming message exists.</p> <p>If this option is clicked, <code>Record Exists</code> will be displayed under Guard on the list table for the currently selected event.</p> <p>For a data model-driven message-based stub, <code>Record Exists</code> causes the stub to use mappings (from message contents to data model tag names) defined in the Message Editor when looking up records in the data model.</p> <p>If a record already exists, the <code>Record Exists</code> guard passes. If a record does not already exist, the <code>Record Doesn't Exist</code> guard passes.</p> <p>This enables you to have two events for an incoming message and to respond differently depending on whether the specified data is found.</p> <p>For example, if the service is "Find Customer" and the "Customer ID" field in the message is key field, customer data or an error message can be returned by using the two events and the guards without having to write the business logic.</p>
Record Doesn't Exist	<p>NOTE: This option is displayed only if there is a data model associated with the currently selected stub.</p> <p>Click this option if a data model exists and you want to run the business logic/event if the record in the data model does not exist.</p> <p>If this option is clicked, <code>Record Doesn't Exist</code> will be displayed under Guard on the list table for the currently selected event.</p>
Expression	<p>Click this option if you want to specify a custom Boolean expression for the currently selected event.</p> <p>For more information about this option, refer to Configuring Guard Conditions.</p>
No Match	<p>Click this option if you want an event's specified event (operation) to change state if a matching record is not found.</p> <p>If this option is clicked, <code>No Match</code> will be displayed under Guard on the list table for the currently selected event.</p> <p>NOTE: Essentially, "no match" is the default case of the message. For each operation, there can be only one event with the "no match" guard because it is invalid to have two default cases.</p>

3.3.1.2 Input Tab

For each event, this tab enables you to:

- Configure a guard condition (optional).
- Specify Incoming message field actions (optional).

For example, if you have created a basic (hard-coded) stubs from recorded events (for information about this, refer to [Creating Basic Stubs](#)), the stub will filter out any messages that do not match exactly the message(s) received while recording the events used to build the stub, so the stub will not send a response message.

However, if you want that basic stub to respond to any messages that match the message structure in the stub irrespective of the values within the fields of those messages, you can use the **Input** tab to disable any filtering that checks for exact field matches and thus make your stub more versatile.

The tab comprises a window for configuring a guard condition for an event, and two windows for setting incoming message field actions (for an event).

The following sections describe how to use these controls.

Configuring Guard Conditions

Rational Integration Tester tags can be used to create guard conditions. (For information about creating and using tags, refer to *IBM Rational Integration Tester Reference Guide*.)

If the **Expression** option is clicked in the **Guard** drop-down list in the list table on the upper-half of the **Events** tab, a drop-down list, a text box, and buttons are displayed on the **Input** tab. The controls enable you to create and test your own Boolean expressions.

The drop-down list contains the following options:

- **ECMAScript** (default; for example, JavaScript)
- **Legacy**

Clicking **ECMAScript** displays a **Test** button. Clicking **Test** validates your custom Boolean expression and displays an Output dialog box that lists any errors.

Clicking **Legacy** displays **Add**, **Delete**, and **Test** buttons and an **'OR' Expressions** check box. Clicking **Add** makes the text box available. Clicking **Test** validates your custom Boolean expression and displays an Output dialog box that lists any errors.

If a custom Boolean expression is entered, the starting characters of the expression will be displayed under **Guard** on the list table for the currently selected event.

Incoming Message Field Actions

The Message Header and Message Body windows on the **Input** tab on the **Events** tab enable you to specify actions for each field in the incoming message.

Double-clicking any field on the windows opens the Field Editor dialog box. The **Action** column is a single view of the action groups (Filter, Validate, and Store) for incoming messages.

For example, if you want to disable any filtering for exact field matches:

1. Select all of the fields on the **Input** tab.
2. Right-click the selected fields and click **Contents > Field Actions > Filter > Equality** on the shortcut menus.

Equality checking for the selected fields is disabled and fewer filter icons are displayed on the **Input** tab.

NOTE: Double-clicking a field opens the Field Editor dialog box. (For information about using the Field Editor dialog box, refer to [Field Editor](#).)

3.3.1.3 Business Logic Tab

This tab enables you to specify what actions the stub will take when it executes an event. For data model stubs, it is possible to configure the stub to create, read, update or delete (CRUD) automatically from the associated data model.

If you are not using a data model stub or simple hard-coded stub, you will need to write business logic using actions in the same way that you would for writing a test.

A parameterized stub that uses data sets will need a Lookup action (or similar) in its business logic to retrieve the data. If you created a stub with a data set from the wizard, these actions will have been added automatically into the business logic by the wizard.

The following table describes the five option buttons on the **Business Logic** tab on the **Events** tab.

Option Button	Description
None	Click this option button (it is also the default) if you do not want to specify any processing rules for any incoming messages for the currently selected event.

Option Button	Description
Create	Click this option button if the stub has an associated data model and you want the result of this operation to be a new record in the data model based on the content of the incoming message.
Update or Create	Click this option button if the stub has an associated data model and you want the result of this operation to be a new or updated record in the data model based on the content of the incoming message.
Delete	Click this option button if the stub has an associated data model and you want the result of this operation to be a deleted record in the data model based on the content of the incoming message.
Other	<p>Click this option button to create custom processing rules for any incoming messages for the currently selected event.</p> <p>Clicking this option button opens a Test Editor window on the Business Logic tab.</p> <p>The Test Editor window enables you to define actions that will be executed while processing any incoming messages. (For information about using the Test Editor, refer to <i>IBM Rational Integration Tester Reference Guide</i>.)</p>

NOTE: If you are using Rational Integration Tester 8.0.1 (or later), there is a **Create new Pass Through** action on the toolbar on the **Business Logic** tab that enables you to define explicit pass-through behaviour that must be executed if the currently selected stub cannot process a received message and the transport involved supports the sift-and-pass-through capability available in Rational Integration Tester 8.0.1 (or later). (For more information about this, refer to [Sift-and-Pass-Through Capability](#).)

3.3.1.4 Output Tab

For each event, this optional tab enables you to specify outgoing (reply) message field values.

The tab comprises a **Send Response** check box, a reply header window, and a reply message (body) window.

Select the **Send Response** check box if you want a reply message to be sent in response to each processed incoming message.

The following sections describe how to use the windows on the tab.

Reply Header Window

The Reply Header window is on the upper half of the **Output** tab on the **Events** tab. The window enables you to override transport-specific response settings and header metadata of the reply message for the currently selected event.

For information about editing message headers, refer to *IBM Rational Integration Tester Reference Guide*.

Reply Message Window

The Reply Message window is on the lower half of the **Output** tab on the **Events** tab. The window enables you to view and configure the validation of the reply message for the currently selected event.

Depending on the transport associated with the operation that is associated with the currently selected event, the message-type can be selected by using the **Message Type** list above the message body.

The body of the reply message is displayed in a Publisher/Subscriber view that includes enabled values, which enables you to select or clear the **Enabled Value** check box of each element/field, thus enabling or disabling the value specified for the content of that element/field.

Double-clicking an field on the reply message window opens the Field Editor dialog box. The **Value** column is a single view of the action groups (Value, Validate, and Store) for reply messages. (For information about using the Field Editor dialog box, refer to [Field Editor](#).)

3.3.1.5 Field Editor

The Field Editor dialog box enables you to specify how a message field is populated.

It comprises three tabs (action groups) but the tabs displayed depend on message-type.

Field Editor Tabs for Incoming Messages	Field Editor Tabs for Outgoing Messages
Filter	Value
Validate	Validate
Store	Store

To open the Field Editor dialog box:

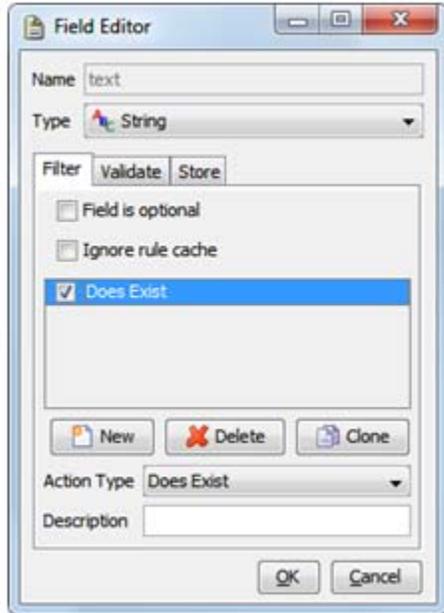
- Double-click a field or element name on the **Input** or **Output** tabs on the **Events** tab of the Stub Editor.
- Alternatively, right-click a field on the **Input** or **Output** tabs on the **Events** tab of the Stub Editor and click **Contents > Edit** on the shortcut menus.

NOTE: The performance of the Field Editor will be degraded if a single line of XML is larger than 50,000 bytes. However, this problem will not arise if the XML is on multiple lines (within reason).

The following sections describe how to use the tabs on the Field Editor dialog box.

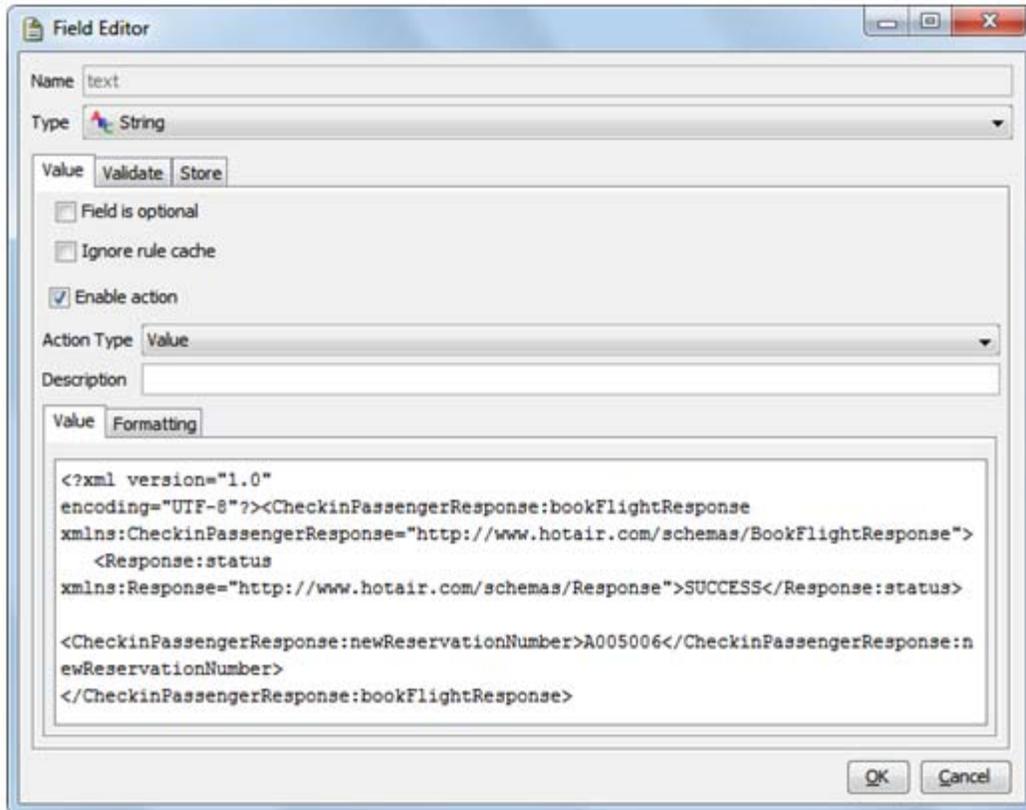
Filter Tab

The **Filter** tab enables you to restrict which messages a subscriber will process based on the content of the selected field.

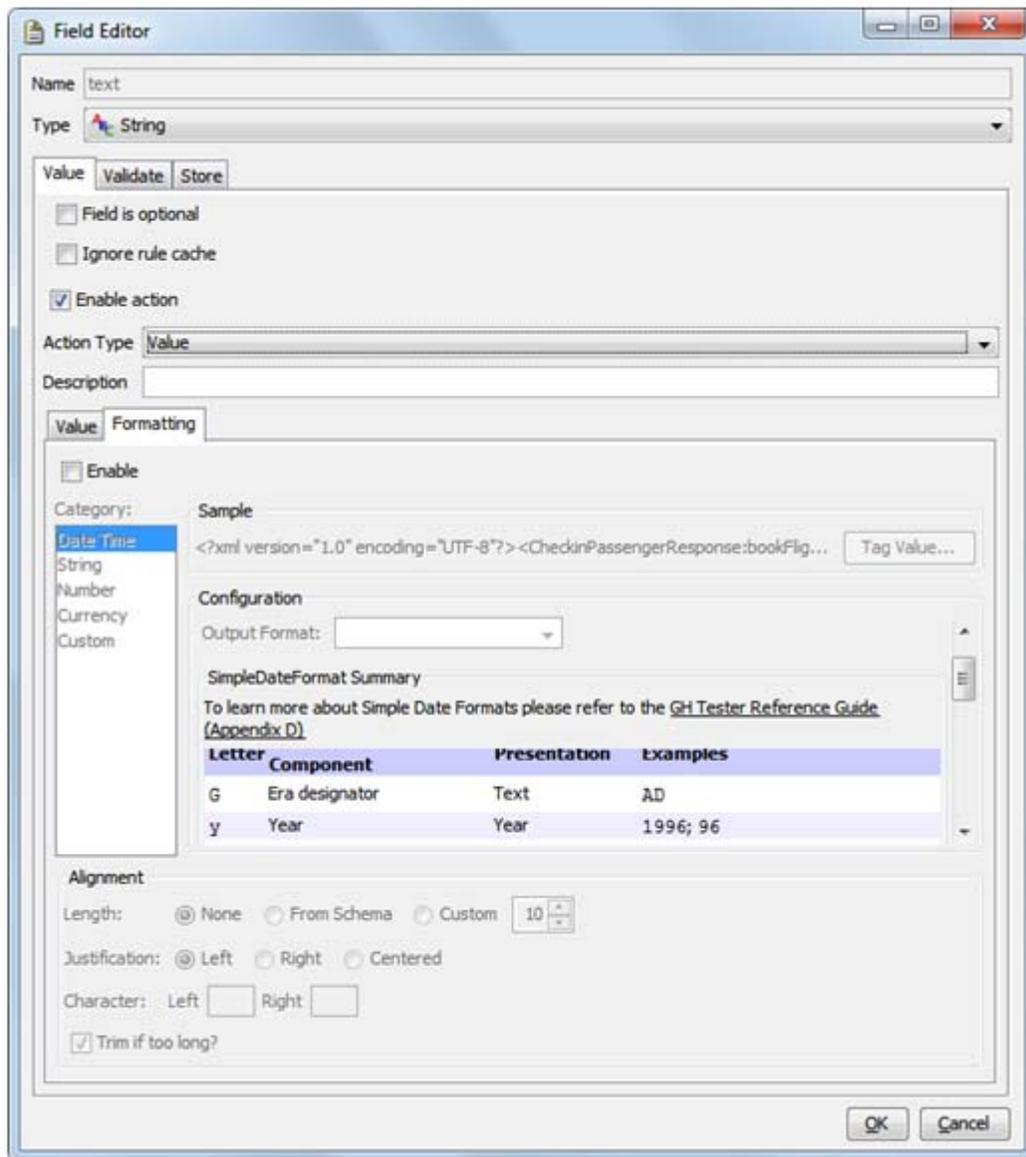


Value Tab

The **Value** tab enables you to specify what value the field should take (for example, when part of a published message).

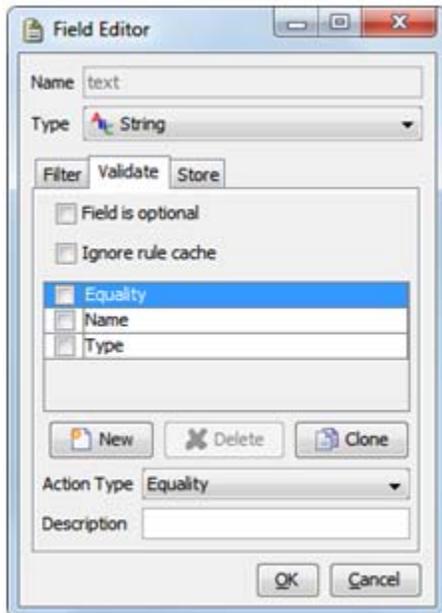


Under the **Formatting** tab (on the **Value** tab), various types of formatting can be applied to field values to generate random values at runtime.



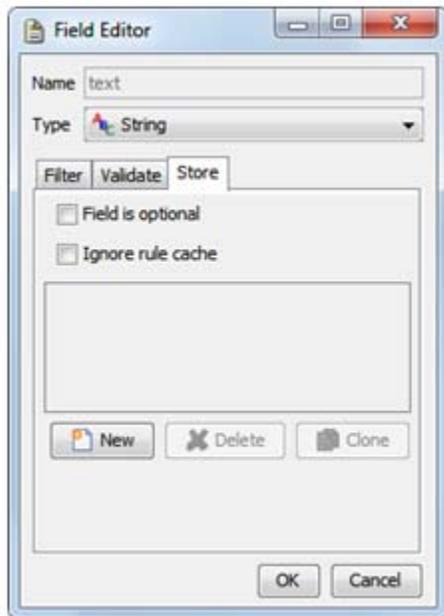
Validate Tab

The **Validate** tab enables you to define how to compare or verify the contents of a field that requires validation (for example, in the context of subscription).



Store Tab

The **Store** tab enables you to store the contents of the field in a tag (for example, when receiving a message, or to record the value of a dynamic field in a published message).



Action-Types

There is only one way at a time to populate a value but there are many different ways to validate or store a value. Therefore, the **Validate** and **Store** tabs can accommodate one or more action-types.

Action-Type	Message-Types	Field Editor Tabs	Description
Does Exist	Incoming	Filter	Accept or ignore incoming messages depending on whether the selected field exists in each message.
Is Null	Incoming, Outgoing	Filter, Validate	Verify whether the selected field is null.
Not Null	Incoming, Outgoing	Filter, Validate	Verify whether the selected field contains a value.

Action-Type	Message-Types	Field Editor Tabs	Description
Name	Incoming	Filter	Accept or ignore incoming messages depending on the name of the selected field in each message.
Type	Incoming	Filter	Accept or ignore incoming messages depending on the type of the selected field in each message.
Assert using Function	Incoming, Outgoing	Filter, Validate	Use a Rational Integration Tester function (that returns a Boolean type) to filter/validate the selected field.
Validate Element Children	Incoming	Validate	Validate the children of the selected field.
Validate Using Tag	Incoming	Validate	Validate a received value against a message or sub-message that has been previously tagged.
Value	Outgoing	Value	Validate the contents of the selected of each outgoing message against a specific manually entered value.
File	Outgoing	Value	Populate the contents of the selected field with the contents of a selected file. You must also select the file.

Action-Type	Message-Types	Field Editor Tabs	Description
Function	Outgoing	Value	Execute a Rational Integration Tester function (including custom functions). The result of the function is used as the field value.
Decrement	Outgoing	Value	Decrease the selected field's value by a specified amount each time that the outgoing message is sent. This involves setting an initial value and a step value.
Increment	Outgoing	Value	Increase the selected field's value by a specified amount each time that the outgoing message is sent. This involves setting an initial value and a step value.

Action-Type	Message-Types	Field Editor Tabs	Description
List	Outgoing	Value	<p>Rational Integration Tester treats each entry in the specified input as a separate value.</p> <p>Values can be separated using any available delimiters (new line, comma, tab, or full stop).</p> <p>The first time that the selected outgoing message is sent, the selected field will be set to the first value.</p> <p>For each subsequent iteration, the next value in the sequence will be used.</p> <p>If a message is published more times than the number of available values, the cycle will restart from the first value.</p> <p>You can also reset the action and restart from the first value.</p>
Null	Outgoing	Value	Clear the contents of the selected field.
Equality	Outgoing	Validate	Verify whether each outgoing message contains a value in the selected field that matches the specified value.

Action-Type	Message-Types	Field Editor Tabs	Description
Length	Outgoing	Validate	Verify whether each outgoing message contains a value in the selected field that falls within the specified maximum and minimum values.
Regex	Outgoing	Validate	Use the specified regular expression to validate the contents of the selected field of each outgoing message.
XPath	Outgoing	Validate	Use the specified XPath expression to validate the contents of the selected field of each outgoing message.
Schema	Outgoing	Validate	Use the specified schema to validate the contents of the selected field of each outgoing message.
XSD Type	Outgoing	Validate	Verify whether the field is of the correct type as specified by the XSD used to build the message.

NOTE: The availability of action-types depends on the transport, formatter, and field-type selected.

Buttons

The following table describes the default buttons on the tabs on the Field Editor dialog box.

Button	Description
New	Add a new validation to the selected field.
Delete	Delete the selected validation from the selected field.
Clone	Duplicate the selected validation for the selected field. NOTE: You cannot duplicate Name and Type validations.

Depending on the action-types selected on some tabs, other buttons are also displayed.

Check Boxes

The following table describes the default check boxes on the tabs on the Field Editor dialog box.

Check Box	Description
Field is optional	
Ignore rule cache	Ignore a specific action-type within a rule that has been set on the selected field
Does Exist	
Accept fields in any order	Rational Integration Tester will ignore the order in which message fields are received. This can be useful since messages sent on certain transports may undergo field-reordering.
Ignore missing fields in received messages	When enabled, any fields present in the expected message structure but absent in the message received will not cause invalidation.
Ignore additional fields in received messages	When enabled, any fields present in the received message but absent in the expected message structure will not cause invalidation.
Enable action	
Enable (Formatting)	

Depending on the action-types selected on some tabs, other check boxes are also displayed.

3.3.2 Behaviour Tab

This optional tab enables you to add reusable behavioural entities, that is, preprogrammed intelligence, to your stub.

NOTE: This tab is not displayed when editing a deterministic stub.

A behaviour can act as a source of events that cause a stub to execute an operation that is not a reaction to an incoming message from an external system. For example, you might want to create a stub that can publish proactively messages that are not just “responses” to a incoming message.

After you have defined one or more behaviours on the **Behaviour** tab, you can use them as “events” on the **Events** tab.

The upper half of the **Behaviour** tab comprises:

- A window that lists any behaviours created for the stub, and buttons for creating, modifying, and deleting behavioural entities.
- A text box that displays a description for each selected behaviour.

The lower half of the tab comprises a Configuration window that displays details about any behavioural entity with custom instance names.

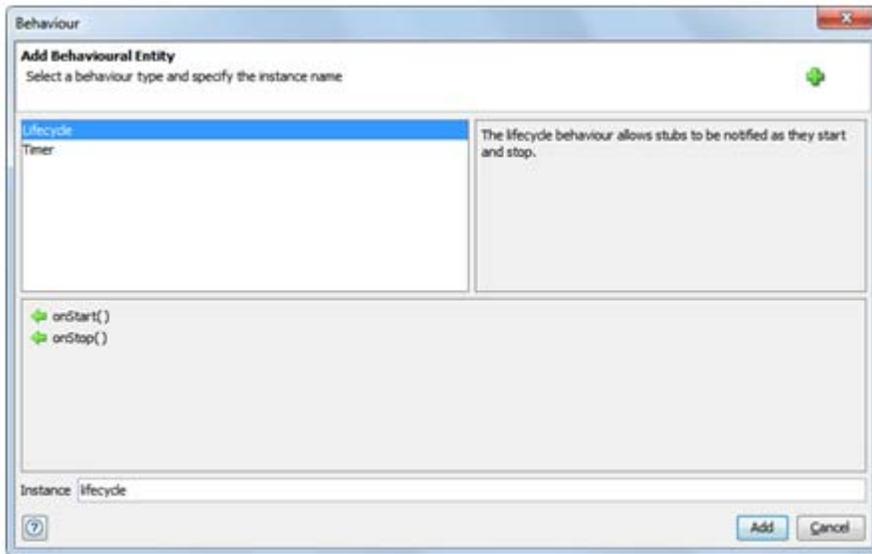
The following table outlines the default behaviour-types that are supplied with Rational Integration Tester.

Behaviour-Type	Description
Lifecycle	This enables you to execute certain actions when the currently selected stub starts up or shuts down. For example, you can set up and later clean up any resources that may be required by the stub.
Timer	This enables you to set up a timer that schedules a future callback after a specified delay so that the stub will receive an event to which it can respond. NOTE: Callback timer settings for a stub can be overridden for each event created for the stub (for information about this, refer to Events Tab).

NOTE: Rational Integration Tester provides the means to create additional custom behaviours. For more information about this, refer to [Appendix A: Using the Data Model Editor](#).

The following table describes the buttons on the upper half of the **Behaviour** tab.

Button	Description
Add	<p>Clicking this button opens the Add Behavioural Entity dialog box.</p> <p>The upper left side of the Add Behavioural Entity dialog box displays the available behaviour-types. The upper right side of the Add Behavioural Entity dialog box displays a description for each selected behaviour-type.</p> <p>The default available behaviour-types are as follows:</p> <ul style="list-style-type: none">• Lifecycle• Timer <p>The lower half of the Add Behavioural Entity dialog box displays the available behaviours for the currently selected behaviour-type and an Instance field, which is an optional field that enables you to create multiple instances of a behaviour.</p> <p>NOTE: To select multiple behaviours for a behavioural entity, hold down CTRL and click each behaviour after selecting a behaviour-type on the upper left side of the Add Behavioural Entity dialog box.</p> <p>After specifying the properties of a behaviour, click Add to complete creating the behaviour and to close the Behaviour dialog box.</p> <p>NOTE: In addition to being displayed on the list window on the upper left side of the Behaviour tab, each newly created behavioural entity is also added to the Actor list on the Events tab for the stub (for information about the Actor field, refer to refer to Events Tab).</p>
Edit	<p>Clicking this button opens the Edit Behavioural Entity dialog box, which enables you to modify instance details for the selected behavioural entity.</p>
Delete	<p>Clicking this button deletes any behavioural entities selected in the list window on the tab.</p> <p>NOTE: To select multiple behavioural entities for deletion, hold down CTRL and click each entity.</p>



3.3.3 Properties Tab

This tab enables you to set up states for your stub and input parameters that can determine how the stub should behave while it is executed.

The upper half of the tab comprises fields and lists that enable you to specify response times that control the performance of the stub, the input tags that should be available to the stub at run time if the stub is part of a test suite, and the number of concurrent instances that can be running at any one time.

The following table describes the fields and lists on the upper half of the **Properties** tab.

Control	Description
Delay Distribution	This list controls the stub's performance, enabling you to create realistic system conditions in a single stub (that is, without having to create multiple stub instances). Delay pattern options are as follows: <ul style="list-style-type: none">• Fixed• Uniform• Gaussian
Minimum Delay (ms)	This field specifies the minimum delay time (in milliseconds) for the delay-type selected in the Delay Distribution list.
Maximum Delay (ms)	This field specifies the maximum delay time (in milliseconds) for the delay-type selected in the Delay Distribution list.
Input	The window and the buttons provided enable you to define the input tags that should be available to the stub at run time if the stub is part of a test suite. All tags are available at run time because they are in the Tag Data Store. These input tags are available to test suites. Specifically, they are displayed on the Data Drive tab of the Scenario Editor dialog box where values can be entered for these tags when the currently selected stub is being started. (For more information about this, refer to <i>IBM Rational Integration Tester Reference Guide</i> .) When creating or modifying a tag, the interface definition can be set by selecting the Expose as input check box on the Create Tag dialog box. (For more information about creating and managing tags, refer to <i>IBM Rational Integration Tester Reference Guide</i> .)
Workers	This field specifies the maximum number of concurrent instances of this stub that can be running at any one time. Default value: 10.

Control	Description
Style	<p>This list enables you to control how the stub will execute. Options are as follows:</p> <ul style="list-style-type: none"> • One-to-one, no looping: For each new request (message or connection) made to the stub, a thread will be taken from the pool of worker threads. After the stub's actions have completed, the thread will be returned to the pool. If there are no worker threads available, the request will wait until one becomes available. • One-to-one, looping: A thread will be assigned from the pool of worker threads to each connection. Each connection's thread will be used for all requests from that connection until the connection is closed. This enables tag values to persist between requests on a connection. If the number of open connections equals the value entered in the Workers field, there will be no further connections until a thread becomes available. This style should be used only for long-lived connections to stubs, such as TCP clients. For other connection types, the behavior of this style is identical to that of the One-to-one, no looping style. • Many-to-one: If this style is selected, the Workers field is ignored, a new thread will be created for each incoming request and destroyed once the request has been processed. This means that Rational Integration Tester will attempt to process a message immediately upon receipt but overall performance will degrade if a large number of requests occur simultaneously.
Data model	This list is used to specify the data model (if any) of the stub.
Version (major.minor)	These fields enable you to implement a version numbering system for your stubs (if necessary).

The lower half of the tab enables you to define pass-through actions, the session states that should be available to the stub, and any environment tasks that need to be run before the stub starts.

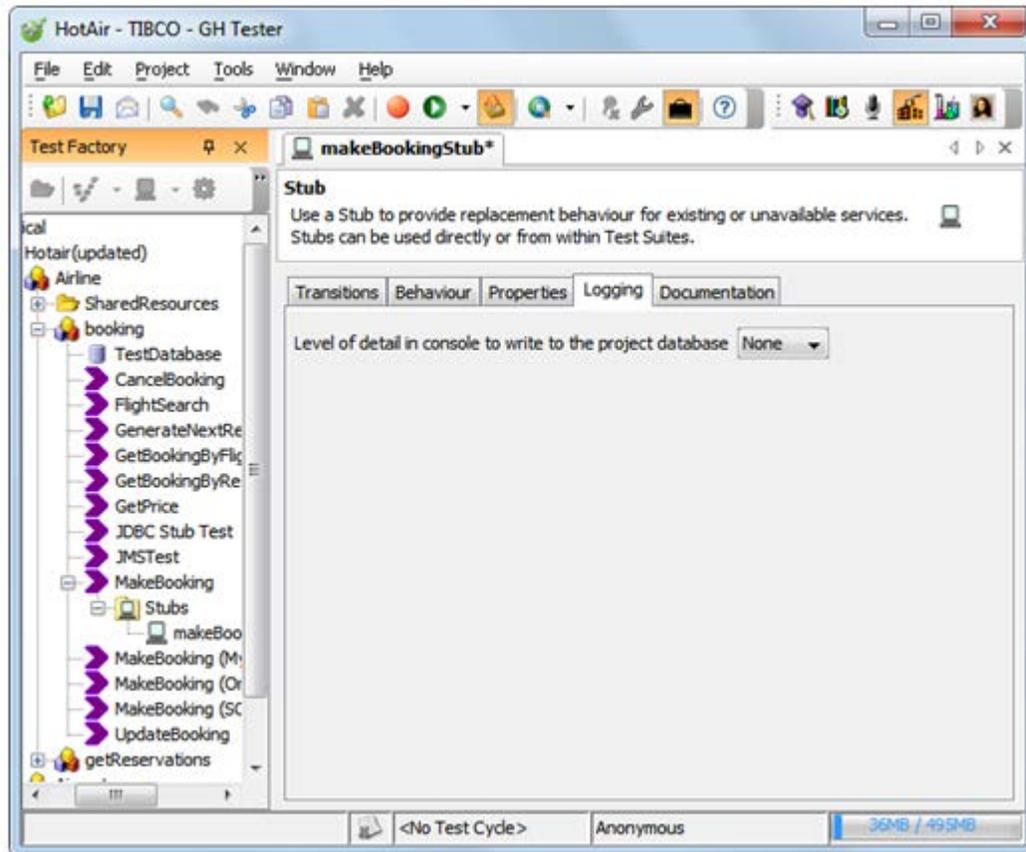
The following table describes the fields and lists on the lower half of the **Properties** tab

Field/List	Description
Pass Through	<p>The Pass Through window lists only the set of operations for which the stub is providing virtual services (refer to Events tab) and which have transports that have sift-and-pass-through capability.</p> <p>NOTE: For background information about this window, refer to Sift-and-Pass-Through Capability.</p> <p>To configure an event's pass-through action:</p> <ol style="list-style-type: none">1. Under Configuration on the Pass Through window, click the event. The Pass Through Configuration dialog box is displayed.2. In the Pass through action list, click one of the following options:<ul style="list-style-type: none">• Discard: HTTP and IBM WebSphere MQ transports only. (For more information, refer to Configuring Transports to Use the Sift-and-Pass-Through Capability.)• Pass Through: For more information, refer to Configuring Transports to Use the Sift-and-Pass-Through Capability.• Simulate Error: HTTP and webMethods Integration Server transports only. (For more information, refer to Configuring Transports to Use the Sift-and-Pass-Through Capability.) <p>NOTE: The list of options displayed depends on the transport type associated with the currently selected operation.</p> <ol style="list-style-type: none">3. Click OK.

Field/List	Description
States	<p>This field specifies the session states that should be available for the stub. These will be added automatically when you add states on the Events tab.</p> <p>One or more states can be entered in this field by:</p> <ul style="list-style-type: none"> • Clicking New beside the field. • Entering details about each state in the New State dialog box. • Clicking OK. <p>NOTE: You can also add new states on the Events tab by selecting any row under From or To on the list table and clicking Add new...</p> <p>When a message case is used within the stub, the state of the stub can be checked before any actions in the case are executed, and a new state can be optionally set after the actions in the case have complete executing.</p> <p>The session state is also stored in a system tag named <code>SESSION/STATE</code>. If desired, the session state could be set by using the <code>setTag</code> function.</p>
Initial State	<p>This field specifies the stub's initial state.</p>
Conversation Keys	<p>This field specifies the conversation (session) keys that can be used to identify a particular conversation.</p> <p>For example, if you are building a stub that can track user sessions, you might want to have a conversation key called <code>SessionID</code>.</p> <p>One or more keys (separated by a semicolon) can be defined. These keys are mapped to a system tag named <code>SESSION/KEY/<Key Name></code>.</p> <p>When an incoming message is received, key values should be stored in these tags, which are then used to identify the conversation to use.</p>
Environment Tasks	<p>Configure any environment tasks that must be executed before the currently selected stub starts.</p>

3.3.4 Logging Tab

This tab enables you to define how much information is recorded while your stub is being executed.



Any information recorded will be logged in Rational Integration Tester's project results database and on the Test Factory perspective's Console window.

The following table describes the list options on the **Logging** tab.

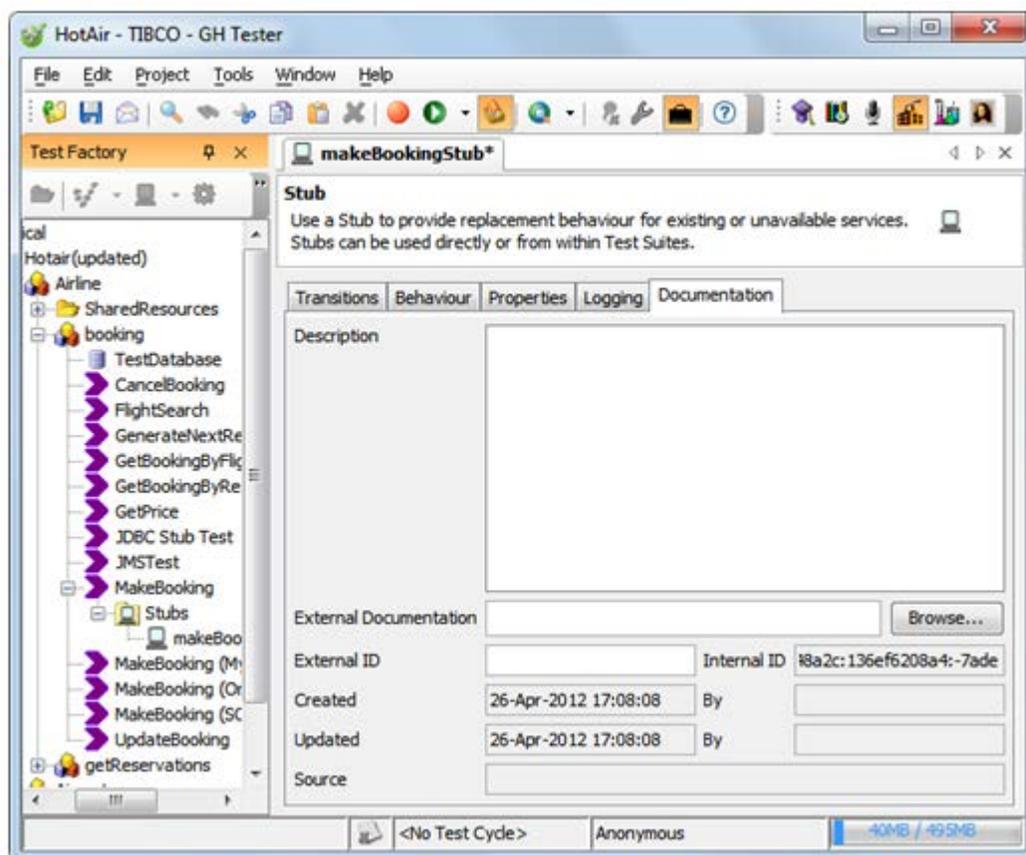
List Option	Description
None	No information is logged.
Normal	Standard information is logged.
Debug	Verbose information is logged.

NOTE: The default list option displayed on the **Logging** tab is determined by the setting of the **Stub default log level** list on the **Console** page of

the Preferences dialog box, which is opened by clicking **Project > Preferences** or **Window > Preferences** on the menu bar. Therefore, selecting (for the currently selected stub) a different log level on the **Logging** tab will overwrite the default log level specified in the Preferences dialog box but only for that stub.

3.3.5 Documentation Tab

This tab enables you to enter extra information about your stub and some of this extra information will be displayed in Rational Test Control Panel's Start Stub dialog box (for information about this dialog box, refer to [Starting Stubs](#)).



Although you do not have to enter such information, it may help future users of your stub and it may also help you to identify in Rational Test Control Panel which stub you want to start if you have created several stubs for a particular operation.

For more information about using this tab, which is a standard tab on many dialog boxes in Rational Integration Tester, refer to *IBM Rational Integration Tester Reference Guide*.

3.3.6 Sift-and-Pass-Through Capability

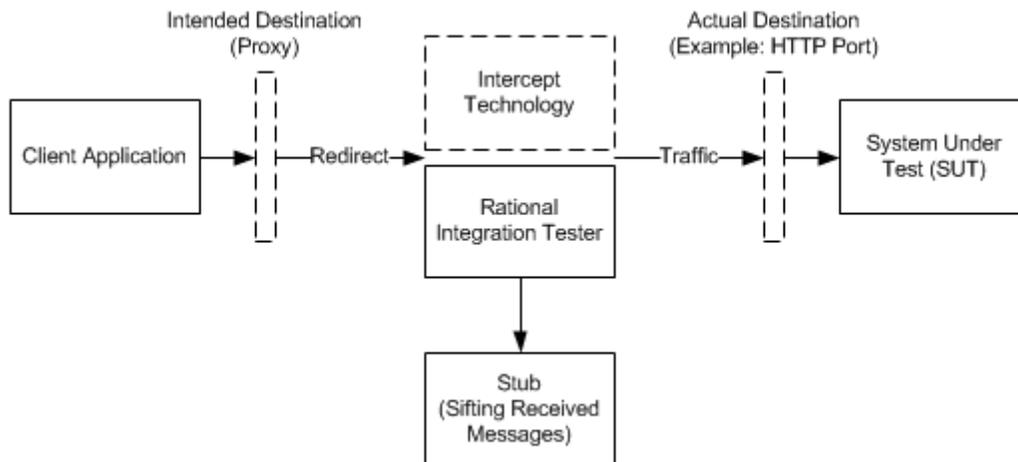
Before Rational Integration Tester 8.0.1, messages received by a stub were handled on “all or nothing” basis because the stub could not be configured to subdivide traffic from client and server applications that needs to use the real system as opposed to a virtual system.

In Rational Integration Tester 8.0.1 (or later):

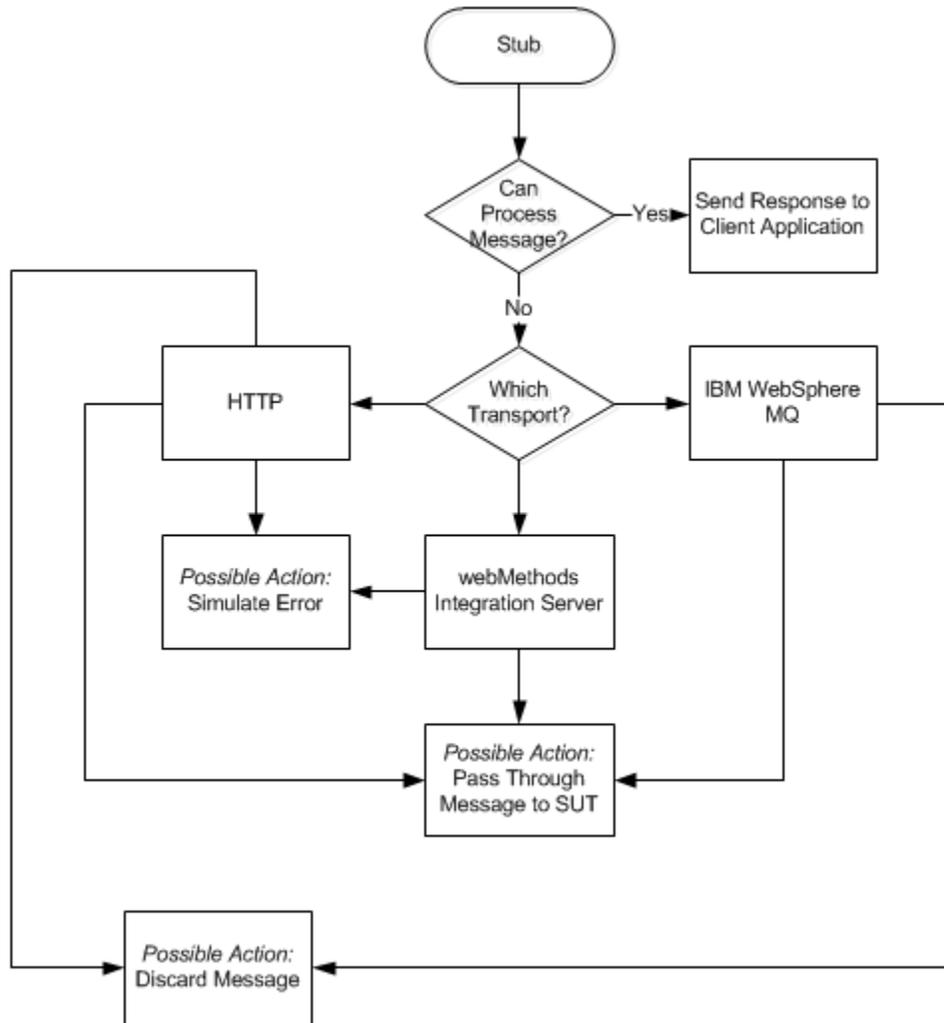
- Traffic from HTTP, IBM WebSphere MQ, or WebMethods Integration Server transports can be routed to a stub by examining data in the traffic on an exact matching basis.
- Logic can be added to a stub to enable it to determine whether it should respond to a received request (“sifting”) or, when possible, send the traffic for onward processing to the real system (“pass-through”).

NOTE: Currently, this “sift-and-pass-through” capability cannot be used with other transports supported by Rational Integration Tester.

Consider the scenario illustrated in the following diagram.



In Rational Integration 8.0.1 (or later), depending on the logic that has been added to the stub, the following graphic illustrates how the stub will respond to any received messages.



To use the sift-and-pass-through capability:

1. Configure the relevant transport (HTTP, IBM WebSphere MQ, or WebMethods Integration Server) to use the capability.
2. Add the required business logic to the stub to enable sifting.
3. Configure pass-through behaviour for each relevant operation in the stub.

The following sections describe these steps.

3.3.6.1 Configuring Transports to Use the Sift-and-Pass-Through Capability

The following table outlines the options for configuring an HTTP, IBM WebSphere MQ, or WebMethods Integration Server transport to use the sift-and-pass-through capability provided by Rational Integration Tester 8.0.1 (or later).

Transport	Description
HTTP	You can configure this transport to have a default pass-through action (see below).
webMethods Integration Server	
IBM WebSphere MQ	Options are as follows: <ul style="list-style-type: none">• Configure a default pass-through action (see below).• Configure a default stubbing mode for each new transport created (see below).• Modify the stubbing mode of each transport irrespective of the version of Rational Integration Tester that was used to create it (for information about this, refer to <i>IBM Rational Integration Tester Reference Guide for IBM WebSphere MQ</i>.)

To configure default pass-through settings for HTTP, IBM WebSphere MQ, or WebMethods Integration Server transports:

1. Click **Project > Preferences** or **Window > Preferences** on the menu bar.
The Preferences dialog box is displayed.
2. Click **Virtualization**.
The Virtualization page is displayed.
3. Under **Pass Through**, double-click the transport that you want to configure.
The Pass Through Configuration dialog box is displayed.

The options displayed in the **Pass Through action** list on the Pass Through Configuration dialog box depend on the transport selected.

The following table describes how to use the fields.

Option	Transport	Description
Discard	HTTP	Rational Integration Tester will discard any message that the stub cannot process and not send any reply.
	IBM WebSphere MQ	

Option	Transport	Description
Pass Through	HTTP	Any message will pass through to the original target of the operation.
	IBM WebSphere MQ webMethods Integration Server	Optional: In the Delay field, enter a delay (in milliseconds).
Simulate Error	HTTP	Rational Integration Tester will send a fake error message back to the client or server application. Instructions: 1. Optional: In the Delay field, enter a delay (in milliseconds). 2. Mandatory: In the Status Code list, click a status code for the error. 3. Optional: In the Status Text field, enter status text for the error.
	webMethods Integration Server	Rational Integration Tester will send a fake error message back to the client or server application. Instructions: 1. Optional: In the Delay field, enter a delay (in milliseconds). 2. Mandatory: In the Exception list, click a supported exception type for the error. 3. Optional: In the Message field, enter text for the error.

4. Click **OK** to close the Pass Through Configuration dialog box.
5. Click **Apply** to save your changes if you want to modify other settings on the Preferences dialog box before closing it. Otherwise, click **OK** to close the Preferences dialog box.

3.3.6.2 Adding Business Logic

Logic can be added to a stub to enable it to determine whether and how it should respond to a received request.

To facilitate this, there is a **Create a new Pass Through** action button on the toolbar of the **Business Logic** tab (for information about this tab, refer to [Business Logic Tab](#)) that enables you to define explicit pass-through behaviour that must be executed depending on the logic added to the stub.

For example, if a received message contains data that matches a particular condition, the configured pass-through action could generate an error. Alternatively, the action could be configured to discard the message or to let the message pass through to the system under test (the possible actions depend on the transport being used).

3.3.6.3 Configuring Stub Pass-Through Behaviour

There are three methods of configuring a stub to use pass-through actions. The following table outlines these methods.

NOTE: The transport associated with a stub must be configured to use the Pass Through action. (For information about this, refer to [Properties Tab Configuring Transports to Use the Sift-and-Pass-Through Capability](#).)

Method	Application	Description
Use the Pass-Through window on the Properties tab of the Stub Editor	Rational Integration Tester 8.0.1 (or later)	<p>The transport type (HTTP, IBM WebSphere MQ, or WebMethods Integration Server) associated with a stub determines the range of possible pass-through actions for each operation within a stub.</p> <p>The Pass Through window on the Properties tab on the Stub Editor is used to configure a specific pass-through action for each operation in the stub (irrespective of the number of events that are defined for each operation) that will be executed if the stub cannot process any requests from any of the relevant transport types (HTTP, IBM WebSphere MQ, or WebMethods Integration Server).</p> <p>Irrespective of the number of events (configured on the Events tab of the Stub Editor) that are defined for each operation within a stub's properties, there can be only one pass-through action for each operation. (For instructions, refer to the table in Properties Tab.)</p>
Customize a stub's execution to use the sift-and-pass-through capability temporarily	Rational Integration Tester 8.0.1 (or later)	<p>Instructions:</p> <ol style="list-style-type: none">1. Right-click the stub on the Test Factory perspective.2. Click Run... on the shortcut menu. The Customized Execution dialog box is displayed.3. Modify the pass-through actions (for information about this method, refer to table in Properties Tab).4. Click Run.

Method	Application	Description
Modify a stub's properties to use the sift-and-pass-through capability before starting the stub	Rational Test Control Panel 8.0.1 (or later)	For instructions, refer to Rational Test Control Panel Method .

Creating & Modifying Database Stubs

Contents

Introduction

Before Creating Database Stubs

Creating Database Stubs

Modifying Database Stubs

This chapter describes how to create and maintain database stubs.

For information about recording SQL events but not creating database stubs, refer to *IBM Rational Integration Tester Reference Guide*.

4.1 Introduction

The following sections provide an introduction to database stubs.

4.1.1 Purpose

If you want to create a test or suite of tests for an application that uses a database, you will need to be able to run the test (suite) in a repeatable fashion against a known set of database contents. Therefore, you will need to stub the database to obtain repeatable conditions.

4.1.2 Benefits

Database stubs enable you to execute tests against some parts of a system under test without affecting a live database.

4.1.3 Key Concepts

The following sections describe key database stubbing concepts in Rational Test Virtualization Server.

4.1.3.1 Persistence

Database stubs created in Rational Test Virtualization Server can be non-persistent or persistent:

- A non-persistent database stub will start from the same known state each time it is run. That is, any changes made to the contents of a database stub during its use will be lost when the stub is stopped. This enables tests to run against a known starting state.
- A persistent database stub will remember its state when it is stopped, overwriting the previous saved state. When it is restarted, the stub will have the same data that it had when it was last stopped.

It is possible to change a stub from being persistent to being non-persistent. This is useful if you want a stub to be persistent while you design, build, edit, and refine it for a test case; and then non-persistent, and thus in a known start state, each time that it is used.

The following sections describe how to use Rational Test Virtualization Server to create, run, and publish database stubs. (For information about recording SQL but not creating or using database stubs, refer to *IBM Rational Integration Tester Reference Guide*.)

4.1.3.2 Rational Integration Tester JDBC Proxy Modes

The Rational Integration Tester JDBC proxy enables Rational Integration Tester and Rational Test Virtualization Server to:

- Record SQL executed against databases from applications that use JDBC.
- Create and edit database stubs (Rational Test Virtualization Server only).

Database stubs contain subsets of data from a “live” (production) database. The contents of the stubs are built by analyzing an application’s use of SQL against the live database.

- Start a database stub (Rational Test Virtualization Server only).

Starting a stub loads the stub data into a simulation database and transparently redirects the application to that simulation database.

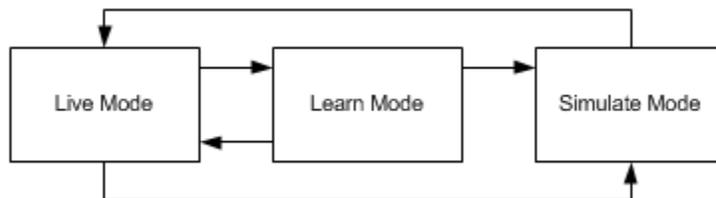
Therefore, users can test JDBC applications in a more deterministic manner.

When creating database stubs, the following modes of the Rational Integration Tester JDBC proxy are relevant:

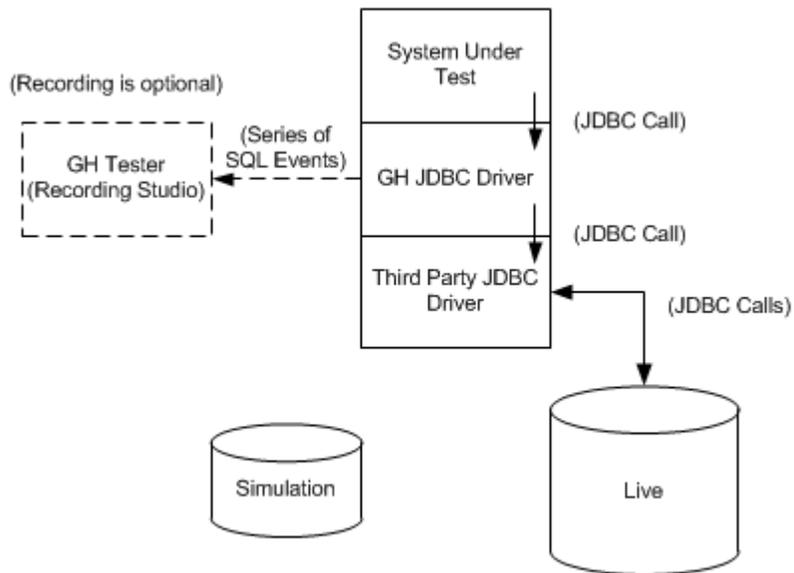
- Live
- Learn
- Simulate

The purpose of learn mode is to fill the simulation database, which facilitates creation of database stubs. Creating a stub then starting it moves learn mode to simulate mode, and stopping the stub moves simulate mode to live mode. When in live mode, it is possible to enter learn mode again.

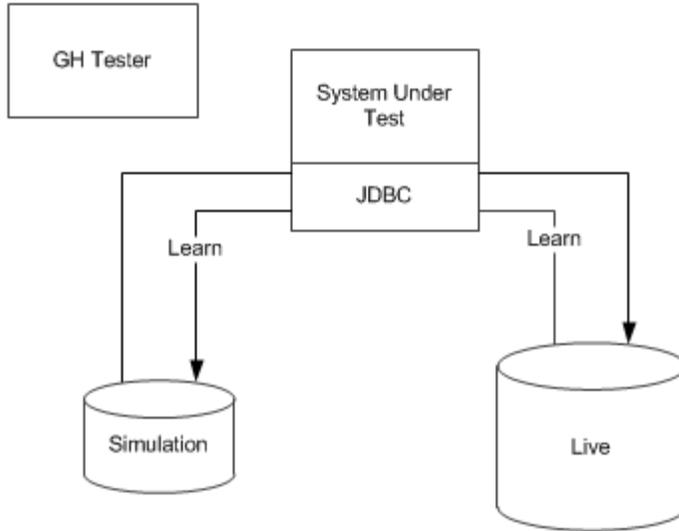
The following diagram illustrates the lifecycle for these modes.



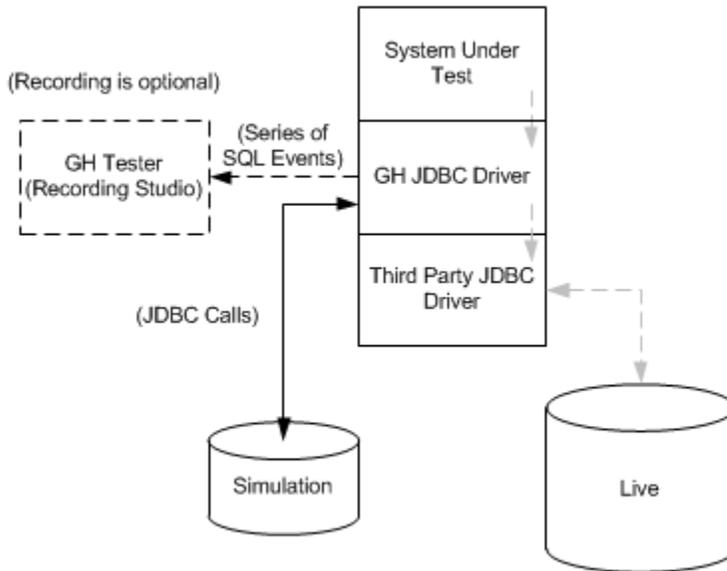
The following diagram illustrates live mode.



The following diagram illustrates learn mode.



The following diagram illustrates simulate mode.



The following sections describe how to set up, create, start, stop, and modify database stubs.

4.1.3.3 Stored Procedures

In Rational Integration Tester 8.0.1 (or later), you can record and virtualize any stored procedures.

With many applications, information persisted within a database is accessed directly through tables and views, and it is also accessed and modified through stored procedures. As with any service, the logic embedded within the stored procedure might not yet be available, or it might be slow to run, or it might even have costly side effects when the procedure runs.

Because of these differing behaviours, it is important to be able to record stored procedure executions to observe the interactions and then construct stubs to mimic existing behaviour or introduce new or even erroneous interactions.

One of the ways that a database stub can do that is to reply parrot-fashion (that is, repeating without understanding) to the stored procedure calls made by any application that is interacting with the physical database. For more sophisticated responses, Rational Integration Tester can also match against input parameters to determine which results should be provided. Further, it is possible to modify the signatures of stored procedures or introduce entirely new ones to aid development or integration testing.

While learning a stored procedure that returns a result set to the application, Rational Integration Tester learns the result set one row at a time.

4.1.3.4 Data Type Support

To facilitate the recording of stored procedures, Rational Integration Tester 8.0.1 (or later) supports most built-in data types in the database management systems that are supported by Rational Integration Tester. (For information about the database management systems supported by Rational Integration Tester, refer to *IBM Rational Integration Tester Installation Guide*.)

NOTE: Rational Integration Tester does not support user-defined data types.

Rational Integration Tester provides two methods for editing data types:

- You can use a spreadsheet program, such as Microsoft Excel, to edit the data. (Prerequisite: You must use the **Applications** page on the Preferences dialog box to link Rational Integration Tester with the spreadsheet program. For more information about this, refer to *IBM Rational Integration Tester Reference Guide*.)
- Alternatively, you can use the Database Stub Editor's Edit External wizard to load a database stub into a simulation database and then use the database management system's tools to edit the data. (For more information about the Edit External

wizard, refer to [External Edit Wizard](#).)

NOTE: If you want to edit large blocks of text, you must use the second method.

4.2 Before Creating Database Stubs

Before creating any database stubs, you must complete any required software installation tasks and you must set up the schema that will be used for stubbing a physical database.

The following sections describe how to complete these prerequisites.

4.2.1 Preparation and Planning

The remainder of this chapter assumes that you have installed and configured the required software.

For information about installing and configuring the required software, and determining which live databases are to be stubbed and whether there are any database schema or data source requirements, refer to *IBM Rational Integration Tester Platform Pack Installation Guide*.

NOTE: The version number of the Rational Integration Tester Platform Pack software that you are using must be the same as the version number of the Rational Integration Tester software that you are using. Therefore, if you have upgraded your Rational Integration Tester installation and you want to record SQL or create database stubs, you must also upgrade your Rational Integration Tester Platform Pack installation and upgrade the JDBC proxy wherever it is deployed.

4.2.2 Setting Up the Schema Used for Stubbing a Physical Database

Before you can create database stubs, you must specify in Rational Integration Tester which database schema is to be used for the simulation database.

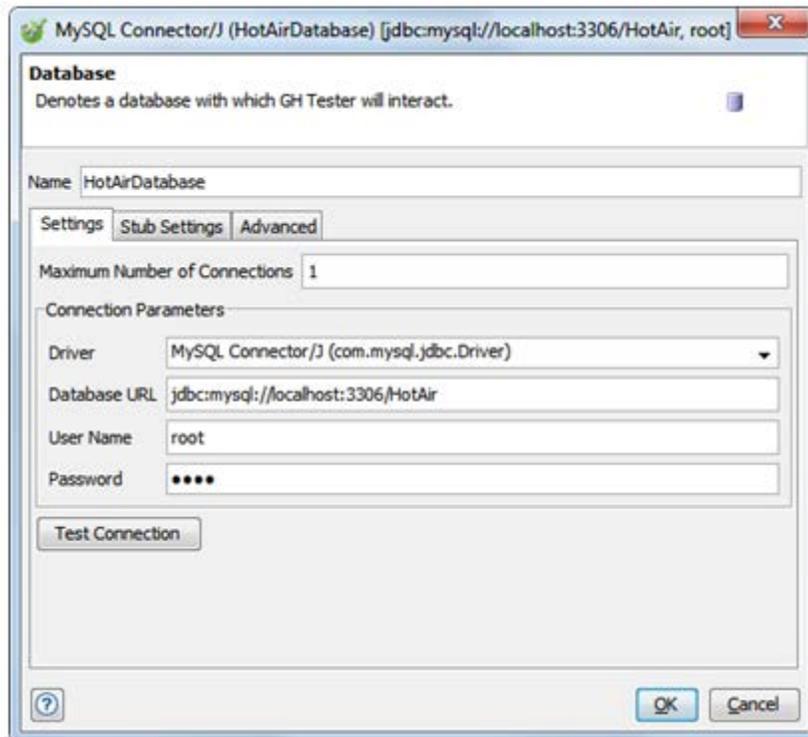
This schema will be used while Rational Integration Tester is “learning” the stubs and it will also be used when a stub is started, that is, the database schema that will be used to store the contents of a database stub.

NOTE: If you want to run more than database stub simultaneously, you will need a database stub schema for each database stub. For example, if you want to run five database stubs simultaneously, you will need five database stub schemas. However, each schema must match the corresponding database. For example, if you want to stub an Oracle database, you will need an Oracle schema. Alternatively, if you want to stub a Microsoft SQL Server database, you will need a Microsoft SQL Server schema, and so on.

To set up the schema that will be used for stubbing a specific physical database:

1. In Rational Integration Tester's Architecture School perspective's Physical View, double-click the physical database that you want to stub.

The Database (Properties) dialog box is displayed.



2. Click the **Stub Settings** tab.

The **Stub Settings** tab is displayed.

3. In the **Database URL** field, enter the URL of the server that will host the database stub schema.
4. In the **User Name** field, enter the name of the relevant database user.
5. In the **Password** field, enter the database user's password.
6. In the **Database Schema** field, enter the name of the database stub schema.

NOTE: Ideally, the database URL and schema used must not overlap with those of the live database because the live database will have tables

that have the same names. However, taking into account the database type, the combination of database URL, schema, and user name may provide sufficient separation between the live database and the database stub.

NOTE: The user name, password, and schema name should be provided by a database administrator (DBA). The database user must also have sufficient privileges to be able to create and delete tables in the simulation database.

7. **Optional:** Modify the default value of the **Max Row Count** field.

The default value of the field is 10,000. You can decrease this default value if you want to limit the maximum number of rows copied when learning from a live database or when copying rows into a database stub.

8. **Optional:** Modify the default value of the **Max Stored Procedure Calls** field. (This field is available only in Rational Integration Tester 8.0.1 (or later).)

The default value of the field is 10,000. If the system under test is very active, you can decrease the value of this field, thus limiting the quantity of data captured by Rational Integration Tester while learning.

9. **Optional:** Modify the default value of the **Max Result Set Row Count** field. (This field is available only in Rational Integration Tester 8.0.1 (or later).)

The default value of the field is 10,000. You can decrease this default value if you want to limit the maximum number of rows copied when learning a result set returned from a stored procedure and thus limit the amount of information that Rational Integration Tester will learn while creating a new database stub. You might want to use this field if you cannot or do not want to modify the behaviour of the system under test.

10. **Optional:** Click the **Unique** option button (default) if you want to record only unique invocations of a stored procedure based on the input parameters.

Alternatively, click the **All** option button if you want to record all invocations of a stored procedure. (These option buttons are available only in Rational Integration Tester 8.0.1 (or later).)

11. Click **Test Stub Connection** to verify Rational Integration Tester's connection to the database stub.

12. Click **OK**.

You are now ready to create database stubs.

4.3 Creating Database Stubs

Rational Test Virtualization Server provides users with two methods of creating a database stub:

- Use Rational Integration Tester's Recording Studio.
- Change the Rational Integration Tester JDBC proxy's modes manually.

The following sections describe how to use each of these methods.

4.3.1 Recording Studio Method

Under this method, Rational Integration Tester's Recording Studio perspective is used to create a database stub while SQL events are being recorded from a database.

To create a database stub, you must record SQL statements from a live application to populate the stub with data from the live database. Each SQL `SELECT` statement that is recorded will be analyzed by Rational Integration Tester and the corresponding results from the live system will be copied into the database stub that is being "learned" during recording. Rational Integration Tester attempts to copy the data matching the SQL `WHERE` clause.

For example, if the recording contains only the reading of one customer, the database stub will contain only data about that customer. Alternatively, if the recording contains the reading of, say, all customers with first name "John", the database stub will contain the same data.

NOTE: If there is no primary key on a database table that is being "learned", the simulation database is will probably contain duplicate records when learning is complete. However, Rational Integration Tester will display error messages if it detects any table that do not have any primary keys.

To create a database stub from recorded SQL events:

1. In Rational Integration Tester's Recording Studio perspective, select (for recording) the database that you want to stub.

NOTE: The database selected must be the database that was specified in [Before Creating Database Stubs](#).

2. Click the **Record** button (●) on the Events View toolbar.

The Create Stubbed Database Whilst Recording? dialog box is displayed.



3. Click the **Yes, record the SQL and create a virtual database** option button.
4. Click the **Create a database stub and start it** option button or the **Create a database stub but don't start it** option button.
5. In the **Stub name** field, enter a name for the stub.



-
6. Click **Start Recording**.
 7. Run a test case or a test case suite against the live system to populate the stub with sufficient data to make it useful.

You will be able to add more data to the stub later. During recording, Recording Studio will display information relating to stub creation, such as messages about database table creation. However, the stub will not be created **until** you stop recording.

8. On the Create Stubbed Database Whilst Recording? dialog box, click **Stop Recording**.

If you clicked the **Create a database stub and start it** option button or the **Create a database stub but don't start it** option button, the stub is now created.

If you clicked the **Keep learning** option button, you will have to execute another (or perhaps many more) stub recording session(s) to generate sufficient SQL to create the stub.

In Rational Integration Tester's Test Factory perspective, the newly created database stub is displayed under the relevant logical resource on the component tree.

You can now start, stop, or modify the new database stub. (These operations are described elsewhere in this chapter.)

4.3.2 Proxy Mode Change Method

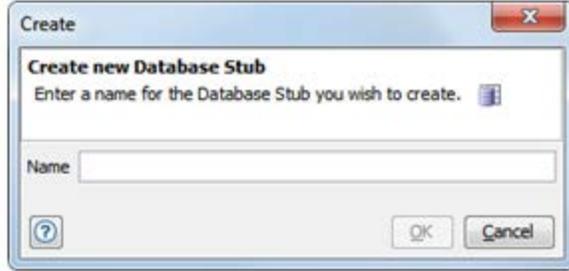
Under this method, users manage the live/learn/simulate lifecycle of the driver manually. (For information about this lifecycle, refer to [Rational Integration Tester JDBC Proxy Modes](#).)

To create and use a database stub by changing the Rational Integration Tester JDBC proxy's modes manually:

1. In Rational Integration Tester's Test Factory perspective, right-click the database that you want to stub and click **New > Stubs > Database Stub** on the shortcut menus.

NOTE: The database selected must be the database that was specified in [Before Creating Database Stubs](#).

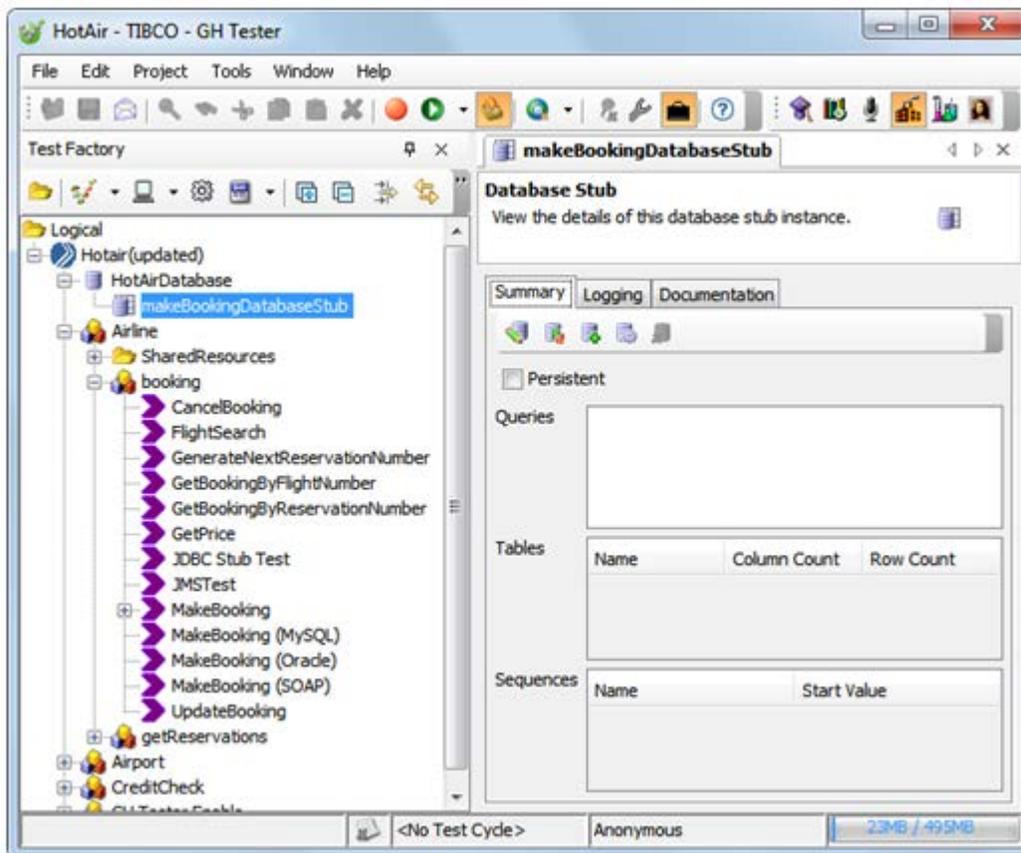
The Create a New Database Stub dialog box is displayed.



2. In the **Name** field, enter a name for the database stub.
3. Click **OK**.

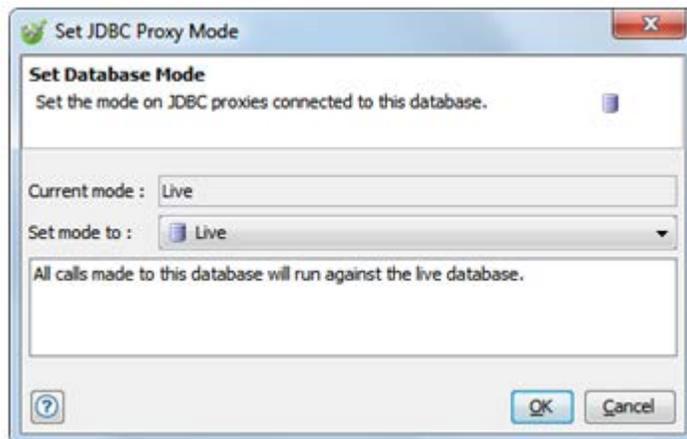
The stub is displayed under the relevant logical resource on the Test Factory perspective's component tree.

Selecting the empty stub opens it in the Database Stub Editor.

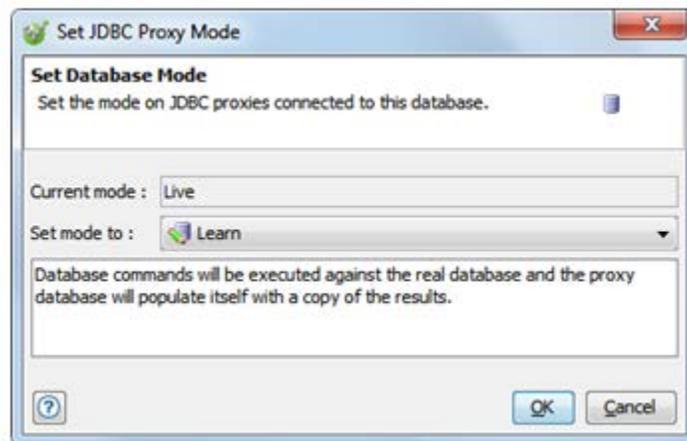


-
- In Rational Integration Tester's Architecture School perspective's Logical View, right-click the logical database and click **Set Proxy Mode** on the shortcut menu.

The Set JDBC Proxy Mode dialog box is displayed.



- In the **Set mode to** list, click **Learn**.



- Click **OK**.
- Populate the simulation database with live database data by using any of the following methods:
 - Run any test cases created for the system under test.
 - Use the system under test.
 - Wait for the system under test to populate the live database with query results.

8. After a period of time has elapsed (the period length will depend on the method chosen to populate the simulation database), open the Architecture School perspective's Logical View.

9. Right-click the database that is being stubbed and click **Set Proxy Mode** on the shortcut menu.

The Set JDBC Proxy Mode dialog box is displayed.

10. In the **Set mode to** list, click **Live**.

Rational Integration Tester will stop "learning" SQL and a stub has been created.

11. Click **OK**.

In Rational Integration Tester's Test Factory perspective, the newly created database stub is displayed under the relevant logical resource on the component tree.

You can now start, stop, or modify the new database stub.

The following sections describe these operations.

4.4 Modifying Database Stubs

When you open a database stub in Rational Integration Tester's Test Factory perspective, a Database Stub Editor screen is displayed. The following table summarizes the tabs on this screen.

Tab	Functionality
Summary	<p>This tab contains global information about the currently selected database stub.</p> <p>The tab contains the following:</p> <ul style="list-style-type: none">• An Editing Options toolbar, which is context sensitive (for information about this, refer to Editing Options Toolbar).• A Persistent check box for specifying whether a database stub is persistent or non-persistent (for information about this, refer to Persistence).• Option buttons for controlling default invocation matching and error behaviour for any stored procedures in the currently selected database stub.• Summary information about the physical and simulation databases associated with the currently selected database stub.
Queries	<p>This tab lists any SQL queries that were recorded during learn mode.</p> <p>The tab includes an Editing Options toolbar, which is context sensitive (for information about this, refer to Editing Options Toolbar). You can also access these Editing Options by right-clicking any part of the window on the tab.</p>
Tables	<p>This tab displays any database tables contained in the stub. For each table, the table name and the row and column counts are displayed.</p> <p>The tab includes an Editing Options toolbar, which is context sensitive (for information about this, refer to Editing Options Toolbar).</p>

Tab	Functionality
Stored Procedures	<p>This tab enables you to:</p> <ul style="list-style-type: none"> • Add new stored procedures. • Add new invocations to an existing stored procedure. • Modify invocation matching and error behaviour of a stored procedure. <p>The tab contains the following:</p> <ul style="list-style-type: none"> • An Editing Options toolbar, which is context sensitive (for information about this, refer to Editing Options Toolbar). • A Procedure list for selecting a stored procedure in the currently selected database stub. • An Edit button to enable you to use an external application (for example, Microsoft Excel) to create or edit a stored procedure. • Option buttons for specifying matched and unmatched invocation behaviours. • An Invocations toolbar that enables you to add, reorder, and delete invocations for the currently selected stored procedure. • An Invocations table that displays the input parameters to the currently selected stored procedure and a summary of the outputs from the stored procedure.
Sequences	<p>This tab displays any sequences contained in the currently selected database stub. For each sequence, the name and start value are displayed.</p> <p>The tab includes an Editing Options toolbar, which is context sensitive (for information about this, refer to Editing Options Toolbar).</p>
Logging	<p>This tab enables you to define how much information is recorded while the currently selected database stub is being executed. Any information recorded will be logged in Rational Integration Tester's project results database and on the Test Factory perspective's Console window.</p> <p>Options are as follows:</p> <ul style="list-style-type: none"> • None (no information is logged). • Normal (standard information is logged). • Debug (verbose information is logged)

Tab	Functionality
Documentation	<p>This tab enables you to enter extra information about the currently selected database stub and some of this extra information will be displayed in Rational Test Control Panel's Start Stub dialog box (for information about this dialog box, refer to Starting Stubs).</p> <p>Although you do not have to enter such information, it may help future users of your stub and it may also help you to identify in Rational Test Control Panel which stub you want to start if you have created several for the same database.</p> <p>For more information about using this tab, which is a standard tab on many dialog boxes in Rational Integration Tester, refer to <i>IBM Rational Integration Tester Reference Guide</i>.</p>

The following sections describe how to use **Summary** tab.

4.4.1 Editing Options Toolbar

In Rational Integration Tester 8.0.1 (or later), the Editing Options toolbar is context-sensitive, so some of the buttons on the toolbar operate differently on different tabs.

The following table describes how to use the toolbar on the tabs.

NOTE: The Editing Options toolbar is not displayed on the **Logging** and **Documentation** tabs. On the **Queries**, **Tables**, and **Sequences** tabs, you also can access the Editing Options shortcut menu by right-clicking any data on the tab.

Clicking this button...	On this tab...	Opens this...
 (Edit)	Summary, Stored Procedures	<p>The first screen of the Edit Database Stub wizard for the currently selected database stub.</p> <p>For more information, refer to Edit Database Stub Wizard.</p>
	Queries, Tables, Sequences	<p>The second screen of Edit Database Stub wizard for the currently selected database stub.</p> <p>For more information, refer to Edit Database Stub Wizard.</p>

Clicking this button...	On this tab...	Opens this...
 (Update)	Summary, Stored Procedures	The first screen of the Refresh wizard for the currently selected database stub. For more information, refer to Refresh Wizard .
	Queries, Tables	The second screen of the Refresh wizard for the currently selected database stub. For more information, refer to Refresh Wizard .
	Sequences	The alternate second screen (for sequences) of the Refresh wizard for the currently selected database stub. For more information, refer to Refresh Wizard .
 (Add Content)	Summary	The Add wizard for the currently selected database stub with options for adding to the stub by using queries, tables, stored procedures, or sequences. For more information, refer to Add Wizard
	Queries	The second screen of the Add wizard (for queries) for the currently selected database stub. For more information, refer to Add Wizard .
	Tables	The second screen of the Add wizard (for tables) for the currently selected database stub. For more information, refer to Add Wizard .
	Stored Procedures	The second screen of the Add wizard (for stored procedures) for the currently selected database stub. For more information, refer to Add Wizard .
	Sequences	The second screen of the Add wizard (for sequences) for the currently selected database stub. For more information, refer to Add Wizard .
 (External Edit)	Summary, Queries, Tables, Stored Procedures, Sequences	The External Edit wizard. For more information, refer to External Edit Wizard .

Clicking this button...	On this tab...	Opens this...
 (Delete)	Summary	(You cannot select anything on the Summary tab, so the Delete button on the tab is always unavailable.)
	Queries	Deletes the currently selected SQL queries from the currently selected database stub.
	Tables	Deletes the currently selected tables from the currently selected database stub.
	Stored Procedures	Deletes the currently selected stored procedure from the currently selected database stub.
	Sequences	Deletes the currently selected sequences from the currently selected database stub.

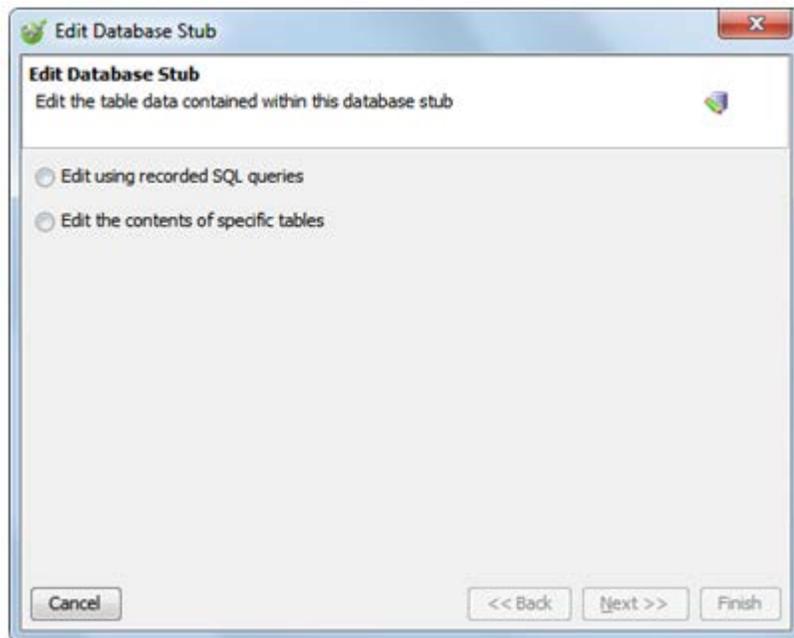
4.4.2 Wizards

The following sections describe how to use the Database Stub Editor wizards to edit a database stub.

4.4.2.1 Edit Database Stub Wizard

The Edit Database Stub wizard enables you to edit the contents of the currently selected database stub.

Opening the Edit Database Stub wizard from the Editing Options toolbar on the **Summary** or **Stored Procedures** tab opens the first screen of the Edit Database Stub wizard, which enables you to choose the set of SQL queries or tables that you want to use to edit the currently selected database stub.



SQL Queries

If you click the **Edit using recorded SQL queries** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to select the queries that you want to edit.

Alternatively, opening the Edit Database Stub wizard from the Editing Options toolbar on the **Queries** tab opens the second screen of the Edit Database Stub wizard.

After you have chosen the set of queries that you want to edit, clicking **Next** opens a spreadsheet containing those queries.

You can edit any value that you select in the spreadsheet. You can also add columns to the tables by entering a new column name in the header cell along with the database-type to be used for the column, and then entering values for each row. (The type name should be in parentheses. For example: `NEW_COL (VARCHAR(20))`.)

After saving your changes and closing the spreadsheet, the SQL query data displayed on the **Queries** tab is updated.

Tables

If you click the **Edit contents of specific tables** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to select the tables that you want to edit.

Alternatively, opening the Edit Database Stub wizard from the Editing Options toolbar on the **Tables** tab opens the second screen of the Edit Database Stub wizard.

After you have chosen the tables that you want to edit, a spreadsheet containing those tables is displayed. You can edit any value that you select in the spreadsheet.

After saving your changes and closing the spreadsheet, the tables data displayed on the **Tables** tab is updated.

4.4.2.2 Refresh Wizard

The Refresh wizard enables you to import additional data from a live database to the currently selected stub.

Opening the Refresh wizard from the Editing Options toolbar on the **Summary** or **Stored Procedures** tab opens the first screen of the Refresh wizard, which enables you to specify whether you want to learn by using a recorded SQL query, the contents of specific tables, or specific sequences.

SQL Queries

If you click the **Refresh using recorded SQL queries** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to select the queries that you want to use to update the currently selected database stub.

Alternatively, opening the Refresh wizard from the Editing Options toolbar on the **Queries** tab opens the second screen of the Refresh wizard.

After you have chosen the queries that you want to use (select the relevant check boxes), clicking **Next** updates the currently selected database stub with data based on the selected queries from the live database.

NOTE: To ensure that the currently selected database stub does not capture any duplicate values, the **Clear matching data before copying** check box is selected by default.

Click **Finish** to close the Refresh wizard.

Tables

If you click the **Refresh the contents of specific tables** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to select the tables in the live database that you want to use to update the currently selected database stub.

Alternatively, opening the Refresh wizard from the Editing Options toolbar on the **Tables** tab opens the second screen of the Refresh wizard.

After you have chosen the tables that you want to use (select the relevant check boxes), clicking **Next** updates the currently selected database stub with data from selected tables in the live database.

NOTE: To ensure that the currently selected database stub does not capture any duplicate values, the **Clear matching data before copying** check box is selected by default.

Click **Finish** to close the Refresh wizard.

Sequences

If you click the **Refresh sequences** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to select the sequences in the live database that you want to use to update the currently selected database stub.

Alternatively, opening the Refresh wizard from the Editing Options toolbar on the **Sequences** tab opens the second screen of the Refresh wizard.

After you have chosen the sequences that you want to use (select the relevant check boxes), clicking **Next** updates the currently selected database stub with data from selected sequences in the live database.

Click **Finish** to close the Refresh wizard.

4.4.2.3 Add Wizard

The Add wizard enables you to add more queries, tables, stored procedures, or sequences to the currently selected database stub.

Opening the Add wizard from the Editing Options toolbar on the **Summary** tab opens the first screen of the Add wizard.

SQL Queries

If you click the **Query** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to enter the query that you want to add to the currently selected database stub.

Alternatively, opening the Add wizard from the Editing Options toolbar on the **Queries** tab opens the second screen of the Add wizard.

After entering the query in SQL, clicking **Next** adds the query/queries to the currently selected database stub.

NOTE: If you select the **Use the new query to learn from the live database** check box before clicking **Next**, the currently selected query is used to learn from the live database, thus running a refresh for the query. If you select the **Clear matching data before learning from the database** check box before clicking **Next**, the currently selected database stub will not capture any duplicate values whenever the currently selected query is run.

Click **Finish** to close the Add wizard.

Tables

If you click the **Table** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to select the table in the live database that you want to add to the currently selected database stub.

Alternatively, opening the Add wizard from the Editing Options toolbar on the **Tables** tab opens the second screen of the Add wizard.

After selecting the live database table that you want to add, click **Next**.

NOTE: If you select the **Copy row from the live database** check box before clicking **Next**, the data as well as columns from the selected live database table will be added to the currently selected stub. Otherwise, only the column definitions of the table are added to the stub.

Click **Finish** to close the Add wizard.

Stored Procedures

If you click the **Stored Procedure** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to enter a new stored procedure or to select a stored procedure in the live database that you want to add to the currently selected database stub.

Alternatively, opening the Add wizard from the Editing Options toolbar on the Stored Procedures tab opens the second screen of the Add wizard.

To add a new stored procedure to the currently selected stub:

1. In the **Name** field on the Add Stored Procedure screen of the Add wizard, enter the name of the new stored procedure.
2. **Optional:** Click the **plus** button on the Parameters toolbar to add one or more new input, input/output, or output parameters to the new stored procedure. For example, you could use an input parameter to help find a match.
3. Click **Next**.
4. Click **Finish** to close the Add wizard.

To add an existing stored procedure from the live database to the currently selected database stub:

1. Using the **Name** list, select the stored procedure that you want to add to the stub.
2. **Optional:** Click the **plus** button on the Parameters toolbar to add one or more new input, input/output, or output parameters to the new stored procedure.
3. Click **Next**.
4. Click **Finish** to close the Add wizard.

To define the invocation matching and error behaviour of the new stored procedure:

1. Using the **Procedure** list on the **Stored Procedures** tab, select the stored procedure that you want to modify.
2. Click **Edit**.

The Invocation Matching dialog box is displayed.

3. Use the arrow buttons to move specific or all parameters between the Ignore and Matched windows.
4. Click **OK**.

If you used Recording Studio to create the currently selected database stub, invocations may already be displayed on the Invocations window on the **Stored Procedures** tab. The Invocations window can be used to add, modify, and delete invocations.

To add an invocation to the new stored procedure:

1. Click the **plus** button on the Invocations toolbar.

A new invocation is displayed under **Outbound**.

2. Select the new invocation and click the Spreadsheet button on the Invocations toolbar. Alternatively, double-click the new invocation

The spreadsheet application (if any) associated with Rational Integration Tester is displayed. (For information about associating applications with Rational Integration Tester, refer to IBM Rational Integration Tester Reference Guide.)

3. In the spreadsheet, add an inbound parameters worksheet, an outbound parameter worksheet, and one or more result set worksheets (format of worksheet names: Result Set 1, Result Set 2, and so on). Each worksheet must have one or more columns.
4. Save and close the spreadsheet.
5. Click **Finish** to close the Edit Database Stub wizard.
6. **Optional:** Select the **Default response** check box if you want a specific invocation to provide the default response for the currently selected stored procedure.

NOTE: A stored procedure can have only one default response.

Before running the currently selected database stub with the new stored procedure, you need to use the option buttons on the **Stored Procedures** tab to determine matched and unmatched invocation behaviour.

The following table outlines how the option buttons operate.

If matched invocation behaviour is...	And unmatched invocation behaviour is...	Then...
First match	Raise exception	If the input parameters that the application uses match any of the invocations stored in the stub, the first matching invocation is returned. Alternatively, if there are no matching invocations in the stub, a SQL exception will be returned to the application.
	Return to the application	If the input parameters that the application uses match any of the invocations stored in the stub, the first matching invocation is returned. Alternatively, if there are no matching invocations in the stub, the stored procedure call will return to the application without raising a SQL exception.

If matched invocation behaviour is...	And unmatched invocation behaviour is...	Then...
Replay (that is, next match)	Raise exception	<p>For matched invocations (for example, where there is one (or more) invocation(s) within the stub that match), while the application continues to call the stub, those invocations will be returned in turn. After those invocations are exhausted, the call will be treated as an unmatched invocation.</p> <p>Alternatively, if there are no matching invocations in the stub, a SQL exception will be returned to the application.</p>
	Return to the application	<p>For matched invocations, if Replay is selected and repeated calls exhaust the set of invocations, the call is treated as an unmatched invocation.</p> <p>Alternatively, if there are no matching invocations in the stub, the stored procedure call will return to the application without raising a SQL exception.</p>

NOTE: The invocation behaviour option buttons on the **Summary** tab determine the default option buttons on the **Stored Procedures** tab.

Sequences

If you click the **Sequence** option button on the first screen of the wizard and click **Next**, the second screen of the wizard enables you to select the sequence in the live database that you want to add to the currently selected database stub.

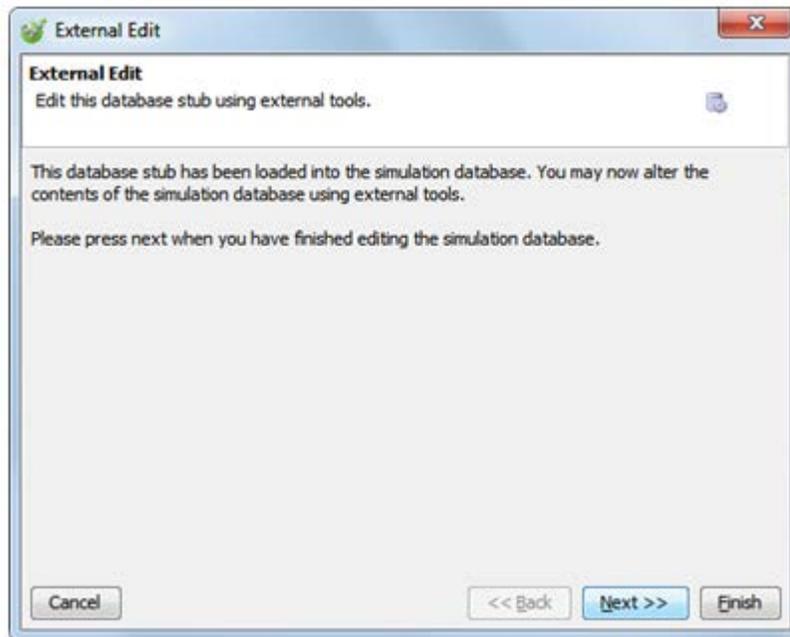
Alternatively, opening the Add wizard from the Editing Options toolbar on the **Sequences** tab opens the second screen of the Add wizard.

After selecting the live database sequence that you want to add, click **Next**.

Click **Finish** to close the Add wizard.

4.4.2.4 External Edit Wizard

The External Edit wizard enables you to use any database management systems tools (to which you have access) to edit the contents of the currently selected database stub.



Opening the External Edit wizard and clicking **Next** loads the currently selected database stub into the simulation database and Rational Integration Tester waits for you confirm that you have completed your edits.

After you have confirmed completion of your edits, the modified database stub is loaded from the simulation database back into Rational Integration Tester.

Click **Finish** to close the External Edit wizard.

Publishing & Running Stubs

Contents

[Publishing Stubs](#)

[Running Stubs](#)

This chapter describes how to publish and run stubs.

For a brief overview of Rational Test Control Panel, refer to [Using Rational Test Control Panel](#).

5.1 Publishing Stubs

Stubs that have been built inside Rational Integration Tester can be published to a Rational Test Control Panel instance, which facilitates deployment and control of stubs without requiring all users of those stubs to have access to Rational Integration Tester.

To publish a stub:

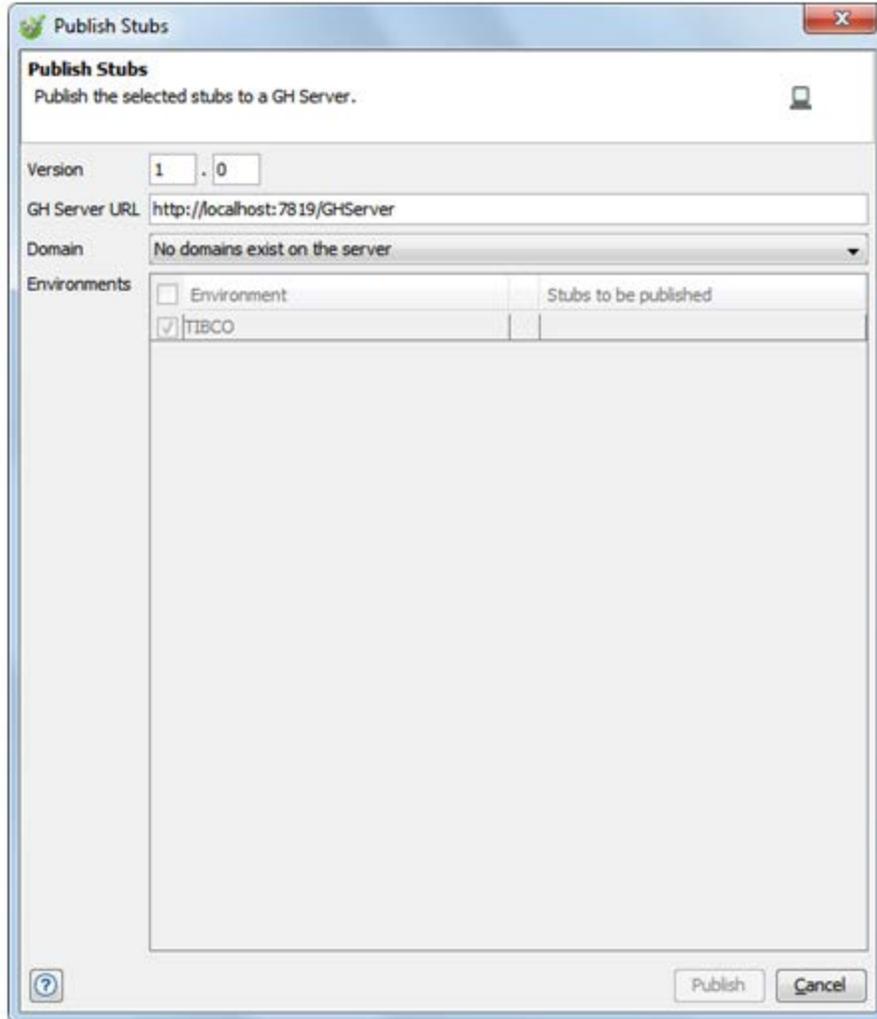
1. In Rational Integration Tester's Test Factory perspective, right-click the **Stubs** folder that contains the stub that you want to publish and click **Publish Stubs...** on the shortcut menu.

Alternatively, right-click the **operation or component** that contains the stub that you want to publish and click **Publish Stubs...** on the shortcut menu.

You cannot publish a single stub by right-clicking it. This is because a stub must be published from a container (that is, an operation, component, or the logical root of the tree) and not from stub itself. All stubs that are within the scope of the selected **Stubs** folder or operation or component or logical root (whichever is applicable) will be published.

NOTE: If the operation or component or logical root selected does not contain any stubs, an error message is displayed.

The Publish Stubs dialog box is displayed.



2. In the **Version** fields, enter a version number (an integer) for the stub or set of stubs if more than one was selected for publication.

NOTE: A stub's version number assists Rational Test Control Panel users when they are selecting the stubs that they want to run.

3. Edit (if necessary) the **Rational Test Control Panel URL** field to the URL of the Rational Test Control Panel instance to which you want to publish the selected stub(s).

NOTE: The default value in the **Rational Test Control Panel URL** field is determined by the URL specified on the **Server Settings** tab on

Rational Integration Tester's Project Settings dialog box, which is opened by clicking **Project > Project Settings** on the menu bar.

4. In the **Domain** list, click the domain where you want to publish the selected stub(s).

A domain represents a logical grouping of related systems that are part of a real business project and it is the basic unit of management within Rational Test Virtualization Server.

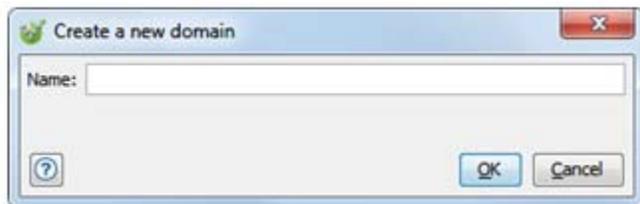
NOTE: The default value in the **Domain** list is determined by the default domain specified in the **Domain** list on the **Server Settings** tab on Rational Integration Tester's Project Settings dialog box.

NOTE: The **Domain** list will display error messages if a connection cannot be established to the specified Rational Test Control Panel instance. You will not be able to publish any stubs to the specified Rational Test Control Panel instance unless and until the connection to the instance is re-established.

Optional: To create a new domain:

- In the **Domain** list, click **Create new domain**.

The Create a new domain dialog box is displayed.

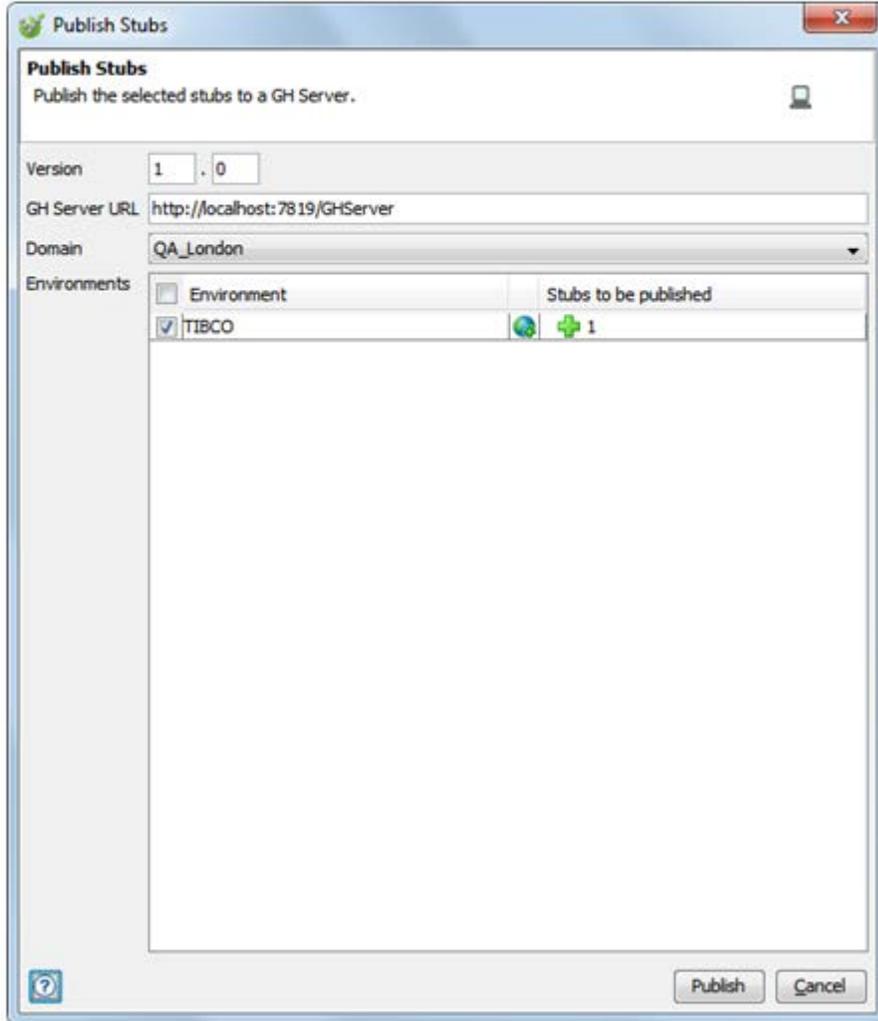


- Click **OK** on the Create a new domain dialog box to close it.

The new domain is displayed in the **Domain** list on the Publish Stubs dialog box.

NOTE: If you are a Rational Test Control Panel administrator, you can use Rational Test Control Panel to create and configure new domains. For information about this, refer to *IBM Rational Test Control Panel System Administration Guide*.

5. On the **Environments** window, select the check box(es) of the environment or environments where you want to publish the selected stub(s).



An environment enables Rational Integration Tester and Rational Test Virtualization Server users to define groups of variables or tags that can be used in both tests and transport definitions.

NOTE: By default, the check box of the currently selected environment in Rational Integration Tester is selected.

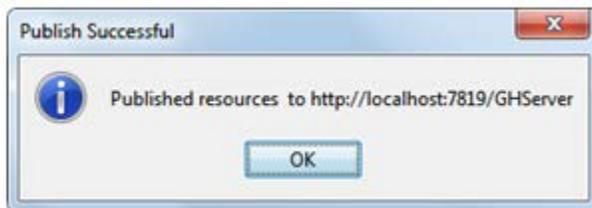
The following table describes the icons that can be displayed on the **Environments** window.

Icon	Description
	The specified environment will be created on the specified Rational Test Control Panel instance.
	The specified environment already exists on the specified Rational Test Control Panel instance.
 "N"	"N" number of new stubs will be published ("created") to the specified Rational Test Control Panel instance.
 "N"	"N" number of existing stubs will be republished ("updated") at the same version level to the specified Rational Test Control Panel instance.
 "N"	"N" number of new versions of existing stubs will be published ("replaced") to the specified Rational Test Control Panel instance.

6. Click **Publish**.

The selected stub(s) is (are) published to the specified Rational Test Control Panel instance.

A status message is displayed to confirm this.



7. Click **OK**.

5.1.1 Verifying Publication of Stubs

To verify that a stub (or collection of stubs) has (have) been published to a specific Rational Test Control Panel instance:

1. Log into the specified Rational Test Control Panel instance.

NOTE: For a brief overview of Rational Test Control Panel, refer to [Using Rational Test Control Panel](#).

2. Click the **VIE** icon or navigation link.
3. Click the specified domain and environment.



4. Click **View Dashboard**.

The selected stub(s) should be displayed on the VIE Dashboard page.



NOTE: For information about cancelling the deployment of a stub, refer to [Cancelling Startup of Single Stub](#).

5.2 Running Stubs

The following sections describe how to start and stop stubs.

5.2.1 Starting Stubs

Both Rational Test Control Panel and Rational Integration Tester can be used to start stubs. There are some limitations on stubs if they are started by Rational Integration Tester. The following table describes those limitations.

If a stub is published to Rational Test Control Panel and started by Rational Test Control Panel...	If a stub is started by Rational Integration Tester...
There are no running time limits on the stub.	It will run for only five minutes unless the associated test scenario requires it to run for a longer period.
There are no transactions per second (TPS) limits on the stub.	There is a limit of one TPS on the stub.

The following sections describe how to use Rational Test Control Panel and Rational Integration Tester to start stubs.

5.2.1.1 Rational Test Control Panel Method

Rational Test Control Panel's VIE Dashboard page enables you to control stubs that have been created and published from Rational Integration Tester, so you do not need to use (or to have access to) Rational Integration Tester to start them.

The VIE Dashboard page also enables you to:

- Override a stub's tag setting.
- Control a stub's behaviour configuration.
- Adjust a stub's response time.

In Rational Test Control Panel 8.0.1 (or later), you can also start scenarios, which are collections of configured stubs that can be started and stopped together.

The following sections describe how to use Rational Test Control Panel to start stubs and scenarios.

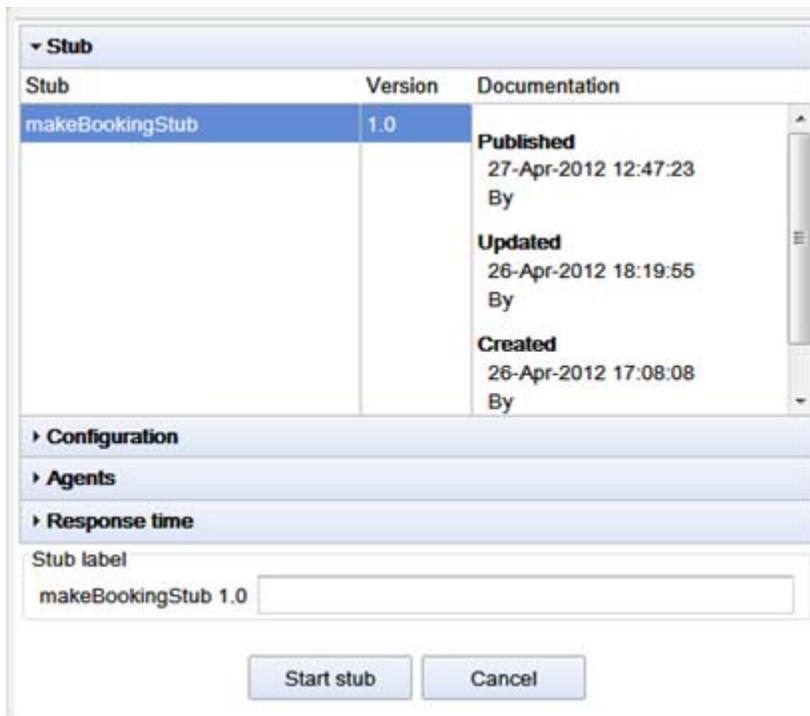
Starting Single Stubs

To start a single stub:

1. Log into Rational Test Control Panel.
Rational Test Control Panel's application window is displayed.
2. Click the **VIE** icon or navigation link.
The **VIE** page is displayed.
3. Click the relevant domain and environment.
4. Click **View Dashboard**.
The VIE Dashboard page is displayed.
5. Under the **Satisfied by** column, click the stub's **plus** button (+).

NOTE: If the **plus** button is unavailable, there are no available stubs for the selected operation.

The Start Stub dialog box is displayed on the VIE Dashboard page.



The following table describes the controls on the Start Stub dialog box.

Control	Description
Stub (page)	<p>This enables you to select the stub, including the stub version, that you want to start.</p> <p>For each stub listed on the page, the Documentation column displays any text entered for that stub on the Documentation tab of the Stub Editor in Rational Integration Tester.</p> <p>After selecting a stub, you can modify its properties by using the other pages on the dialog box.</p> <p>NOTE: For database stubs, only one stub for each database can be run at any given time.</p>
Configuration (page)	<p>This enables you to override the default input tag values (for each specified behaviour) that were published with the currently selected stub.</p> <p>Under the Input tags column, each behaviour specified for the stub is listed by name.</p> <p>Clicking a behaviour displays its default input tag values. If you want to override an input tag's default value, click the behaviour and then click the tag and enter a new value under Value. If you want to force the value of the tag to be null instead of being blank, select the Null? check box.</p> <p>Clicking Logging enables you to modify the currently selected stub's log level, which determines how much logging will be written to the project results database. By default, a stub uses the log level that was published with it.</p> <p>Log level options are as follows:</p> <ul style="list-style-type: none">• Default (as defined in the stub)• None• Normal• Debug
Agents (page)	<p>This enables you to select the agents where the currently selected sub will run after it is started.</p> <p>The available agents are displayed, including host name, status, software version number, and system statistics. To select an agent, select its check box.</p> <p>NOTE: If the currently selected stub is a database stub, it can run on only one agent at any given time.</p>

Control	Description
Response time (page)	<p>This enables you to adjust the currently selected stub's response time by entering (in milliseconds) a fixed delay or a delay distribution with minimum and maximum delay times.</p> <p>Delay distributions options are as follows:</p> <ul style="list-style-type: none"> • Fixed • Uniform • Gaussian <p>NOTE: After a stub has started, its response time can be modified while it is running (for information about this, refer to Changing Response Times).</p>
Pass Through	<p>Clicking an operation on the page displays the Action list. (For information about the options in this list, refer to Configuring Transports to Use the Sift-and-Pass-Through Capability.)</p>
Stub label (field)	<p>This is an optional field that enables you to enter a custom label for the currently selected stub so that you distinguish multiple running instances of the stub from each other.</p>

- After selecting a stub on the Start Stub dialog and making any desired changes to the stub, clicking **Start Stub** on the dialog box creates a new row under the currently selected operation on the VIE Dashboard page, and `Deploying...` is displayed under the **Status** column.

NOTE: For information about viewing stub error messages and stub logs, refer to [Troubleshooting](#).

Cancelling Startup of Single Stub

If a stub does not deploy reasonably quickly, or if an agent is not operating correctly, you can cancel the deployment of the stub.

To cancel the deployment of a stub:

- Log into Rational Test Control Panel.
Rational Test Control Panel's application window is displayed.
- Click the **VIE** icon or navigation link.
The **VIE** page is displayed.
- Click the relevant domain and environment and then click **View Dashboard**.
The VIE Dashboard page is displayed.

- Under the **Satisfied by** column, click the stub's **arrow** button (↘) and then click **Cancel deployment** on the shortcut menu.



The deployment of the selected stub is cancelled.

Starting Scenarios

In Rational Test Control Panel 8.0.1 (or later), you can create scenarios, which are collections of configured stubs that can be started and stopped together.

NOTE: In each environment in Rational Test Control Panel, only one scenario can be running at any one time.

Before you can create a scenario in Rational Test Control Panel:

- Optional:** Lock the environment that you are using to prevent other users from interrupting the creation of your scenario. (For information about this, refer to [Locking and Unlocking Environments](#).)

If you lock the currently selected environment, the locked status message is displayed on the upper half of the VIE Dashboard page.

- Configure and start each stub that will be included in the scenario. (For information about this, refer to [Starting Single Stubs](#).)
- For each stub started, wait until **Ready** is displayed under the **Status** column.

Otherwise, each stub that is not yet fully deployed will be excluded from the scenario that you want to create.

To create a scenario in Rational Test Control Panel:

1. Click the **arrow** button (▾) next to the **Scenario** icon in the upper-right corner of the VIE Dashboard page.
2. Click **Save new scenario** on the shortcut menu.

The Save new scenario dialog box is displayed.

3. In the **Scenario name** field, enter a unique name for the new scenario.
4. Click **Save scenario**.

If you have already created a scenario with the same name in the current environment, you will be given the option to overwrite it or cancel.

Otherwise, the new scenario is created and a confirmation message is displayed, or an error message is displayed if saving was not completed successfully.

5. Click **OK**.

NOTE: Any stubs included in the new scenario that are still displayed as running on the VIE Dashboard page are not marked as being part of the new scenario because they were not started by the scenario.

The scenario that you have just created is now available for any user to start.

To start a scenario in Rational Test Control Panel:

1. **Optional:** Lock the environment that you are using to prevent other users from interrupting your testing. (For information about this, refer to [new section] Locking and Unlocking Environments.)

If you lock the currently selected environment, the locked status message is displayed on the upper half of the VIE Dashboard page.

2. Click the **arrow** button (▾) next to the **Scenario** icon in the upper-right corner of the VIE Dashboard page.
3. Click **Start scenario** on the shortcut menu.

The Start scenario dialog box is displayed.

The drop-down list on the Start scenario dialog box displays each scenario created for the currently selected environment and the user name of the user who created it.

-
4. Using the drop-down list, select the scenario that you want to start.
 5. Click one of the option buttons outlined in the following table.

Option Button	Description
Do not stop any stubs (default)	Scenario will be started alongside any other stubs currently running in the currently selected environment.
Stop all stubs	Before it starts, the scenario will stop all other stubs currently running in the currently selected environment.
Stop stubs for operations used in this scenario	Before it starts, the scenario will stop all other stubs in the currently selected environment that are running under operations that are also satisfied by stubs that are included in the currently selected scenario.

6. Click **Start scenario**.

The scenario starts. For each stub started, **Deploying...** is displayed under the **Status** column. If any stubs are stopped because of the startup of the scenario, **Stopping** is displayed under the **Status** column for each stub affected.

NOTE: While a scenario is running, **Start scenario** on the Scenario shortcut menu (opened by clicking the **arrow** button ▾ next to the **Scenario** icon) is unavailable. However, you can use a scenario that is still running to create another scenario.

You can modify any stub in a scenario that is running and thus modify that scenario.

To modify a scenario that is currently running:

1. Stop the stub that you want to edit. (For information about stopping stubs, refer to [Stopping Stubs](#).)

NOTE: The scenario will still be displayed as running unless all of its stubs are stopped.

2. Modify the stub or start any new stubs that you want to include in the scenario. (For information about this, refer to [Starting Single Stubs](#).)
3. Under the **Satisfied by** column, click the scenario's **arrow** button (▾).
4. Click **Save changes** on the shortcut menu.

The Save new scenario dialog box is displayed.

-
5. Click **Save scenario**.

NOTE: If you modify the name of the scenario in the **Scenario name** field before clicking **Save scenario**, a new scenario will be created based on the changes to the scenario that you have modified but the changes to the original scenario will not be saved.

A confirmation prompt is displayed if you have made changes to an existing scenario or if you have entered a name for a new scenario (based on the changes to the scenario that you have modified) that is a duplicate of a scenario that already exists in the currently selected environment.

6. Click **OK**.

The changes to the modified scenario will take effect the next time the scenario is started.

A user can delete any scenarios that he or she has created by clicking the **arrow** button (↖) next to the **Scenario** icon in the upper-right corner of the VIE Dashboard page and clicking **Manage scenarios** on the shortcut menu.

NOTE: An administrator can delete any user's scenarios by using the **Scenarios** page on the **Domains and environments** tab on the Rational Test Control Panel Administration page.

5.2.1.2 Rational Integration Tester Method

Rational Integration Tester can be used to start stubs but Rational Test Control Panel provides more control over the startup of a stub or a set of stubs.

To start a stub:

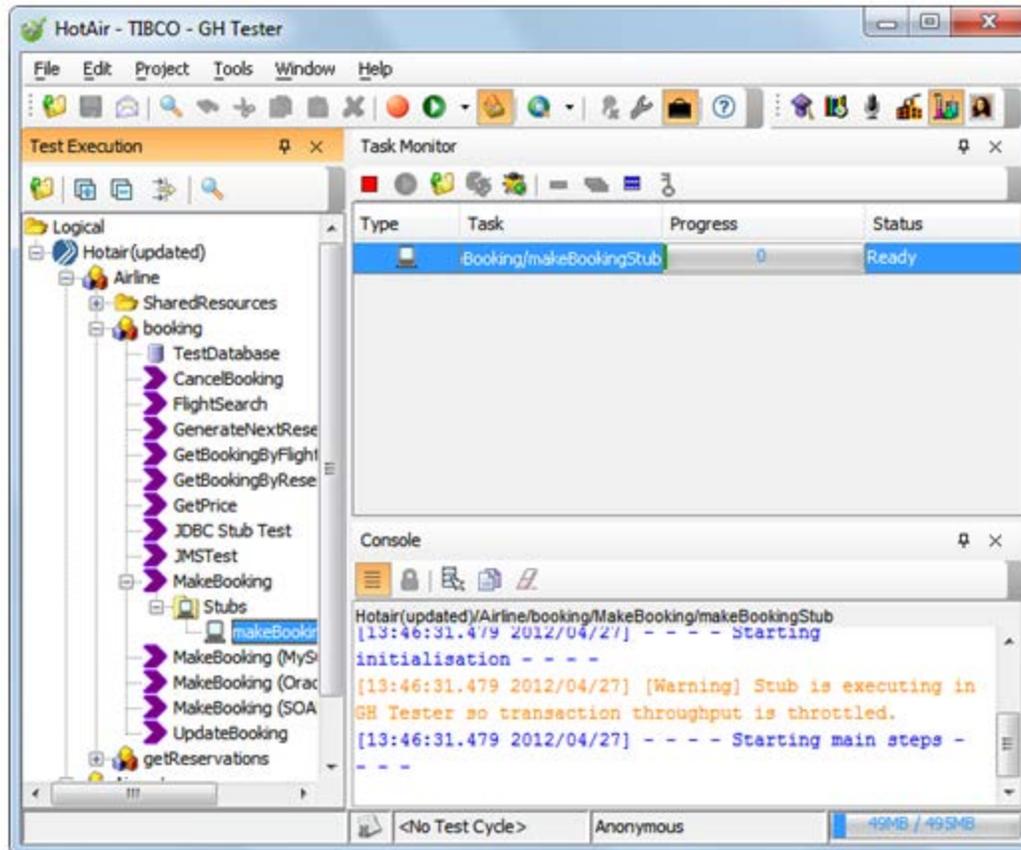
1. In Rational Integration Tester's Test Lab perspective, find the stub that you want to start.
2. Right-click the stub and click **Run** on the shortcut menu.

Alternatively, select the stub and click **Run** on the Task Monitor window's toolbar.

NOTE: If you want to run all stubs for a particular operation, right-click the operation's **Stubs** folder and then click **Run All > Stub** on the shortcut menu.

Starting a message-based stub causes the following to happen:

- The Test Lab perspective's Console window displays progress of the stub's startup and messages from the stub as it runs. If the stub's log level has been set to **Debug**, you will be able to view detailed information about what the stub is doing. (For information about setting a stub's log level, refer to [Rational Test Control Panel Method](#).)



- The stub will start listening on queues/ports.
- If proxies are being used, traffic will be routed to the stub automatically.
- If you are using Rational Test Control Panel 8.0.1 (or later), the Rational Test Control Panel Activity log and the stub's Console output will display the number of proxies affected by any recording rules specified.

NOTE: For information about restrictions on stubs started by Rational Integration Tester, refer to [Starting Stubs](#).

Starting a database stub causes the following to happen:

-
- The data contained in the stub is loaded into the simulation database.
 - The Rational Integration Tester JDBC proxy is switched to simulate mode, which affects the system under test but not Rational Integration Tester, and all interactions between the system under test and the live database are directed to the stub. (For information about simulate mode, refer to [Rational Integration Tester JDBC Proxy Modes](#).)
 - If you are using Rational Test Control Panel 8.0.1 (or later), the Rational Test Control Panel Activity log will record the number of proxies affected by any recording rules specified.
 - The Test Lab perspective's Console window displays progress of the stub's startup. If you are using Rational Test Control Panel 8.0.1 (or later), the Console window also displays the number of proxies affected by any recording rules specified.

NOTE: If you are using Rational Integration Tester 8.0.1 (or later), clicking default case console output or failed validation console output on the Console window opens the Message Differences window, which enables you to debug the stub. (For information about debugging a stub, refer to [Debugging Message-Based Stubs](#). For general information about the Message Differences window, refer to *IBM Rational Integration Tester Reference Guide*.)

5.2.2 Stopping Stubs

The following sections describe how to use Rational Test Control Panel and Rational Integration Tester to stop stubs.

5.2.2.1 Rational Test Control Panel Method

Rational Test Control Panel provides several different methods of stopping individual stubs and multiple stubs. The following table describes these methods.

To stop...	Do this...
A stub running on an agent (at agent level)	<p>If using Rational Test Control Panel's VIE Dashboard page:</p> <ol style="list-style-type: none">1. Click Show Agents (if necessary). A pop-up window is displayed for each agent that is running (in the selected domain/environment, if applicable).2. On the pop-up window of the agent in which you are interested, if the stub that you want to stop is displayed, click the Stop button (⏹) next to the stub's name. <p>Alternatively, if using Rational Test Control Panel's Agents page:</p> <ol style="list-style-type: none">1. Click the magnifying glass button (🔍) of an engine. The Agent deployed projects and stubs pop-up window is displayed.2. On the pop-up window of the agent in which you are interested, if the stub that you want to stop is displayed, click the Stop button (⏹) next to the stub's name.
A stub (on the VIE Dashboard page)	<ol style="list-style-type: none">1. Under the Satisfied by column, click the stub's arrow button (-).2. Click Stop stub on the shortcut menu.
All stubs at operation level (on the VIE Dashboard page)	<ol style="list-style-type: none">1. Under the Satisfied by column, click the operation's arrow button (-).2. Click Stop all stubs on the shortcut menu.
All stubs at component level (on the VIE Dashboard page)	<ol style="list-style-type: none">1. Under the Satisfied by column, click the component's arrow button (-).2. Click Stop all stubs on the shortcut menu.
All stubs for the selected domain/environment (on the VIE Dashboard page)	Click Stop All Stubs (next to Show Agents).

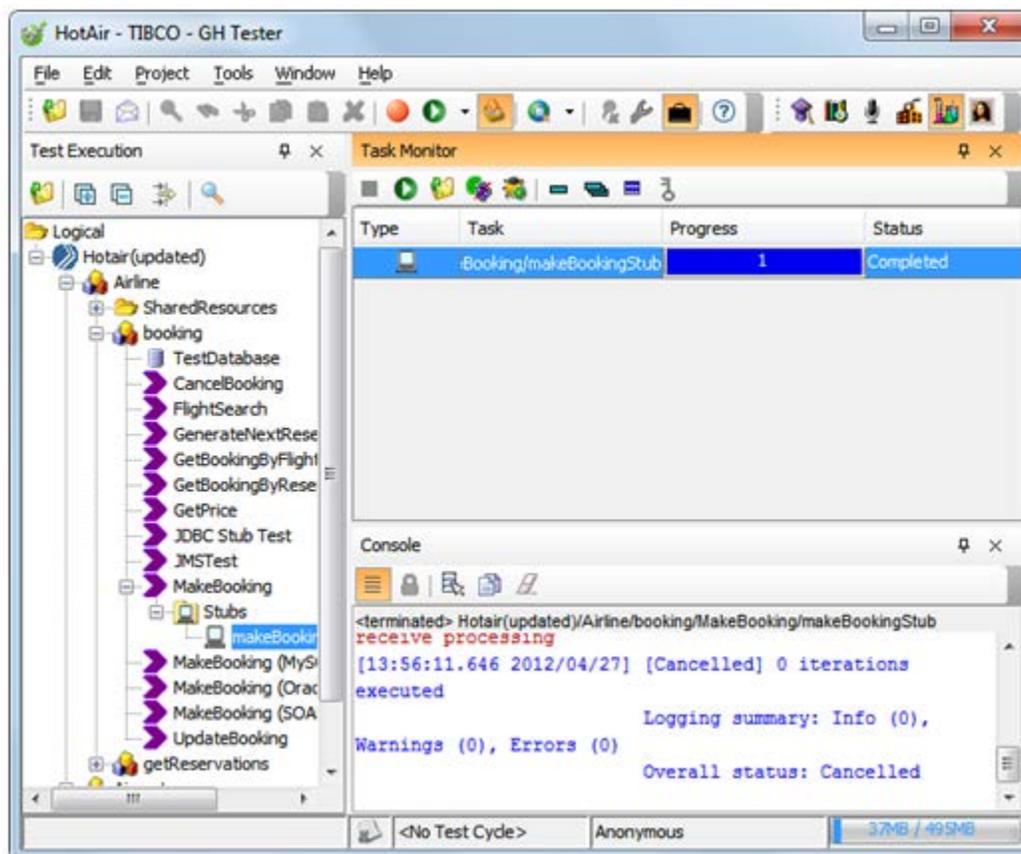
To stop...	Do this...
A scenario for the selected domain/environment (on the VIE Dashboard page)	<p>Option one:</p> <ol style="list-style-type: none"> 1. Under the Satisfied by column, click the scenario's arrow button (-). 2. Click Stop scenario on the shortcut menu. <p>Option two:</p> <ol style="list-style-type: none"> 1. Click the arrow button (-) next to the Scenario icon in the upper-right corner of the VIE Dashboard page. 2. Click Stop scenario on the shortcut menu. <p>Option three:</p> <p>Stop each stub that is included in the scenario as follows:</p> <ol style="list-style-type: none"> 1. Under the Satisfied by column, click the each stub's arrow button (-). 2. Click Stop stub on the shortcut menu. <p>Option four:</p> <p>Click the Stop all stubs button next to the Scenario icon in the upper-right corner of the VIE Dashboard page.</p> <p>NOTE: A scenario is displayed as running unless all of its stubs are stopped.</p>

5.2.2.2 Rational Integration Tester Method

To stop a stub:

1. In Rational Integration Tester's Test Lab perspective, find the stub that you want to stop.
2. Select the stub and click **Stop** on the Task Monitor window's toolbar.

Alternatively, select the stub and click **Stop** on the Task Monitor's toolbar.



Stopping a message-based stub causes the following to happen:

- If traffic was being routed automatically routed to the stub by means of a Rational Integration Tester Proxy, the proxy will be configured to resume sending messages to the live system.

Stopping a database stub causes the following to happen:

- Rational Integration Tester instructs each proxy to return to the state it was in before the stub was started.

-
- The simulation schema is cleared.

NOTE: Stopping a persistent stub updates the stub with a fresh snapshot of data. A non-persistent stub is not updated and, if started again, it will revert to its original state. If there is no connection to the database, exceptions are recorded in a log. In Rational Test Virtualization Server, stubs cannot be persistent.

5.2.3 Modifying Running Stubs

If you are using Rational Integration Tester to run a stub, you can modify the stub while it is running.

NOTE: If you are using Rational Integration Tester 8.0.1 (or later), clicking default case console output or failed validation console output on the Test Lab perspective's Console window opens the Message Differences window, which enables you to debug the stub. (For information about debugging a stub, refer to [Debugging Message-Based Stubs](#). For general information about the Message Differences window, refer to *IBM Rational Integration Tester Reference Guide*.)

To modify a running stub:

1. In Rational Integration Tester's Test Factory perspective, double-click the stub to open it in the Stub Editor.
2. Use the Stub Editor to make and save any desired changes to the stub. (For information about using the Stub Editor, refer to [Modifying Message-Based Stubs](#).)
3. Open the Test Lab perspective.

The Task Monitor window shows that the modified stub was stopped and restarted by Rational Integration Tester.

5.2.4 Controlling Running Stubs

After a stub is running, the following information is displayed on the VIE Dashboard page:

- Ready is displayed under the **Status** column. (If **Error** is displayed, refer to [Stopping Stubs](#).)
- The number of requests that the stub has handled is displayed under the **Handled** column.

-
- The number of requests that the stub has handled since the last metrics reset is displayed under the **Since reset** column.

Apart from stopping the stub (for information about this, refer to [Stopping Stubs](#)), you can also reset its metrics, change its response times, and cancel its deployment.

The following sections describe these actions.

5.2.4.1 Resetting Metrics

To reset the number of requests that a running stub has handled:

1. Log into Rational Test Control Panel.
Rational Test Control Panel's application window is displayed.
2. Click the **VIE** icon or navigation link.
The **VIE** page is displayed.
3. Click the relevant domain and environment and then click **View Dashboard**.
The VIE Dashboard page is displayed.
4. Under the **Satisfied by** column, click the stub's **arrow** button (↕) and then click **Reset metrics** on the shortcut menu.

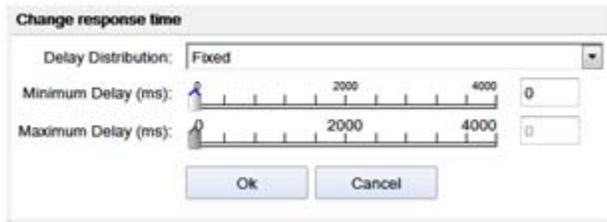
The value displayed under **Since reset** column for the selected stub is reset to 0.

5.2.4.2 Changing Response Times

To change the response time of a running stub:

1. Log into Rational Test Control Panel.
Rational Test Control Panel's application window is displayed.
2. Click the **VIE** icon or navigation link.
The **VIE** page is displayed.
3. Click the relevant domain and environment and then click **View Dashboard**.
The VIE Dashboard page is displayed.
4. Under the **Satisfied by** column, click the stub's **arrow** button (↕) and then click **Change response time...** on the shortcut menu.

A Change response time dialog box is displayed on the VIE Dashboard page. (For information about the fields on this dialog box, refer to [Starting Stubs](#).)



5. After making any desired changes, click **OK**.

5.2.5 Locking and Unlocking Environments

Locking a Rational Test Control Panel environment prevents other users from starting or stopping stubs in that environment.

After an environment is locked:

- The user who locked the environment can make any changes to the environment as normal.
- Only the user who locked the environment or a Rational Test Control Panel administrator can unlock it.
- Any other user can request that the environment be unlocked.

When locking an environment, there is the option to specify an expected duration for the lock, which will be displayed to any users who try to access the environment while it is locked. However, the lock will not expire if the specified duration is exceeded.

The following sections describe how to lock and unlock an environment.

5.2.5.1 Locking an Environment

To lock an environment:

1. Log into Rational Test Control Panel.

The Rational Test Control Panel application window is displayed.

2. Click the **VIE** icon or navigation link.

The VIE page is displayed.

3. Click the relevant domain and environment.

4. Click **View Dashboard**.

The VIE Dashboard page is displayed.

-
5. Click the **Lock** icon in the upper-right corner of the VIE Dashboard page.

The Lock environment dialog box is displayed.

The Domain and Environment fields display the names of the currently selected domain and environment.

6. **Optional:** In the **Expected duration** field, enter the lock's estimated maximum duration.

NOTE: This field does not set a maximum time limit for the lock.

7. **Optional:** Using the drop-down list next to the **Expected duration** field, specify whether the duration will be measured in minutes, days, weeks, or months.

8. **Optional:** In the **Reason for lock** field, enter a reason for locking the environment.

This reason will be displayed to all users who view the VIE dashboard and it will be logged in the Rational Test Control Panel Audit log. (For information about this log, refer to *IBM Rational Test Control Panel System Administration Guide*.)

9. Click **Lock environment**.

A banner is displayed on the upper half of the VIE Dashboard to indicate that the environment is now locked. If you are the user who locked the environment, you can continue to use the environment as normal but other users will be prevented from making any changes to it.

The following table outlines what is displayed on various Rational Test Control Panel pages while an environment is locked.

Rational Test Control Panel Page	User Who Locked Environment	Other Users Accessing Locked Environment
VIE page (for selecting domain and environment)	<p>Lock icon with tick mark is displayed next to the locked environment.</p> <p>The user can select the environment and use the environment as normal.</p>	<p>Lock icon with an information icon is displayed next to the locked environment.</p> <p>This indicates that the environment is locked to another user.</p> <p>If the user clicks this Lock icon on the VIE page, the View environment lock information dialog box is displayed, which shows user assistance text and the following details:</p> <ul style="list-style-type: none">• The user name of the user who locked the environment.• The names of the domain and the environment.• The start date and time of the lock.• The expected end date and time of the lock (if expected duration was entered).• The lock's reason (if entered). <p>The user can select the locked environment and click View Dashboard.</p>

Rational Test Control Panel Page	User Who Locked Environment	Other Users Accessing Locked Environment
VIE Dashboard page	<p>If the Agents panel is displayed on the lower half of the VIE Dashboard page, a Lock icon with tick mark is displayed next to any stub running in the locked environment.</p> <p>If another user sends a request to unlock the environment, the Environment unlock request dialog is displayed, which enables the user who locked the environment to approve or reject the request and thus unlock the environment or retain the lock (as desired).</p> <p>NOTE: If the user who locked the environment does not respond to an unlock request message within a stated response time range (measured in seconds), Rational Test Control Panel will unlock the environment automatically.</p> <p>NOTE: If an unlock request message is sent to the user who locked the environment but that user is not currently logged into that Rational Test Control Panel instance, Rational Test Control Panel will unlock the environment automatically after the stated response time range has expired because there is no means to notify the user who locked the environment.</p>	<p>The locked environment is displayed but most of the stub actions are unavailable (exceptions include View log and Manage scenarios) and the locked banner on the upper half of the VIE Dashboard page displays summary information about the lock, including the user name of the user who locked the environment and the expected finish time (if specified).</p> <p>Clicking the Information icon on the locked banner displays the View environment lock information dialog box, which provides additional information about the lock.</p> <p>Clicking the Open Lock icon on the locked banner displays the Request unlock dialog box, which enables another user to send a message (within Rational Test Control Panel) to the user who locked the environment requesting that environment to be unlocked. This message will be displayed to the locking user and will be logged in the Rational Test Control Panel Audit log. If the request is rejected, an Environment unlock request status message is displayed.</p>

Rational Test Control Panel Page	User Who Locked Environment	Other Users Accessing Locked Environment
Agents page	<p>Lock icon with a tick icon is displayed for each running project that is running in a locked environment.</p> <p>Unlike other users, the user who locked the environment can use the Agents page to clear down project and to stop stubs.</p>	<p>NOTE: If the user requesting the unlock is an administrator user, the Unlock environment dialog box is displayed, which enables the user to unlock the environment immediately without having to submit a request.</p> <p>NOTE: If the security functionality of Rational Test Control Panel is disabled (for information about this, refer to <i>IBM Rational Test Control Panel Installation Guide</i> or <i>IBM Rational Test Control Panel System Administration Guide</i>), all users have administrator privileges, so any user can unlock a locked environment at any time.</p> <p>If the Agents panel is displayed on the lower half of the VIE Dashboard page, a Lock icon with an information icon is displayed next to any stub running in the locked environment.</p>

NOTE: If the security functionality of Rational Test Control Panel is disabled (for information about this, refer to *IBM Rational Test Control Panel Installation Guide* or *IBM Rational Test Control Panel System Administration Guide*), user names are not displayed when displaying information about a locked environment.

5.2.5.2 Unlocking an Environment

To unlock an environment:

1. Log into Rational Test Control Panel.

NOTE: If you are not an administrator user, you must use the user name that you used to lock the environment.

The Rational Test Control Panel application window is displayed.

2. Click the **VIE** icon or navigation link.

The VIE page is displayed.

3. Click the relevant domain and environment.

4. Click **View Dashboard**.

The VIE Dashboard page is displayed.

5. Click the **Open Lock** icon in the upper-right corner of the VIE Dashboard page.

Alternatively, click the **Open Lock** icon on the locked banner.

The environment is unlocked.

NOTE: All environment locking and unlocking events, including reasons, are recorded in the Rational Test Control Panel Audit log. (For information about this log, refer to IBM Rational Test Control Panel System Administration Guide.)

5.2.6 Viewing Stub Error Messages

On Rational Test Control Panel's VIE Dashboard page, if **Error** is displayed, resting your mouse pointer over **Error** displays a message about the error.

If an error message includes the "see agent log", you should refer to the agent's console output and not to any Rational Test Control Panel logs.

5.2.7 Viewing Stub Logs

Rational Test Control Panel enables Rational Test Virtualization Server users to view the console output of stubs that have been run in the currently selected domain and environment.

If a domain-level project results database override has been set up for the currently selected domain (for information about this, Rational Test Control Panel

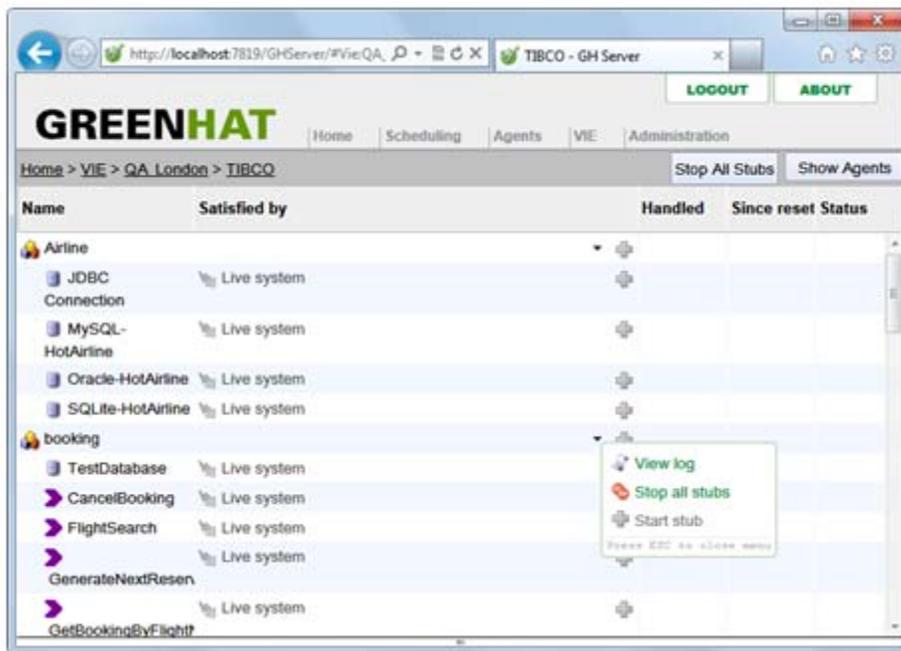
administrators should refer to *IBM Rational Test Control Panel System Administration Guide*), only the results database specified in the override is used for displaying the console output of stubs in that domain.

Otherwise, the results databases for all projects published to the currently selected environment that contain the selected stub are used to display the console output of that stub.

When stub logs are viewed, the connection to the specified project results database will be made from the computer running Rational Test Control Panel. When a stub writes its logs, the connection will be made from the computer running the agent upon which the stub is running. (This may not be the computer running Rational Test Control Panel.)

To view a stub's log:

1. On Rational Test Control Panel's VIE Dashboard page, under the **Satisfied by** column, click the **arrow** button () of the **component** (not the operation) that contains the stub.



2. Click **View log** on the shortcut menu.

The operation's stubs log page is displayed.

NOTE: If the log page is not displayed or if an empty log page is displayed, refer to [Resolving Stub Log Display Problems](#).

3. **Optional:** Under **Auto scroll**, selecting the **Enable auto scroll on refresh** check box scrolls displays the bottom of the page automatically when new log data arrives.
4. **Optional:** Under **Refresh**, select a refresh interval in the **Interval** list:
 - Clicking **Manual** in the **Interval** list makes the **Refresh** button available, which enables you to refresh the page as often as you wish.
 - Clicking one of the other options in the **Interval** list will refresh the page at set intervals (**10 seconds**, **30 seconds**, **1 minute**, or **5 minutes**).

Under **Warnings**, messages are displayed if results database connection details are incorrect, or if the specified results database is not running, or if its drivers cannot be detected.

Under **Sessions**, any session interactions with the stub are displayed. The same information is also displayed in Rational Integration Tester's Test Lab perspective's Console window.

5. **Optional:** Under **Event types**, selecting and clearing the check boxes of the various event-types under which stub log entries can be classified enables you to filter the log entries displayed on the stubs log page.

5.2.8 Debugging Message-Based Stubs

If you are using Rational Integration Tester 8.0.1 (or later), you can use the Message Differences window to debug any message-based stubs that generate error messages on the Test Lab perspective's Console window. Debugging the stub enables you to determine why one (or more) events in the stubs was (were) not matched.

The main benefits of using the Message Differences window in this situation are as follows:

- You do not have to move from the Test Lab perspective to the Test Factory perspective and analyze each expected message to determine why it did not match any of the events.
- You have access to the received messages for side-by-side comparison with all expected messages from the events associated with the relevant operation in the currently selected stub.

NOTE: This section discusses only features of the Message Differences window that are relevant to stubs. For general information about

using the Message Differences window, refer to *IBM Rational Integration Tester Reference Guide*.

NOTE: The Message Differences window cannot be used to debug a database stub created by Rational Integration Tester.

To debug a message-based stub that is taking the default case when it should be matching a particular event:

1. Click the stub's default case console output or failed validation console output on the Test Lab perspective's Console window.

The Message Differences window is displayed.

NOTE: You can use all of Rational Integration Tester's perspectives while the Message Differences window is displayed. However, if you use the Stub Editor (or the Test Editor) to modify a stub (or a test) while also using the Message Differences window to modify the same resource, there will be two sets of pending changes to that resource. Therefore, when you attempt to close the Message Differences window, Rational Integration Tester will prompt you to select the set of changes you want to keep.

The Message Differences window collects all of the events associated with the relevant operation in the stub into one location. The events are displayed on the left side of the window and the message received is displayed on the right side of the window.

2. Click the navigation buttons under **Expected Message** on the upper-left side of the window to move up and down through the events.
3. Analyze any errors on the left side of the Message Differences window.
4. Use the buttons under **Repair** and **Action** on the upper-right side of the Message Differences window to edit or repair any messages as required; or to close and run the stub again; or to raise a defect if necessary. (For more information about these buttons, refer to *IBM Rational Integration Tester Reference Guide*.)

NOTE: The **Re-run** button is unavailable because it applies only to tests.

5. Click **Close and Re-run** to save your changes and to run the stub again.

Alternatively, if running Rational Integration Tester from memory, click **Apply** to save your changes and to run the stub again.

Managing Agents

Contents

[Viewing All Running Agents Registered with Current RTCP Instance](#)

[Viewing Agents For a Specific Domain/Environment](#)

This chapter describes how to use Rational Test Control Panel to view Rational Integration Tester Agents.

Agents are important components of Rational Test Virtualization Server because they execute stubs that have been published to Rational Test Control Panel.

An agent must be installed on each computer where you want to run stubs and there must be at least one agent in an environment.

Each agent must register with a Rational Test Control Panel instance so that it can accept requests to run stubs.

6.1 Viewing All Running Agents Registered with Current RTCP Instance

NOTE: For background information about agents, refer to *IBM Rational Integration Tester Reference Guide*. For information about deploying agents with Rational Test Control Panel, refer to *IBM Rational Test Control Panel Installation Guide*. For information about installing agents for Rational Test Virtualization Server, refer to *IBM Rational Integration Tester Agent Installation Guide*.

To view **all** agents that are registered with the current Rational Test Control Panel instance and that are currently running:

1. Log into Rational Test Control Panel.

Rational Test Control Panel's application window is displayed.

2. Click the **Agents** icon or navigation link.

The **Agents** page is displayed.



Any agents registered with the current Rational Test Control Panel instance that are currently running are displayed. Details displayed include IP address, port number, domain and environment, software version number, configuration file location,

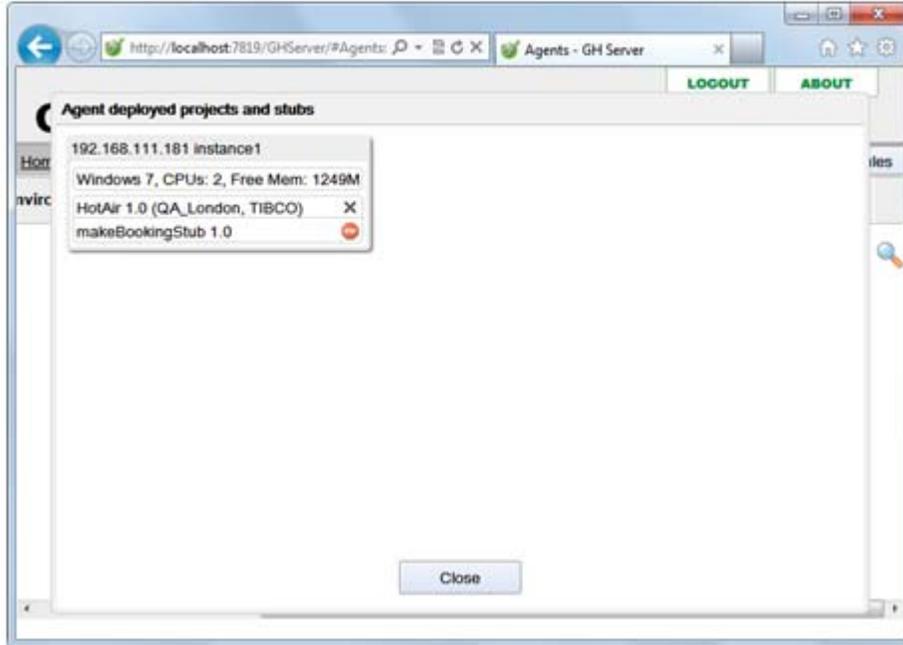
current status, and date and time of last poll. If you are using Rational Test Control Panel 8.0.1 (or later), the agent type is also displayed. For example, **RPTS** for Rational Performance Test Server, or **RTVS** for Rational Test Virtualization Server, or **Unknown** if the agent cannot be determined.

NOTE: A Rational Integration Tester Agent can be registered with only one Rational Test Control Panel instance at any given time.

In Rational Test Control Panel 8.0.1 (or later):

- When Rational Test Virtualization Server, Rational Performance Test Server, or Probe agents disconnect, they remain in **Disconnected** state until they are removed manually from the **Agents** page. They can be removed individually by using the **X** button (✕) on the right of each row or by using the **Remove all disconnected engines** button on the button bar at the top of the **Agents** page. In contrast, HTTP, TCP, and other proxy agents remove themselves from the **Agents** page automatically after a few minutes of being in **Disconnected** state.
 - If a Rational Test Virtualization Server, Rational Performance Test Server, or Probe agent reconnects without changing its configuration (even after a restart), its status will be changed to **OK** (that is, a new row will not be created on the Agents page and the old row on the **Agents** page will be reused). However, if the configuration of a Rational Test Virtualization Server, Rational Performance Test Server, or Probe agent has been changed before restarting, a new row will be created on the **Agents** page and the old row will remain visible on the **Agents** page with status **Disconnected**.
3. Click the **magnifying glass** button (🔍) of the relevant engine.

The Agent deployed projects and stubs pop-up window is displayed on the **Agents** page.



Details about the selected agent are displayed on the pop-up window, including the IP address of computer where the agent is deployed.

4. Click **Close** to close the Agent deployed projects and stubs pop-up window.

6.2 Viewing Agents For a Specific Domain/Environment

To view a listing of the agents for a particular domain/environment:

1. Log into Rational Test Control Panel.

Rational Test Control Panel's application window is displayed.

2. Click the **VIE** icon or navigation link.

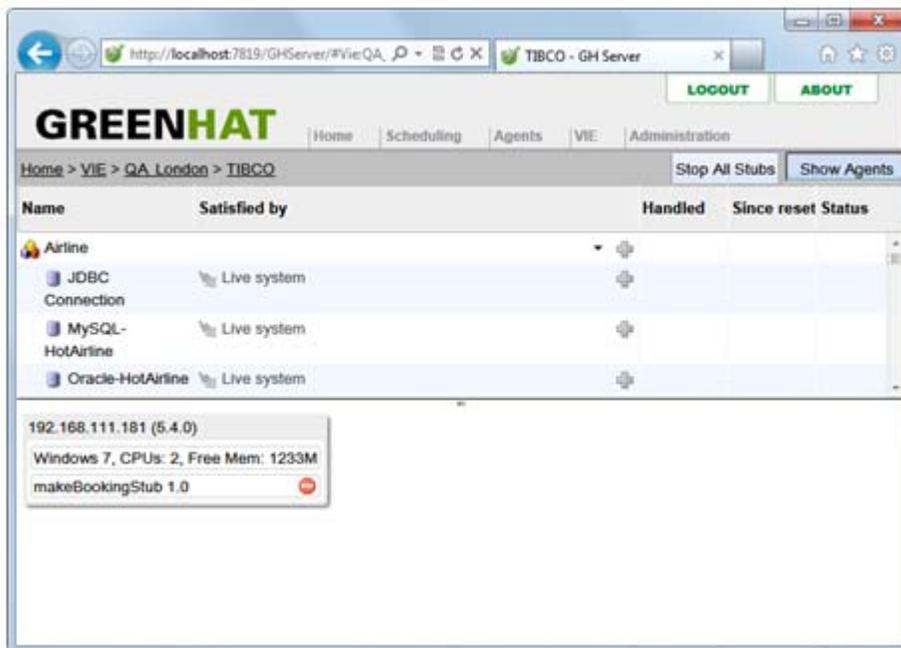
The **VIE** page is displayed.

3. Click the relevant domain and environment and then click **View Dashboard**.

The VIE Dashboard page is displayed.

4. Click **Show Agents**.

The lower half of the VIE Dashboard page displays a pop-up window for each agent for the selected domain/environment that is running.



NOTE: Agent console output is not viewable in Rational Test Control Panel, so you must view the console output of an agent on the computer where it is running.

Troubleshooting

Contents

Handling Agent Failures

Resolving Stub Log Display Problems

This chapter provides information about how to troubleshoot Rational Test Virtualization Server.

7.1 Handling Agent Failures

If an agent fails, you can “close” any deployed Rational Test Virtualization Server projects involving that agent.

To close a deployed project:

1. Log into Rational Test Control Panel.

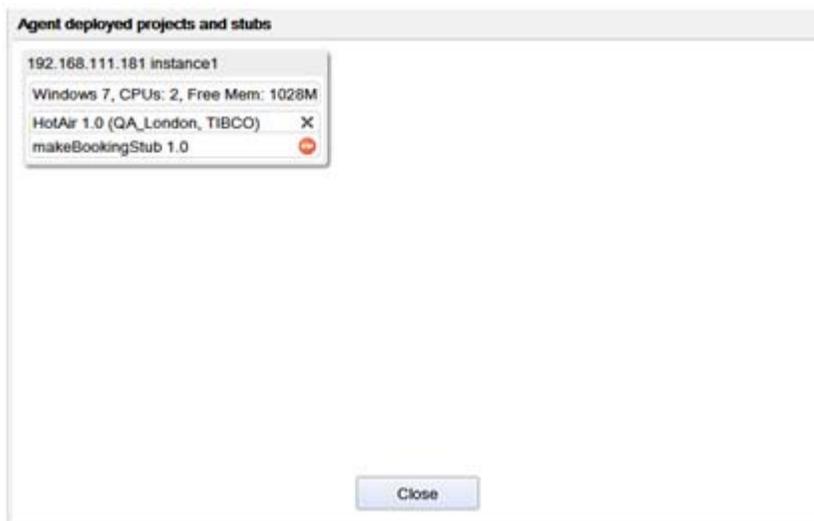
Rational Test Control Panel’s application window is displayed.

2. Click the **Agents** icon or navigation link.

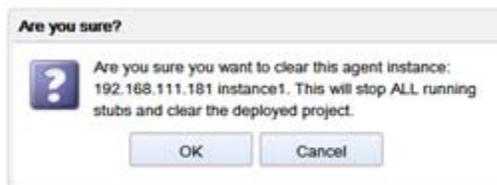
The **Agents** page is displayed.

3. Click the **magnifying glass** button (🔍) of the relevant engine.

The Agent deployed projects and stubs pop-up window is displayed on the **Agents** page.



4. On the pop-up window of the agent that has failed, click the **X** button (✕) next to each project that you wish to close (closing all projects will also stop any stubs that are running for those projects). A confirmation prompt is displayed.



5. Click **OK**.

7.2 Resolving Stub Log Display Problems

The following table outlines reasons why stub logs might not be displayed and describes how to resolve any problems.

Possible Reason	Resolution
The stub is not logging events.	<p>In Rational Test Control Panel's VIE dashboard (or in Rational Integration Tester), verify that the stub has been configured to <code>Log</code>.</p> <p>Alternatively, if <code>localhost</code> has been configured as the logging URL, logging will not work if the stub is run on a different agent.</p>
The results database for the stub (or to be more precise, the project from which the stub was published) is not reachable by the stub running on the agent.	Ensure that the computer running the stub and the computer running Rational Test Control Panel have the required network connectivity to access the project results database.
The required database driver is missing.	<p>If you are using a MySQL driver, ensure that it has been installed correctly (for information about this, refer to <i>IBM Rational Test Control Panel Installation Guide</i>).</p> <p>If you are using a non-standard JDBC driver, ensure that it is included in the <code>CLASSPATH</code> of the computer where Rational Test Control Panel is installed.</p>

Appendix A: Using the Data Model Editor

Contents

[Creating Data Models](#)

[Editing Data Models](#)

[Deleting Data Models](#)

In Rational Integration Tester, you can create a data model while creating a data model-driven stub from recorded events (for information about this, refer to [Creating Data Model-Driven Stubs](#)).

Alternatively, you can use the Data Model Editor.

This appendix describes how to use the Data Model Editor to create, modify, and delete data models.

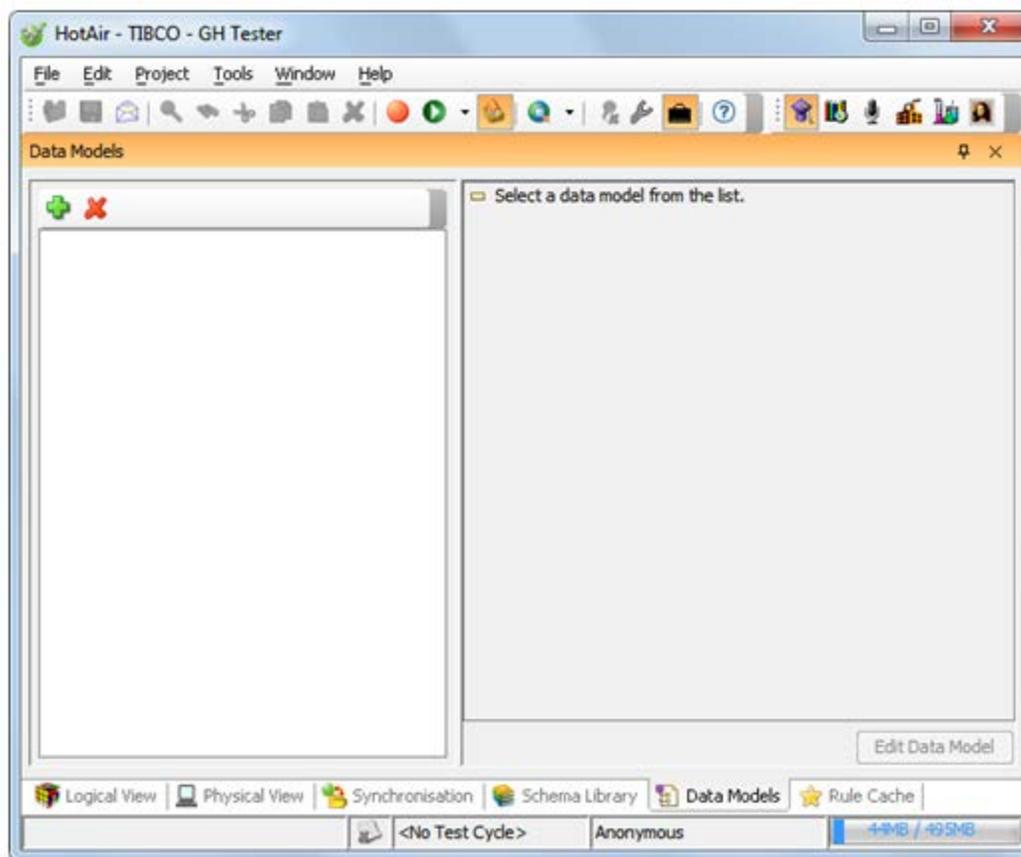
8.1 Creating Data Models

A data model enables you to understand the data held by the system that you want to simulate, and it can provide persistent storage of that data.

To create a new data model:

1. Open Rational Integration Tester's Architecture School perspective.
2. Click the **Data Models** tab.

The Data Models window is displayed.



3. Click the **plus** button (+).

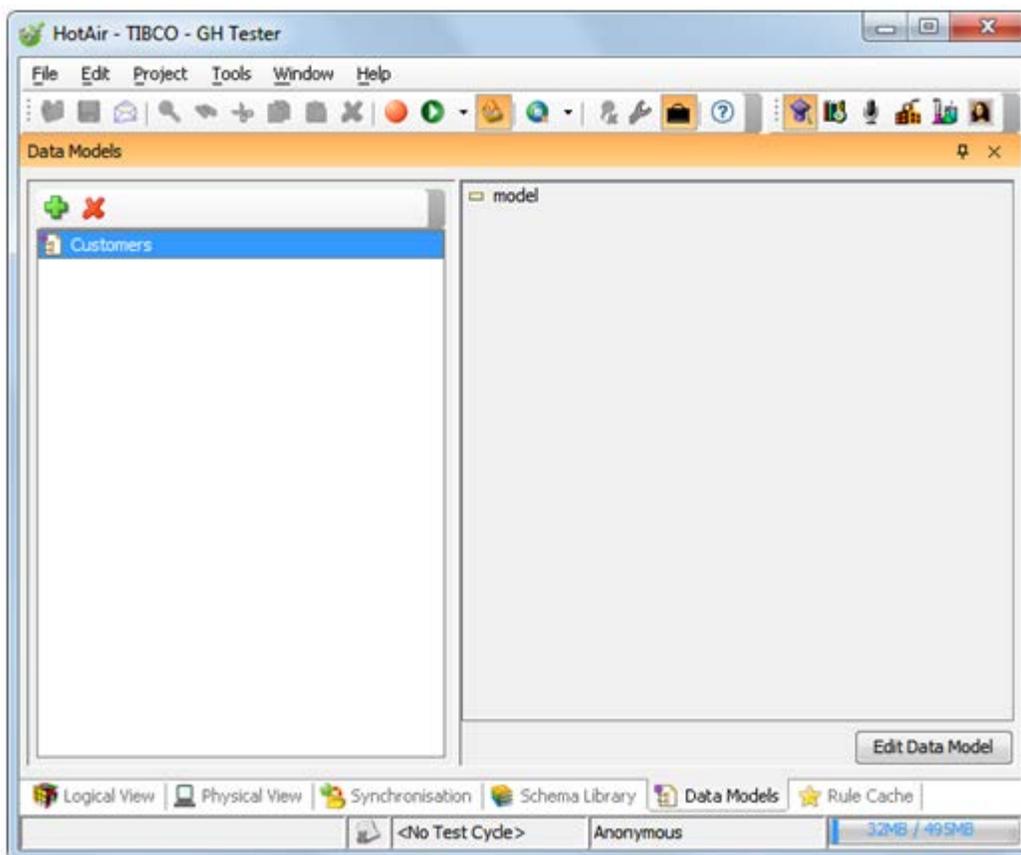
The Input dialog box is displayed.



4. Enter a name for the new data model.
5. Click **OK**.

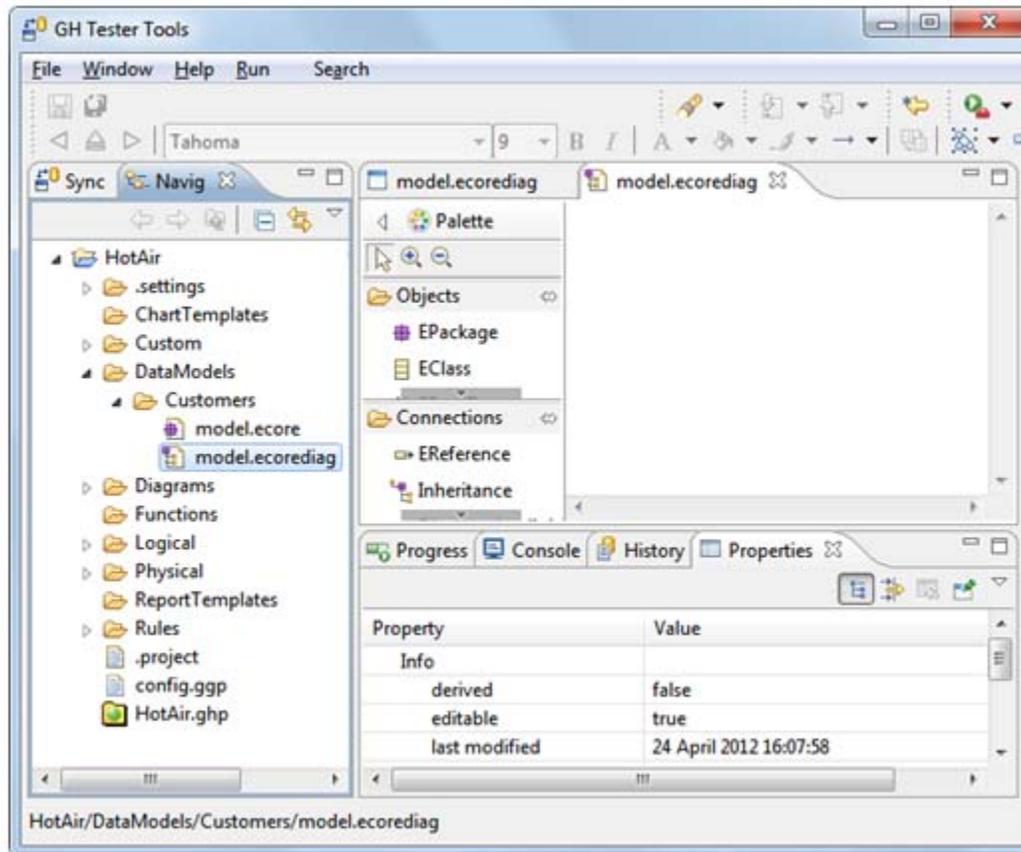
The name of the new data model is displayed on the upper left of the Data Models window.

6. Click the new data model.



7. Click **Edit Data Model**.

The Rational Integration Tester Tools application is opened and the Data Model Editor is displayed.



The instance data for a data model is maintained within a set of comma-separated value (CSV) files. Whenever the data model editor is used to make changes to a data model, appropriate changes are made to the instance data store. The changes are applied as a result of saving the updated data model diagram. If any additional information is needed, it is requested at this point.

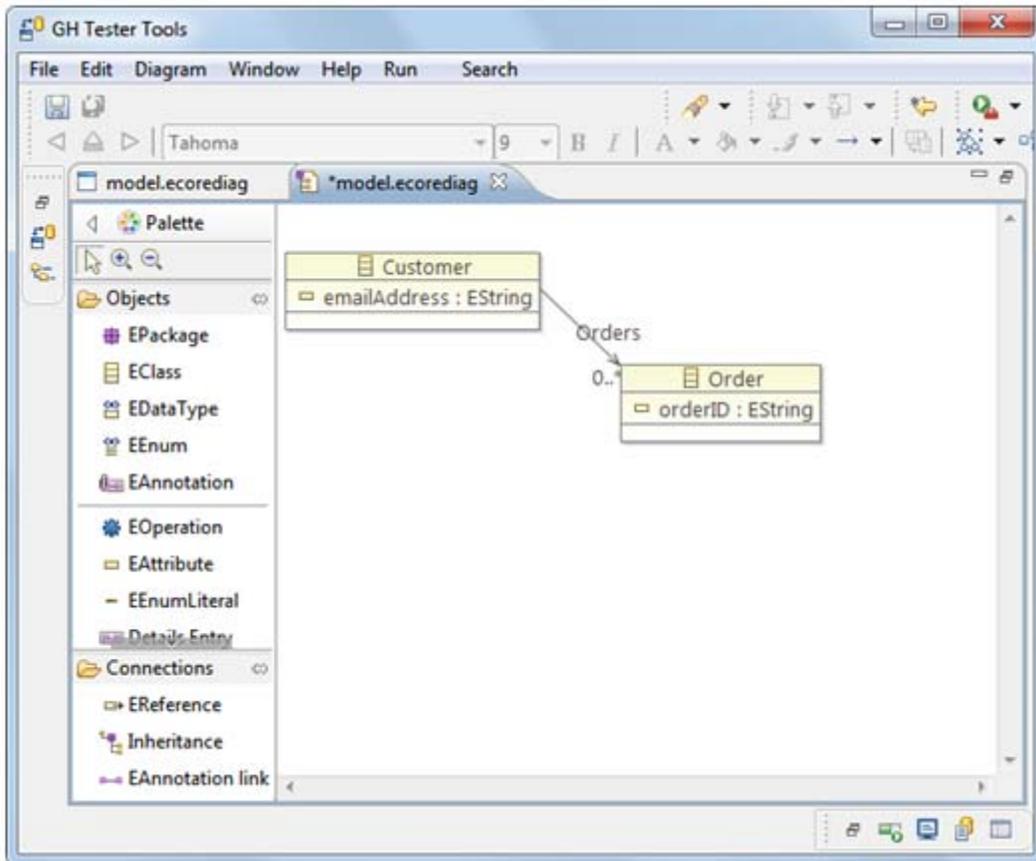
The **Navigator** tab on the left of the Rational Integration Tester Tools application window displays a collection of folders, including a **DataModels** folder, for the currently selected Rational Integration Tester project.

The following table describes how to use the palette (on the **model.ecorediag** tab to the left of the blank canvas) to add entities, attributes, and references.

To add...	Do this...
An entity. For example, <code>Customer</code> .	<ol style="list-style-type: none">1. Click EClass.2. Click anywhere on the blank canvas to insert a new entity.3. Right-click the new entity.4. Click Show Properties View on the shortcut menu.5. In the Name field (on the Properties tab under the canvas), enter a name for the new entity.6. Press ENTER (or move to another field on the Properties tab) to apply the new name.
An attribute to an entity. An attribute is a named scalar property of an instance of an entity. For example, an entity <code>Customer</code> might have an attribute <code>emailAddress</code> of a Java <code>java.lang.String</code> type. In general, the type (EType) that you will select in the Data Model Editor will be one of the following: <ul style="list-style-type: none">• EString (equivalent to a Java <code>java.lang.String</code> type)• EBoolean (equivalent to a Java primitive <code>boolean</code>)• EDate (equivalent to a Java <code>java.util.Date</code> type)• EDouble (equivalent to a Java primitive <code>double</code> type)• EInt (equivalent to a Java primitive <code>int</code> type)• ELong (equivalent to a Java primitive <code>long</code> type)	<ol style="list-style-type: none">1. Click EAttribute.2. Click the relevant entity on the canvas.3. In the second row of the entity object on the canvas, enter the name of the attribute.4. On the Properties tab under the canvas, click the Browse button next to the EType field.5. On the Object selection dialog box, select an appropriate type.6. Click OK.

To add...	Do this...
<p>A reference between entities.</p> <p>A reference indicates a relationship between two entities.</p> <p>For example, there might be a relationship between a customer and the number of orders that that customer has placed</p>	<ol style="list-style-type: none"> 1. Create the entities on the canvas. 2. Click EReference. 3. On the canvas, draw a line from the entity that holds the reference to the entity that is the type of the reference. 4. Right-click the line. 5. Click Show Properties View on the shortcut menu. 6. In the Name field (on the Properties tab under the canvas), enter a name for the reference. 7. In the Upper Bound field, enter 1 (for a 0..1 reference) or -1 (for a 0..* reference). 8. Press ENTER (or move to another field on the Properties tab) to apply the new name.

After you create a data model, the data is held inside a set of entities, and each entity has a set of attributes.



8.2 Editing Data Models

Editing a data model involves modifying or deleting components of the model.

NOTE: When editing data models, ensure that you click the correct **DataModels** folder on the **Navigator** tab on the Rational Integration Tester Tools application window before making any changes.

The following table describes how to use the Data Model Editor in the Rational Integration Tester Tools application to edit a data model.

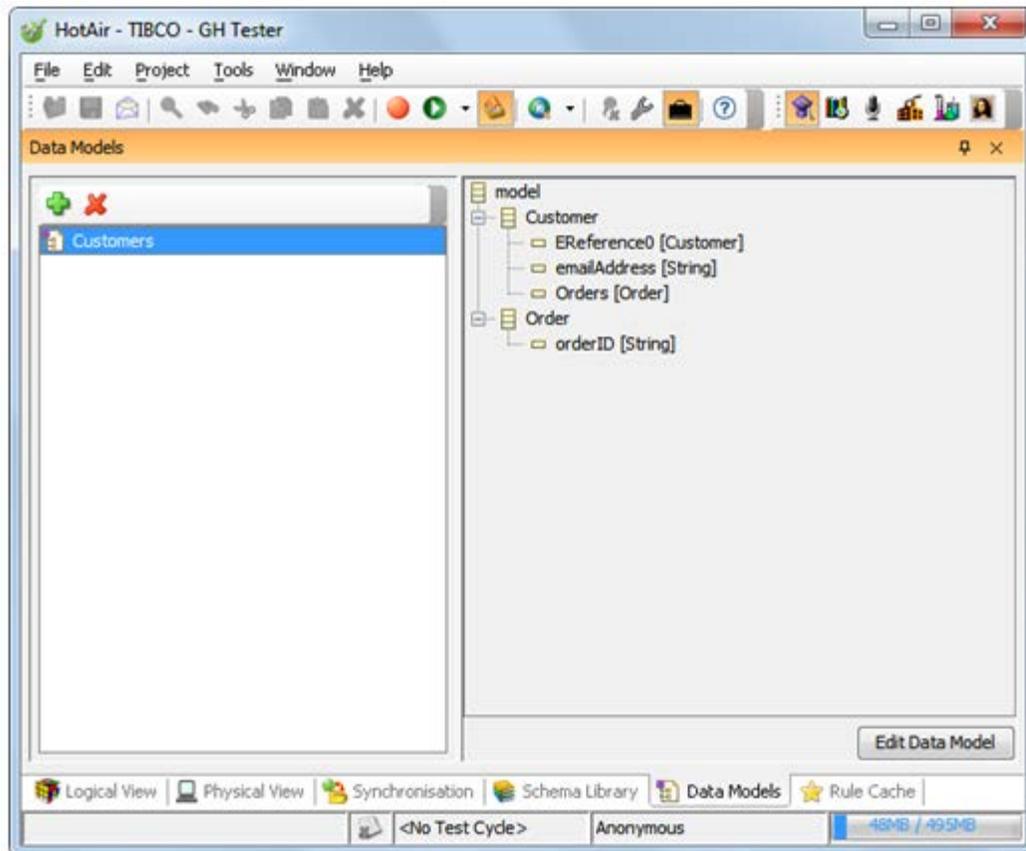
To...	Do this...
Rename an entity, attribute, or reference.	<ol style="list-style-type: none">1. On the canvas, right-click the entity, attribute, or reference (as applicable).2. Click Show Properties View on the shortcut menu.3. In the Name field, modify the name of the entity, attribute, or reference (as applicable).4. Press ENTER.
Move an attribute between entities. NOTE: An attribute can be moved from one entity to another. If the model has instance data, the data can be mapped between instances of the affected entities by providing a foreign key mapping from the original entity to the new entity when the Data Model Editor is saved.	<ol style="list-style-type: none">1. On the canvas, drag an attribute from the original entity object and drop it on the second row of new entity object.2. Click Save to open the Mapping dialog box, which enables you to specify how instances of one entity can be mapped to another entity.3. On the Mapping dialog box, select the Use these mappings for all feature moves from <Source Entity> to <Target Entity> check box if you want to reuse the current mapping for any future attribute moves between the currently selected entities. Alternatively, click No Mapping if you do not want to create a mapping between the currently selected entities.

To...	Do this...
Move a reference between entities.	<ol style="list-style-type: none"> 1. On the canvas, move the source (non-arrow) end of the reference line to the target entity. 2. Click Save to open the Mapping dialog box, which enables you to specify how instances of one entity can be mapped to another entity. 3. On the Mapping dialog box, select the Use these mappings for all feature moves from <Source Entity> to <Target Entity> check box if you want to reuse the current mapping for any future attribute moves between the currently selected entities. Alternatively, click No Mapping if you do not want to create a mapping between the currently selected entities.
Modify the type of a reference.	<ol style="list-style-type: none"> 1. On the canvas, move the destination (arrow) end of the reference line to the target entity. 2. Click Save to open the Mapping dialog box, which enables you to specify how instances of one entity can be mapped to another entity. 3. On the Mapping dialog box, select the Use these mappings for all feature moves from <Source Entity> to <Target Entity> check box if you want to reuse the current mapping for any future attribute moves between the currently selected entities. Alternatively, click No Mapping if you do not want to create a mapping between the currently selected entities.
Modify the cardinality of a reference.	<ol style="list-style-type: none"> 1. On the canvas, right-click the reference. 2. Click Show Properties View on the shortcut menu. 3. In the Upper Bound field, enter 1 (for a 0..1 reference) or -1 (for a 0..* reference). 4. Press ENTER.

To...	Do this...
<p>Merge two attributes of an entity.</p>	<ol style="list-style-type: none"> 1. On the canvas, hold CTRL and click the two attributes of an entity that you want to merge. 2. Click Merge Attributes on the shortcut menu. 3. On the Merge Attributes dialog box: <ul style="list-style-type: none"> • In the Attribute to retain list, click the attribute that you want to retain as the resulting single attribute. • In the Precedent attribute list, click the attribute that has the value that will be used in situations where an instance of that type has data for each attribute.
<p>Delete an entity, attribute, or reference. NOTE: If you delete an entity, any stored instance data for that entity is also deleted. If you delete an attribute or a reference, all data for that attribute/reference is deleted from instances of the entity that contains that attribute/reference.</p>	<ol style="list-style-type: none"> 1. On the canvas, right-click the entity, attribute, or reference (as applicable). 2. Click Delete from Model on the shortcut menu.

To verify any changes that you have made to a data model:

1. After clicking **Save** in the Data Model Editor, view the Data Models window on Rational Integration Tester's Architecture School perspective.



2. If there is more than one data model listed on the upper left of the Data Models window, click the data model that you have modified in the Data Model Editor to refresh its details on the Data Models window. (You might have to reselect the data model to refresh the window.)

8.3 Deleting Data Models

If you delete a data model, any resources in Rational Integration Tester that reference that data model will no longer be valid.

To delete a data model:

1. Open Rational Integration Tester's Architecture School perspective.
2. Click the **Data Models** tab.
3. On the upper left of the Data Models window, click the data model that you want to delete.
4. Click the red **X** button (✘).

A confirmation prompt is displayed.

5. Click **Yes**.

The selected data model is deleted.

Appendix B: Creating Behaviours

Contents

[Introduction](#)

[Creating a Behaviour](#)

[Defining Interfaces](#)

This appendix describes how to create a simple behaviour called “echo” that will expose one function and one event for use by a Rational Test Virtualization Server stub.

For information about how behaviours are used by Rational Test Virtualization Server, refer to [Creating & Modifying Message-Based Stubs](#).

9.1 Introduction

Behaviours are reusable behavioural units that can expose utility or business logic for use within Rational Test Virtualization Server. They can be combined to build rich virtualized applications with minimal development effort.

A behaviour interacts with a stub in two ways:

- It provides methods that can be used within script actions in the business logic of a stub. This enables the stub to control the behaviour at run time.
- It can raise events to the stub, passing information that tells the stub that the behaviour needs to interact in some way, such as sending a message.

Behaviours are discovered by Rational Test Virtualization Server at run time for use in stubs by means of an Eclipse extension point:

`com.greenhat.testers.api.behaviour` and they have the following four primary elements:

1. The **factoryClass** is a Java class whose package is exported by the behaviour-contributing plugin that implements the `com.greenhat.testers.api.behaviour.BehaviourFactory` interface. The implementation of the class creates and returns an implementation of the behaviour and typically will configure and add listeners to it.
2. The **behaviourInterface** is a Java interface that is an extension of the `com.greenhat.testers.api.behaviour.LifecycleAwareBehaviour` interface whose package is exported by the behaviour-contributing plug-in, which exposes the functions of the behaviour.
3. The **callbackInterface** is a Java interface whose package is exported by the behaviour--contributing plugin and exposes the events exposed as call-backs by the behaviour.
4. The **implementation** of the behaviour is custom code that is typically instantiated and configured by the **factoryClass**, which implements the **behaviourInterface** and makes call-backs on the **callbackInterface**.

9.2 Creating a Behaviour

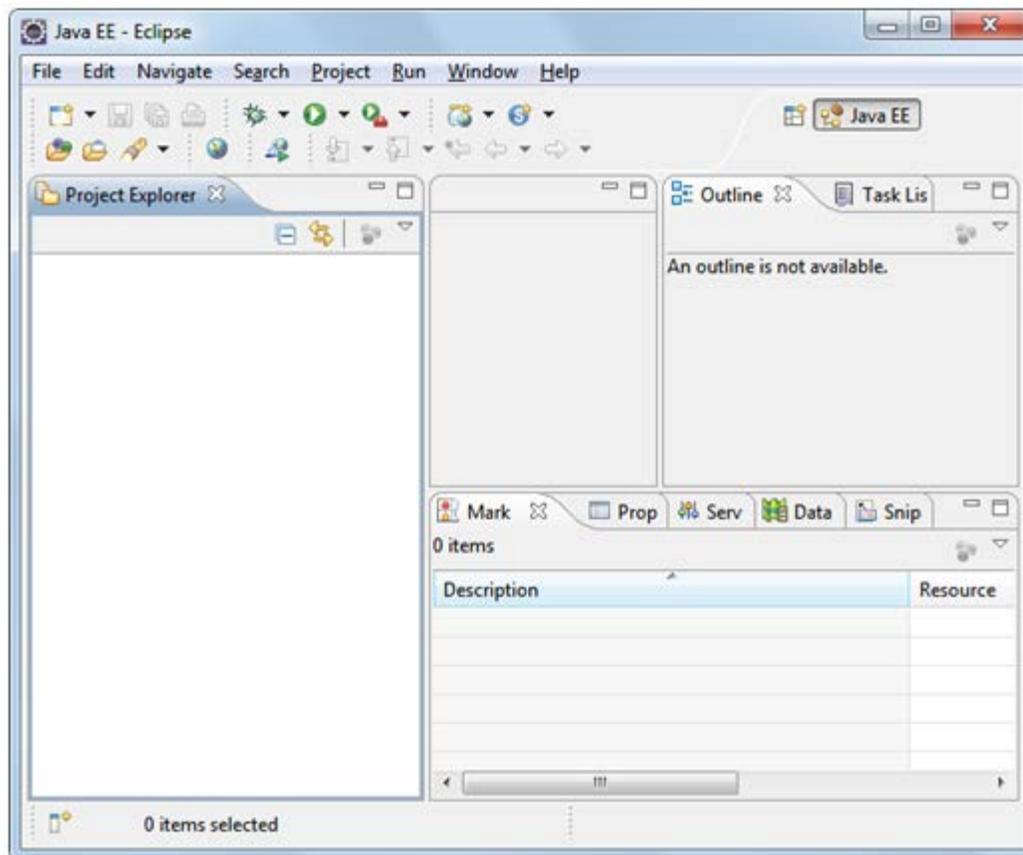
Before creating a behaviour, you will need to:

- Install an Eclipse IDE with the Plug-in Development Environment (PDE). For example, Eclipse for RCP and RAP Developers, which can be downloaded from www.eclipse.org
- Copy the `com.greenhat.testers.api_<Version Number>.jar` file from `<Rational Integration Tester Installation Directory>\plugins` to `<Eclipse IDE Installation Directory>\plugins` on the target computer.

To create a new behaviour for Rational Test Virtualization Server:

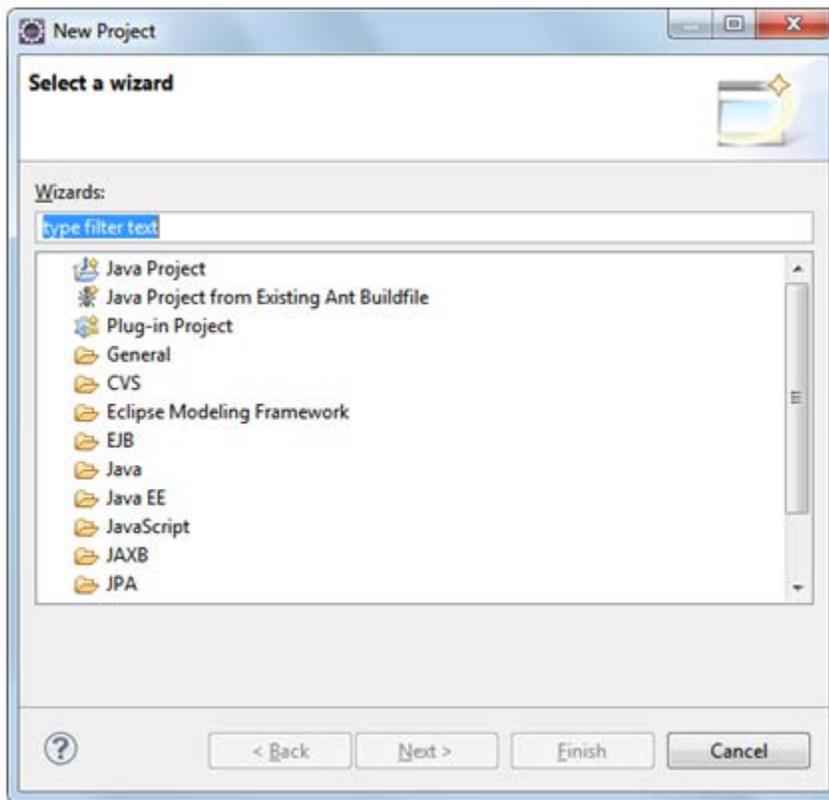
1. Open Eclipse.

The Eclipse application window is displayed.



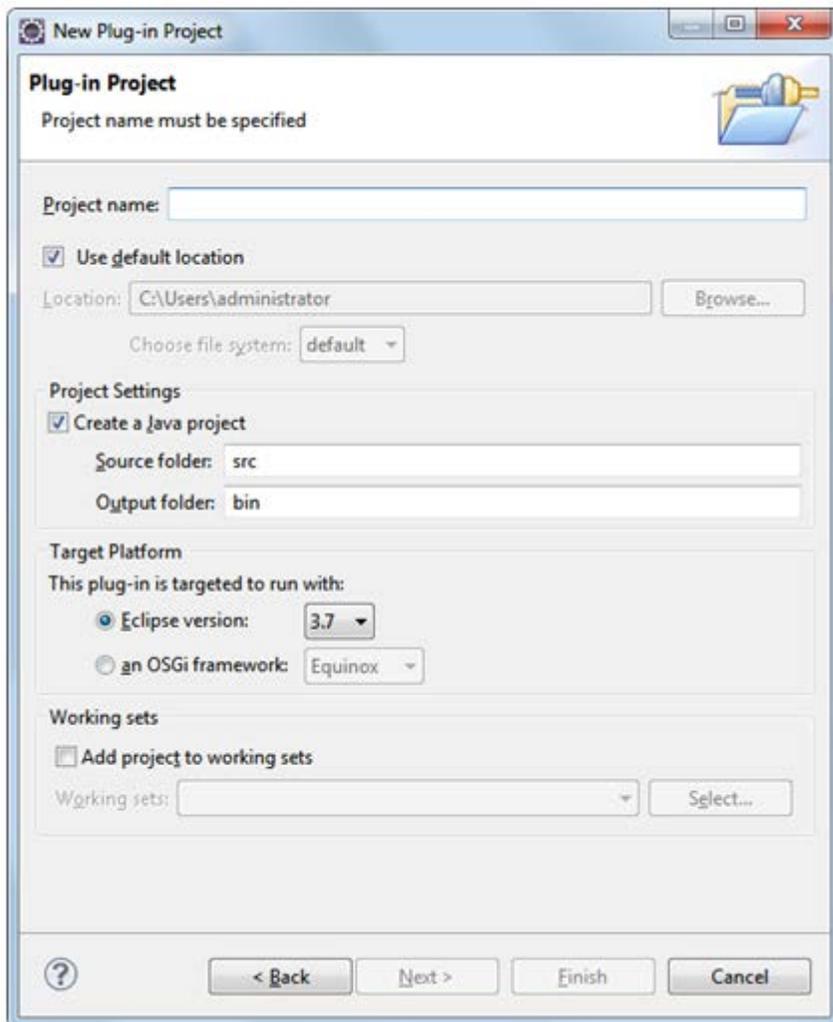
2. Click **File > New > Project**.

The New Project wizard is displayed.

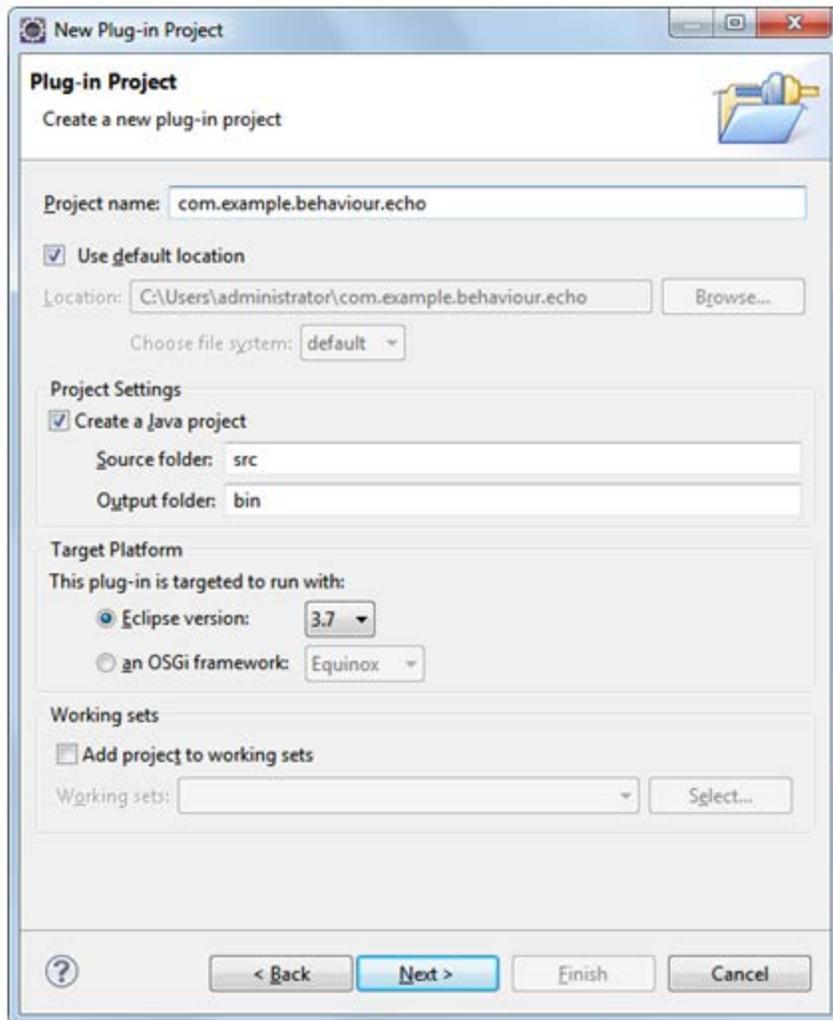


3. Click **Plug-in Project**.
4. Click **Next**.

The first screen of the New Plug-in Project wizard is displayed.

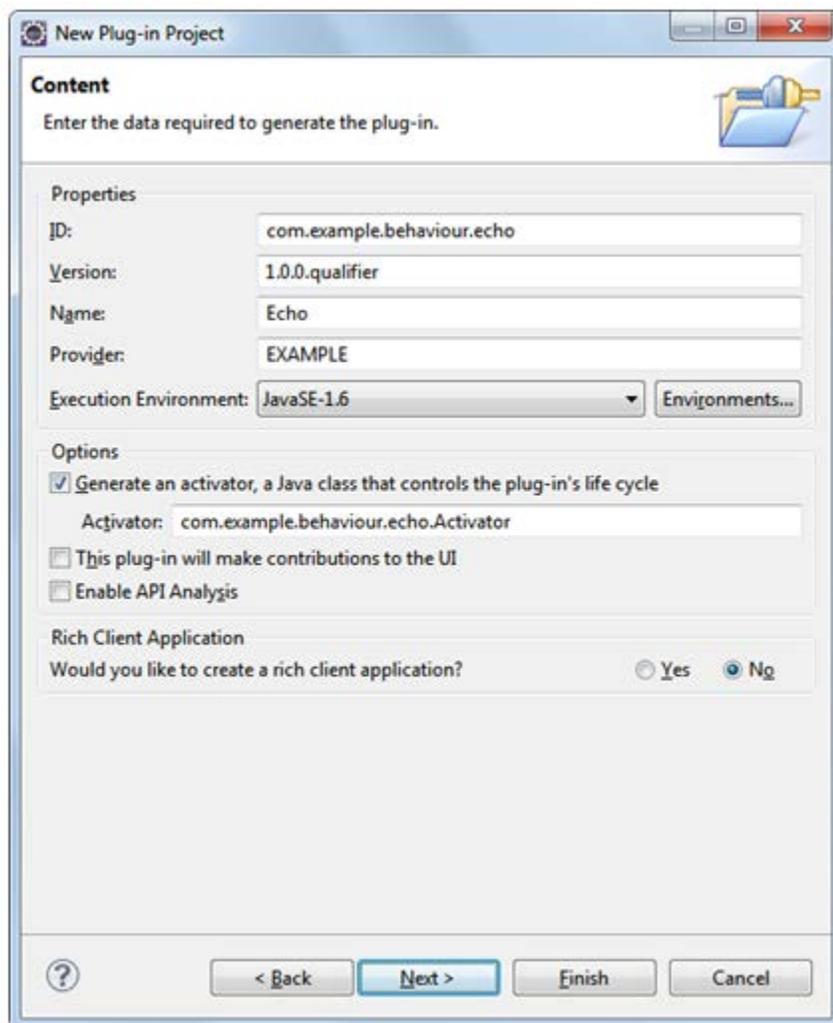


5. In the **Project name** field, enter `com.example.behaviour.echo`.
6. Select the **Use default location** check box.
7. Select the **Create a Java project** check box.
8. In the **Source folder** field, enter `src`.
9. In the **Output folder** field, enter `bin`.
10. Click the **Eclipse version** option button.
11. In the **Eclipse version** list, select the appropriate platform version.
12. Clear the **Add project to working sets** check box.



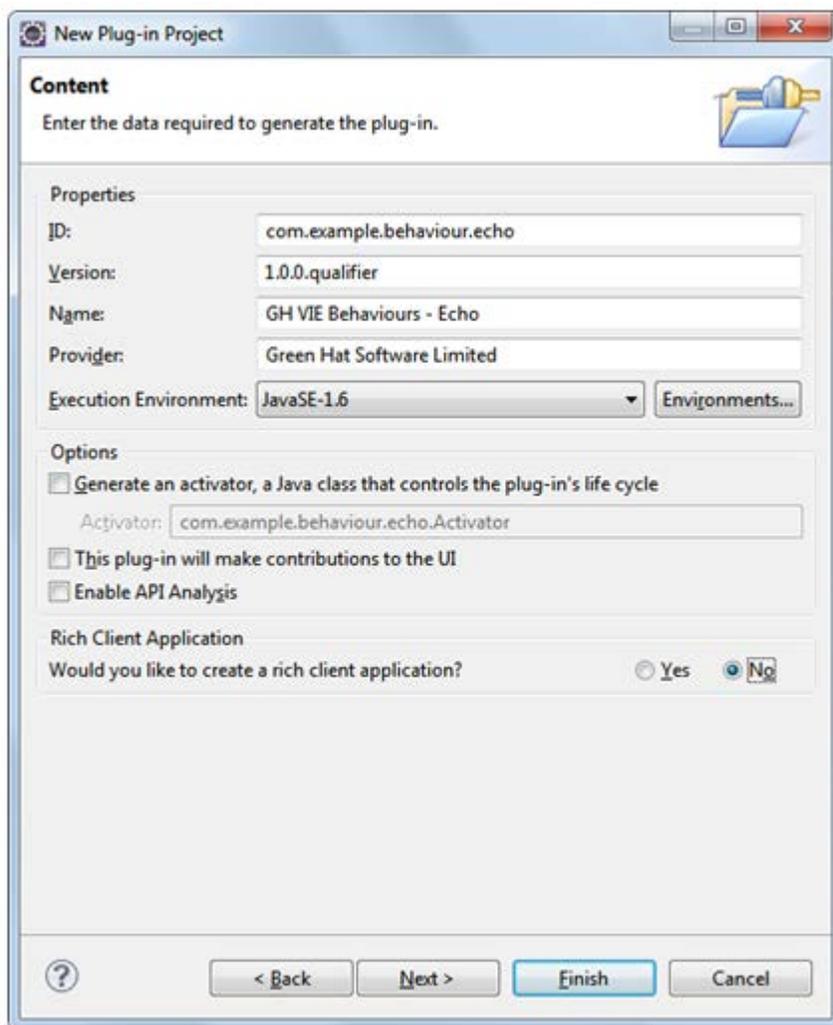
-
- Click **Next**.

The second screen of the New Plug-in Project wizard is displayed.



- In the **ID** field, enter `com.example.behaviour.echo` (if necessary).
- In the **Version** field, enter `1.0.0 qualifier` (if necessary).
- In the **Name** field, enter `Rational Test Virtualization Server Behaviours - Echo` (or an equivalent name).
- In the **Provider** field, enter `IBM Corporation`.
- In the **Execution Environment** field, click **JavaSE-1.6**.
- Under **Options**, clear all the check boxes.

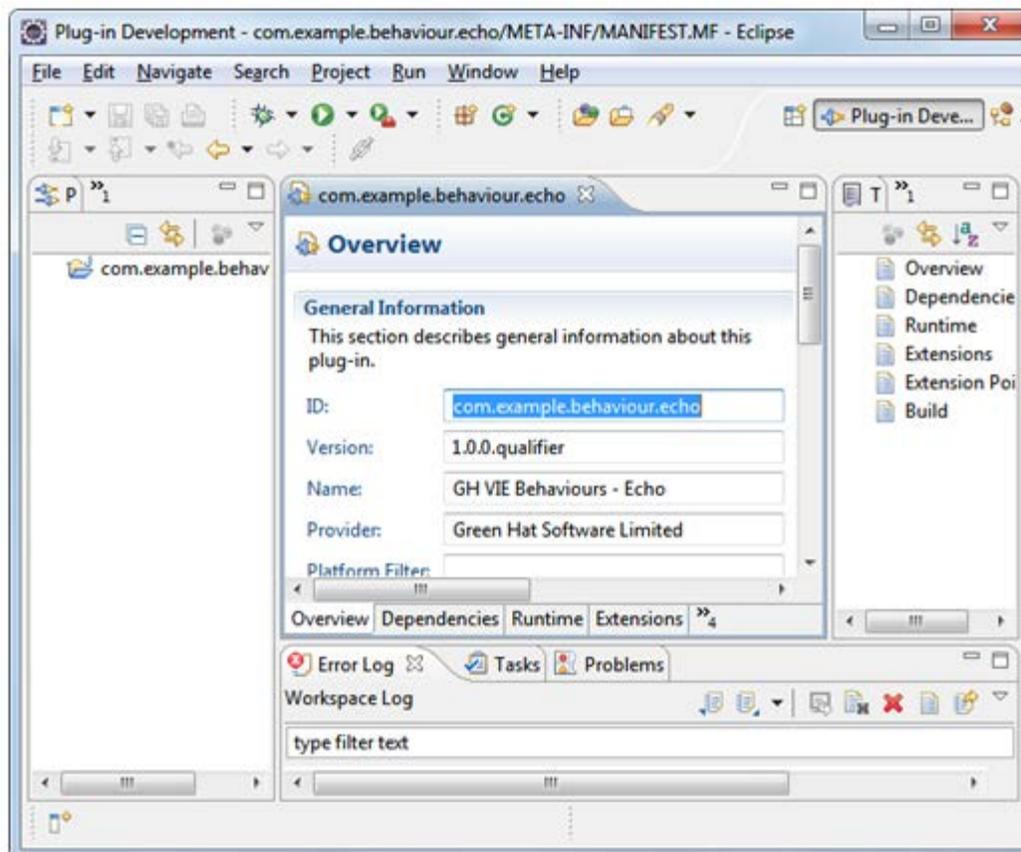
-
20. Under **Rich Client Application**, click the **No** option button.



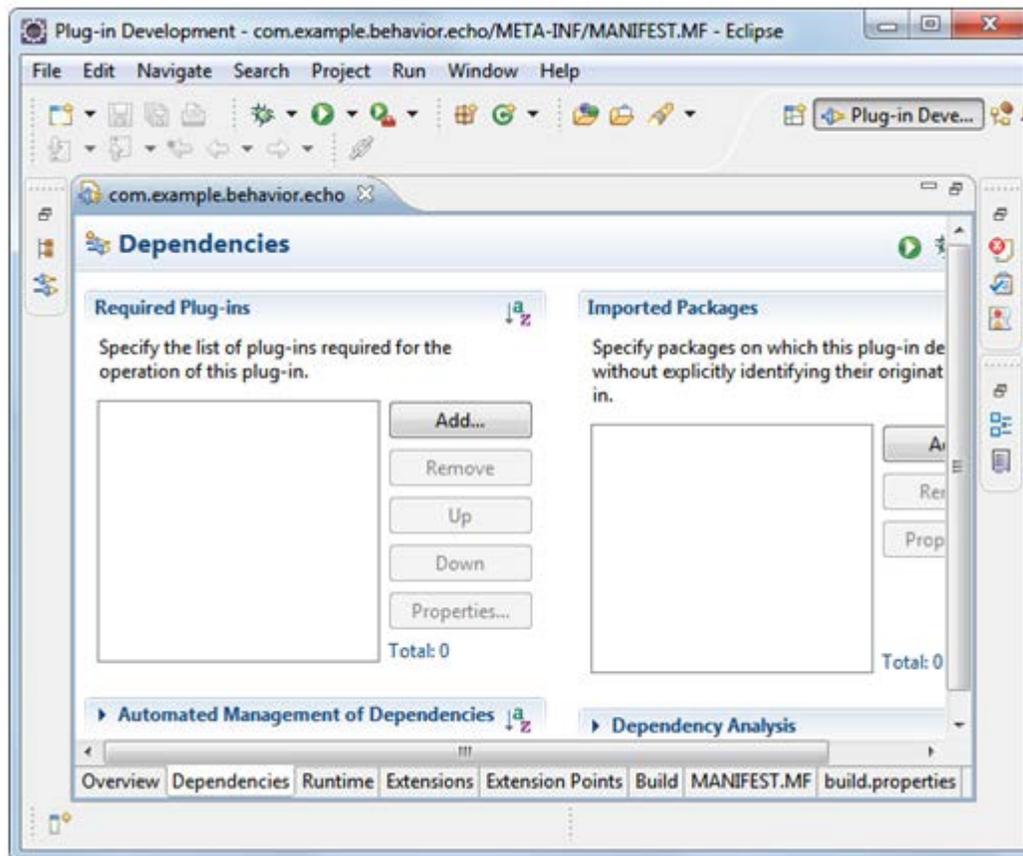
21. Click **Finish**.

NOTE: If a message about switching to Eclipse's Plug-in Development perspective is displayed, click **Yes**.

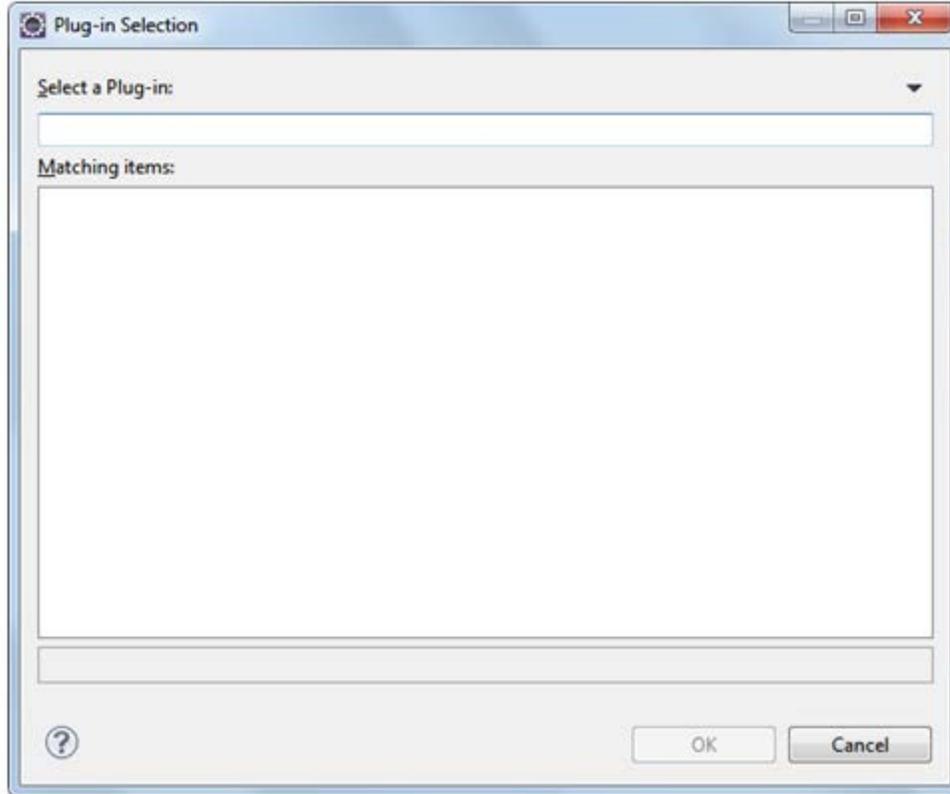
The Manifest Editor for the plug-in is displayed.



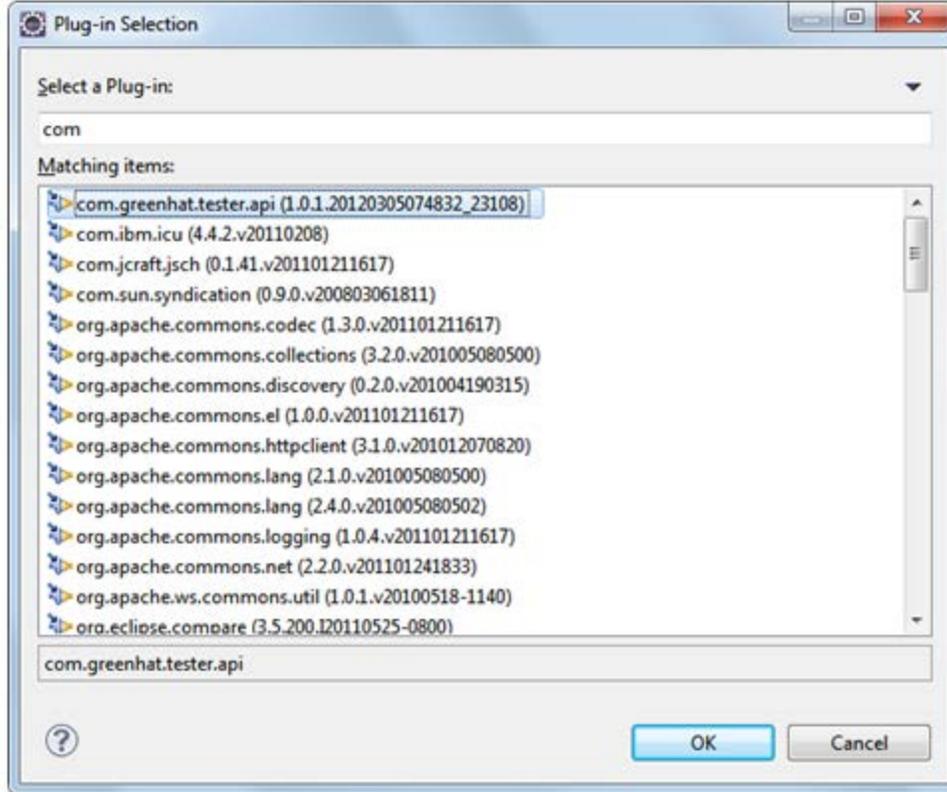
22. On the Manifest Editor window, click the **Dependencies** tab.
The **Dependencies** tab is displayed.
23. **Optional:** Maximize the Manifest Editor window.



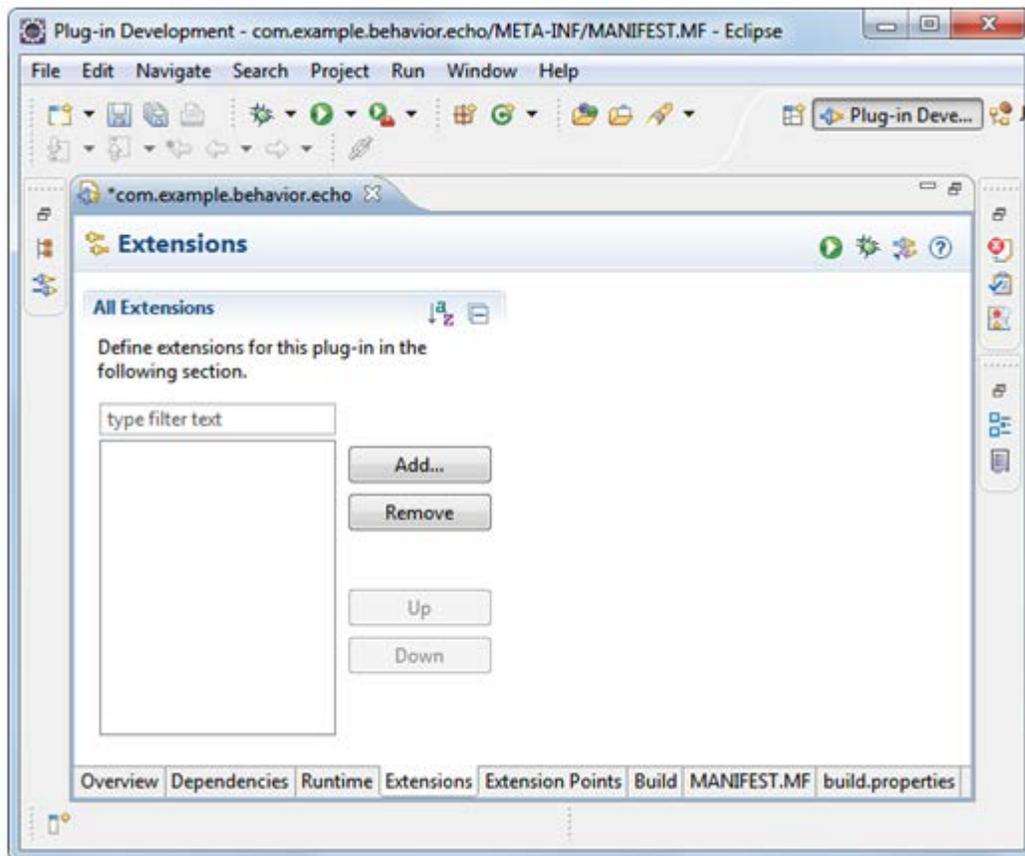
24. Under **Required Plug-ins**, click **Add**.
The Plug-in Selection dialog box is displayed.



25. In the **Select a Plug-in** field, enter at least one character of the JAR file's name to prompt the dialog box to filter the list of plug-ins under **Matching Items**.
26. Click **com.greenhat.testers.api** (*Version Number*).

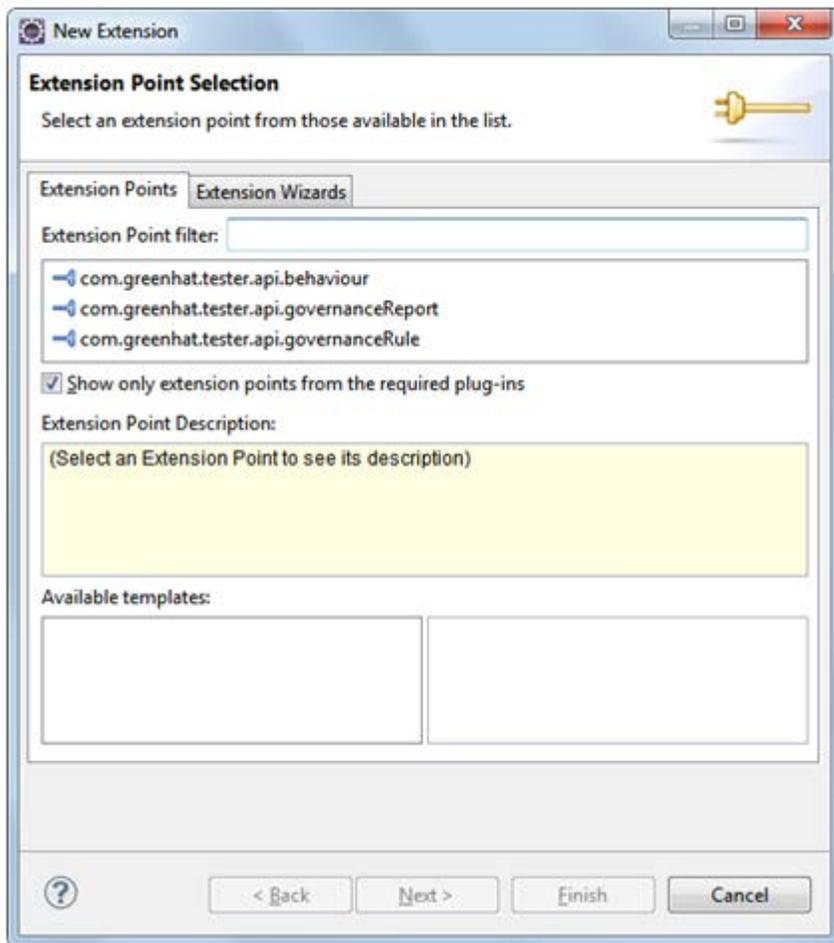


27. Click **OK**.
28. On the Manifest Editor window, click the **Extensions** tab.
The **Extensions** tab is displayed.



29. Under **All Extensions**, click **Add**.

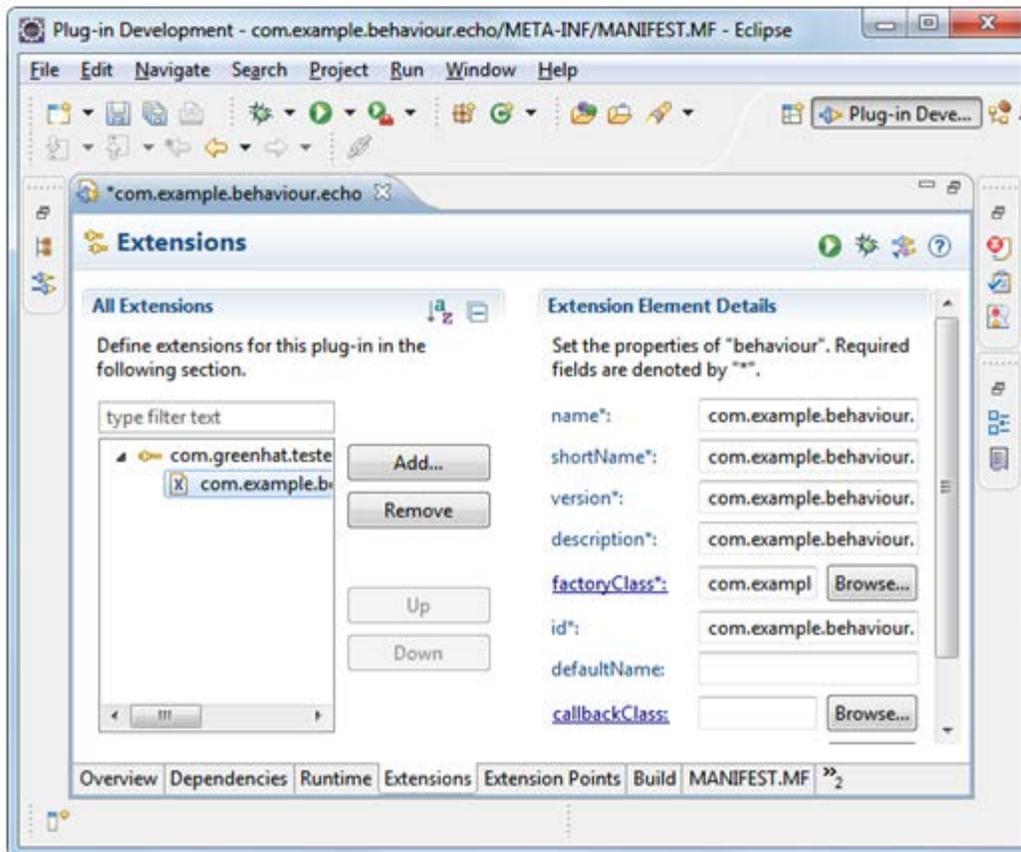
The Extension Point Selection dialog box is displayed.



30. On the **Extension Points** tab, click **com.greenhat.tester.api.behaviour**.

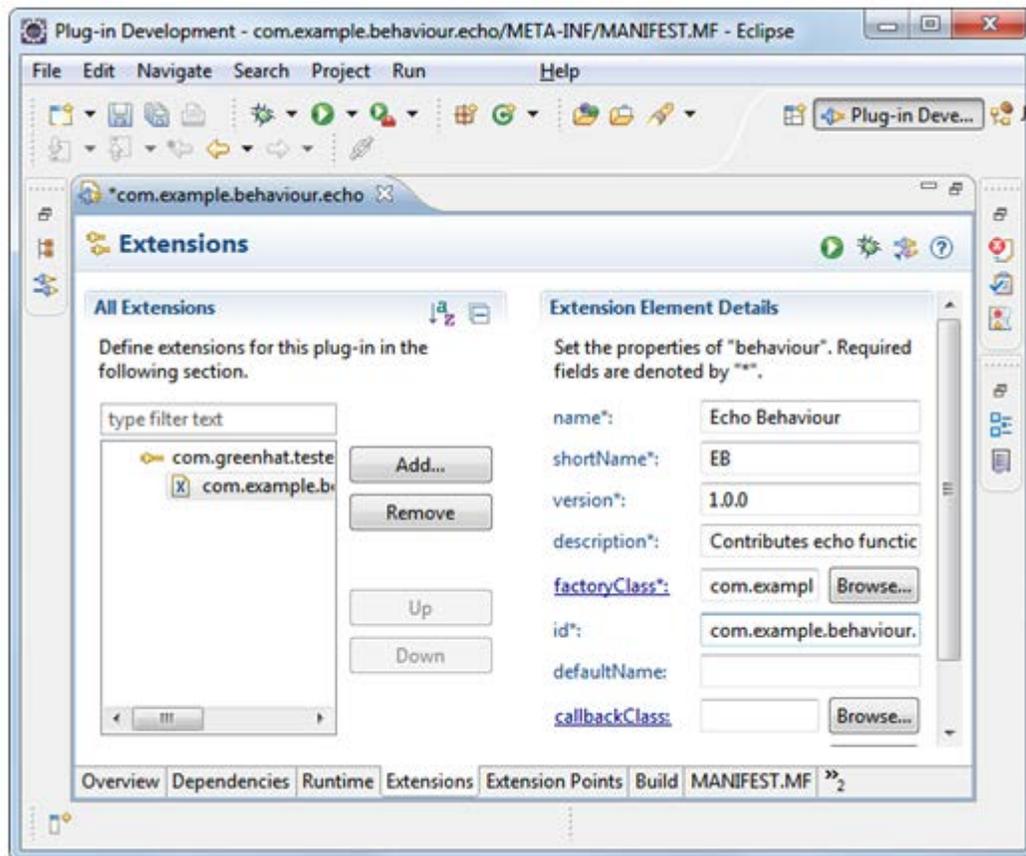
31. Click **Finish**.

The new extension point is displayed under **All Extensions**.



32. Under **Extension Element Details**, enter the details provided in the following table.

Field	Description	Suggested Value
Name	The name of the behaviour.	Echo Behaviour
Short Name	A short name that helps to identify the behaviour.	EB
Version	This is used to determine the compatibility of persisted configuration.	1.0.0
Description	Description of the purpose and capabilities of the behaviour.	Contributes echo functionality
Factory Class	This identifies the class within the selected plugin that is capable of creating instances of the behaviour at run time if BehaviourFactory is implemented.	com.example.behaviour. .echo.EchoFactory
ID	Unique identifier for the behaviour.	com.example.behaviour. .echo



33. Click **File > Save**.
34. Define suitable interfaces for the behaviour and its callback interface. (For information about this, refer to [Defining Interfaces](#).)
35. Under **Extension Element Details** on the **Extensions** tab of the Manifest Editor window, enter the name of the callback interface in the **callbackClass** field.

For example: `com.example.behaviour.echo.EchoListener`.

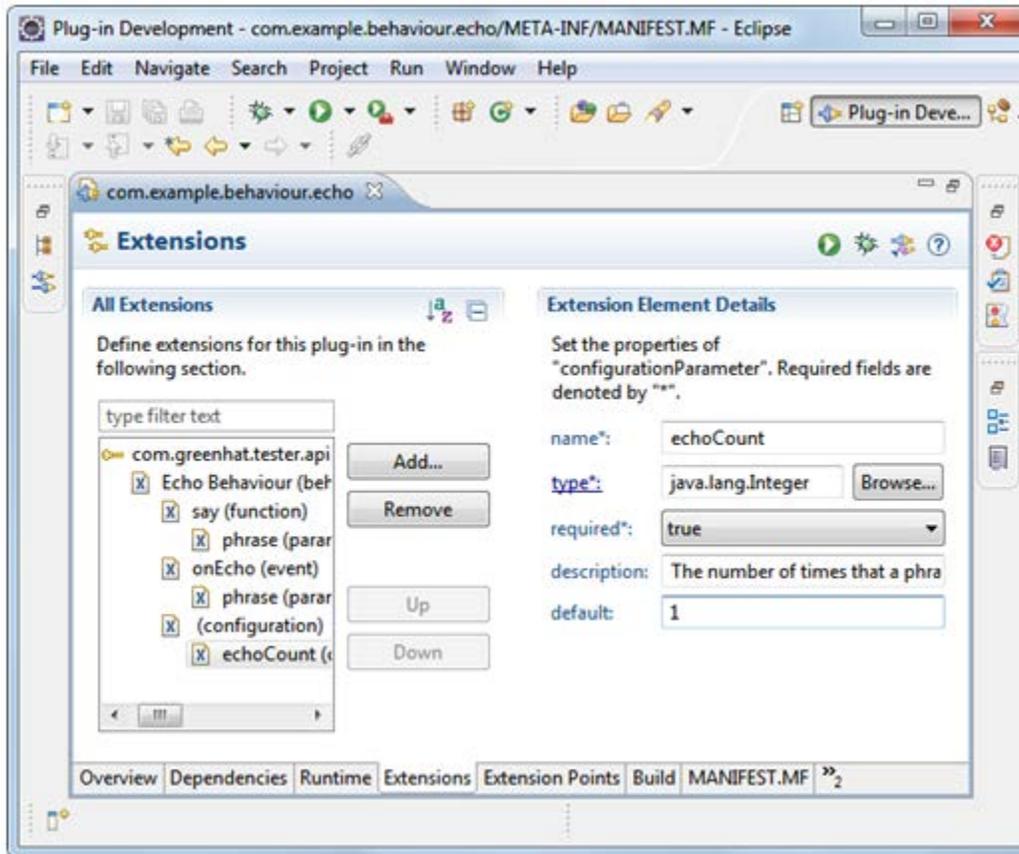
Alternatively, click the field's **Browse** button to select the class.

36. In the **behaviourClass** field, enter the name of the behaviour interface.

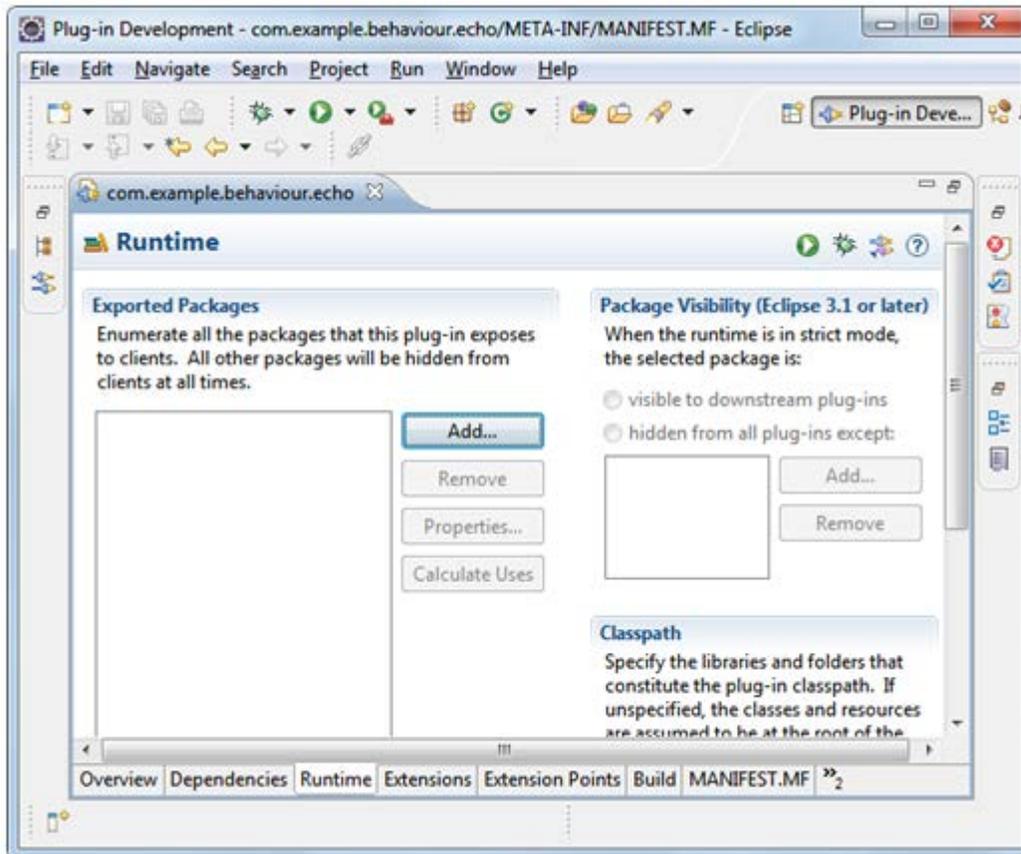
For example: `com.example.behaviour.echo.Echo`.

Alternatively, click the field's **Browse** button to select the class.

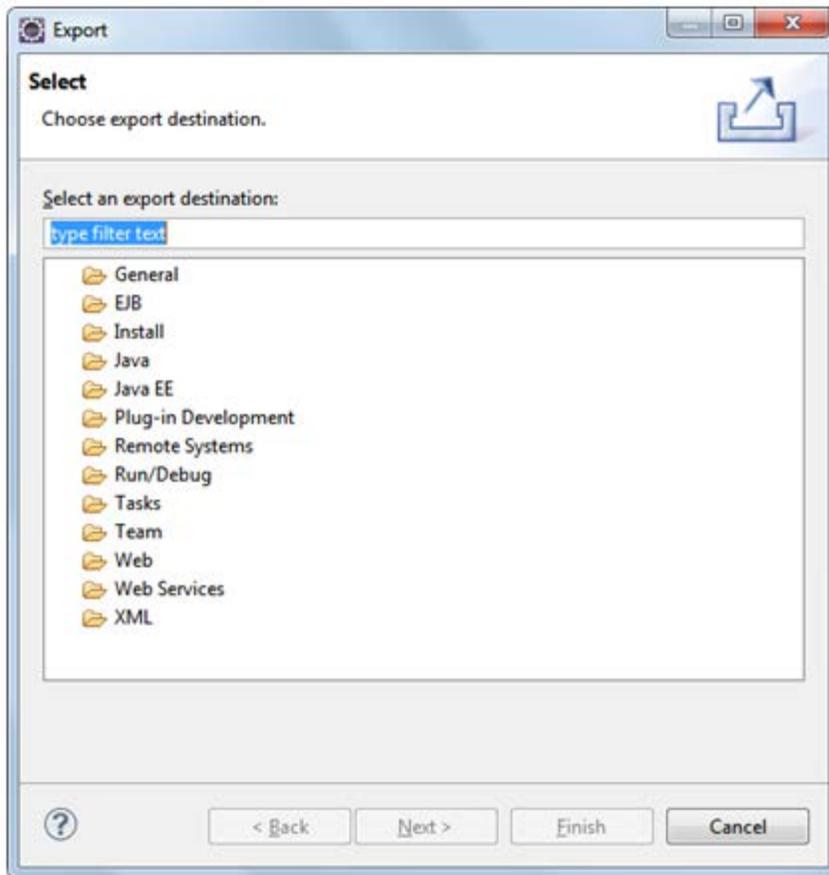
-
37. Declare the exposed function and event with their parameters, and the single configuration parameter, within the echo behaviour element.



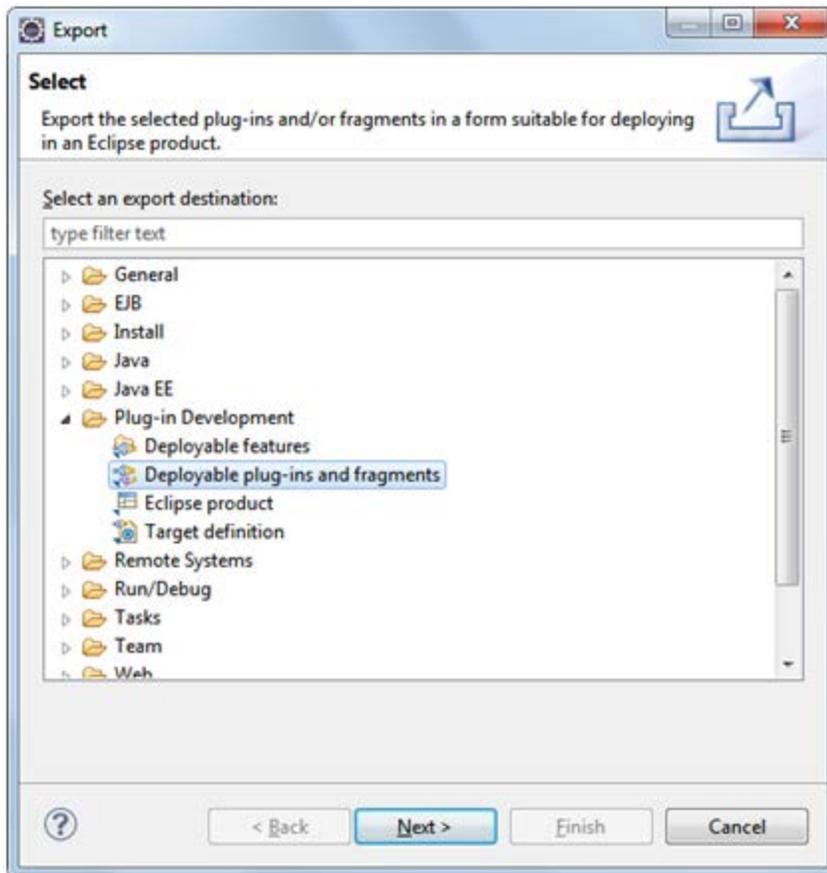
38. Click the **Runtime** tab.
The **Runtime** tab is displayed.



39. Ensure that the package containing the `EchoFactory` is exported from the bundle at run time.
 40. Click **File** > **Save**.
 41. Click **File** > **Export**.
- The Export wizard is displayed.



42. Click **Plug-in Development > Deployable plug-ins and fragments**.



43. Click **Next**.

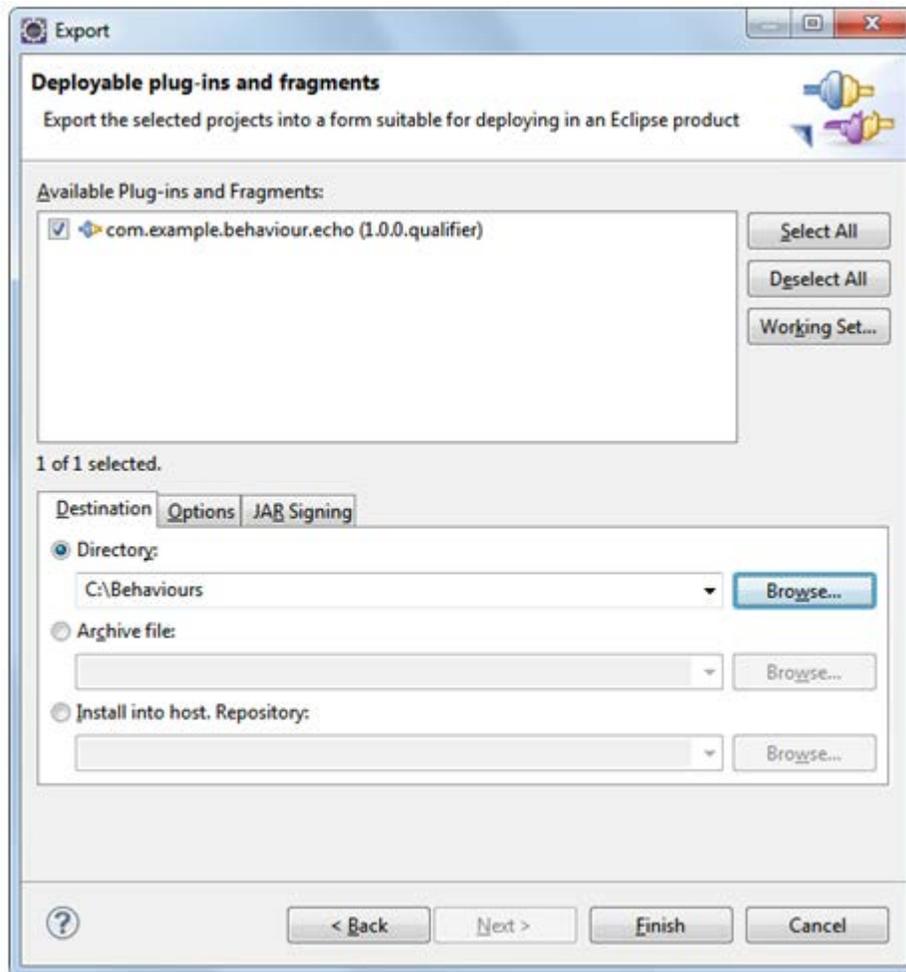
The second screen of the Export wizard is displayed.

44. Under **Available Plug-ins and Fragments**, select the plug-in's check box.

45. On the **Destination** tab, click the **Directory** option button.

46. In the **Directory** field, enter a directory path.

Alternatively, click the **Browse** button next to the **Directory** field and select the directory path.



47. Click **Finish**.

The plug-in is exported as a JAR file to a `plugins` folder in the specified directory. The name of the JAR file includes the current date and time.

48. Copy the JAR file to `<Rational Integration Tester Installation Directory>\plugins`.

This ensures that the next time when Rational Integration Tester is started, the echo behaviour will be available within the Stub Editor.

9.3 Defining Interfaces

The following sections show to define interfaces for the echo behaviour.

9.3.1 Behaviour Interface

```
package com.example.behaviour.echo;
import
com.greenhat.tester.api.behaviour.LifecycleAwareBehaviour;
public interface Echo extends LifecycleAwareBehaviour
{
    void say( String phrase );
}
```

9.3.2 Callback Interface

```
package com.example.behaviour.echo;
public interface EchoListener
{
    void onEcho( string phrase );
}
```

Rational Test Virtualization Server stubs will be able to invoke the `say(String phrase)` operation of this behaviour and can opt to handle the `onEcho(String phrase)` event.

Create a behaviour factory class and add an implementation for the new behaviour. The factory will retrieve configuration from the supplied configuration map and use it to parameterize the echo behaviour instance:

9.3.3 Behaviour Factory

```
package com.example.behaviour.echo;
import java.util.Map;
public class EchoFactory implements BehaviourFactory<Echo>
{
    public EchoFactory()
    {
    }
    @Override
    public Echo create( BehaviourServices services,
    Map<String, Object> configuration, Object callback )
    {
        String echoCountKey = "echoCountKey";
        int echoCount = 1;
        if ( configuration.containsKey( echoCountKey ) )
        {
            echoCount = (Integer)configuration.get(
            echoCountKey );
        }
        return new EchoImpl( (EchoListener)callback,
        echoCount );
    }
}
```

The echo behaviour implementation is shown in the following section.

9.3.4 Behaviour Implementation

```
package com.example.behaviour.echo.impl;
import java.util.concurrent.Executor;
public class EchoImpl implements Echo
{
    private final EchoListener callback;
    private final int echoCount;
    private final Executor executor =
        Executors.newSingleThreadExecutor();
    public EchoImpl( EchoListener callback, int
        echoCount )
    {
        this.callback = callback;
        this.echoCount = echoCount;
    }
    @Override
    public void say( final String phrase )
    {
        executor.execute( new Runnable()
        {
            @Override
            public void run()
            {
                for( int i = 0; i < echoCount; i++ )
                {
                    callback.onEcho( phrase );
                }
            }
        }
        });
    }
    //..
}
```

Appendix C: Using the RTCP Ant Client

Contents

Introduction

Accessing Rational Test Control Panel Ant Tasks

Using Rational Test Control Panel Ant Tasks

Supported Ant Tasks

Exit Codes

This appendix describes how to use the Ant Client application, which is an installation option in Rational Integration Tester Platform Pack 8.0.1 (or later) and which can be used with Rational Test Control Panel 8.0.1 (or later).

10.1 Introduction

Apache Ant is a Java library and command-line tool that is used mainly for building Java applications. (For general information about Ant, go to <http://ant.apache.org>. For information about using Rational Integration Tester with Ant, refer to *IBM Rational Integration Tester Reference Guide*.)

When installing Rational Integration Tester Platform Pack 8.0.1 (or later), you have the option to install the Rational Test Control Panel Ant Client application, which can be used in scripts to run scenarios and stubs, and lock and unlock environments in Rational Test Control Panel. (For information about installing Rational Integration Tester Platform Pack, refer to *IBM Rational Integration Tester Platform Pack Installation Guide*.)

10.2 Accessing Rational Test Control Panel Ant Tasks

The Ant tasks are distributed in *<Rational Integration Tester Platform Pack 8.0.1 (Or Later) Installation Directory>\rtcp-ant-client\com.ghc.ghTester.ant_<Version Number>.jar*.

The tasks depend on the following libraries:

- *org.apache.commons.codec_<Version Number>.jar*
- *org.apache.commons.httpclient_<Version Number>.jar*
- *org.apache.commons.httpclient_<Version Number>.jar*
- *org.apache.commons.logging_<Version Number>.jar*

Example usage of the Ant tasks is contained in the following files in *<Rational Integration Tester Platform Pack 8.0.1 (Or Later) Installation Directory>\rtcp-ant-client*:

- *lock-unlock-environment.xml*
- *start-stop-scenario.xml*
- *start-stop-stub.xml*

10.3 Using Rational Test Control Panel Ant Tasks

The following tasks are available:

- Stubs selection
- Scenario selection
- Environment locking and unlocking

The following sections describe these tasks.

NOTE: For information about using the REST API, refer to the HTML files in the `docs` folder of the Rational Test Control Panel 8.0.1 (or later) installation directory.

10.3.1 Selecting Stubs

When selecting stubs, the domain, environment, component, operation, name and version options must uniquely identify the particular stub and a particular version of that stub to start:

- If, for a given name, there is only one stub with that name in an environment, only domain, environment, and name are required.
- If there are multiple versions:
 - You can specify domain, environment and stub name to start the earliest version of that stub.
 - Alternatively, you can specify domain, environment, stub name and version to start a particular version of the stub.
- If there are multiple stubs of the same name in an environment, the component and operation options can be used to specify which one is to be started. These are the component and operation names under which the stub is available in the Rational Test Control Panel VIE dashboard. They are the same as lowest level service component (or similar) the operation that the stub is under and the operation (if any) that the stub is under in the project, respectively.
- If the command is being used in an automated script, it is suggested that component, operation (if any), and version are always used even if, at the time of writing the script, one of the alternatives above would be sufficient because using all these options will be resilient against future changes in the environment's list of available stubs and versions.

If you have locked the environment that you want to use and Rational Test Control Panel security is enabled (for information about this, refer to *IBM Rational Test Control*

Panel Installation Guide and *IBM Rational Test Control Panel System Administration Guide*), you must include your user name with any commands that you enter. However, if the security functionality of Rational Test Control Panel is disabled, you must not specify the user name option even if the environment is locked.

10.3.2 Selecting Scenarios

When selecting scenarios, the domain, environment, name and owner options must uniquely identify the scenario to start:

- If there is only one scenario in an environment, only domain and environment are required.
- If a user has only one scenario in an environment, domain, environment, and owner can be specified to uniquely identify that scenario. Otherwise domain, environment, and name are required to uniquely identify the scenario.
- If the command is being used in an automated script, it is suggested that the name is always used even if, at the time of writing the script, one of the preceding alternatives would be sufficient because using name will be resilient against future changes in the environment's list of scenarios.
- It is possible to specify domain, environment, name, and owner, which would have the effect of uniquely identifying the scenario and verifying that it was created by a specific user.

10.3.3 Locking and Unlocking Environments

Locking a Rational Test Control Panel environment prevents other users from changing the state of that environment, for example, starting or stopping stubs.

If an environment is locked, you must provide the user name that you used to lock the environment. Otherwise, any stub that you want to start will not be started.

Rational Test Control Panel environment locks do not use authentication because they are intended to facilitate collaborative working and not security. The locks check user names, thus preventing accidental changes to an environment that another user else is using.

If Rational Test Control Panel security is disabled (for information about this, refer to *IBM Rational Test Control Panel Installation Guide* and *IBM Rational Test Control Panel System Administration Guide*), you can still use the Rational Test Control Panel Ant Client to lock an environment because the user name provided to the lock-environment command does not need to be a Rational Test Control Panel user.

10.4 Supported Ant Tasks

The following sections describe the commands supported by the Rational Test Control Ant Client application.

10.4.1 Start Stub Command

Example syntax:

```
<startStub domain="<Domain name>" environment="<Environment name>" component="<Component name>" operation="<Operation name>" name="<Stub name>" version="<Stub version number>" serverURL="http://<Host name or IP address>:7819/RTCP/" />
```

Options	Description
serverURL	URL of Rational Test Control Panel.
domain	Domain name.
environment	Environment name.
component	Component the stub exists within (optional).
operation	The operation the stub exists within on the server (optional).
name	The name of the target stub to start.
version	Version of the target stub to start (optional).
attributes	Comma-separated list of agent attributes identifying the agent to run on (optional).
async	Run asynchronously; defaults to synchronous (flag).
username	The user name under which to perform the operation (optional).
haltOnFailure	Set to true to fail the Ant script if the stub fails to start (flag).
failureProperty	Property to set to true if the stub fails to start (flag).

10.4.2 Stop Stub Command

Example syntax:

```
<stopStub domain="<Domain name>" environment="<Environment name>" component="<Component name>" operation="<Operation name>" name="<Stub name>" version="<Stub version number>" serverURL="http://<Host name or IP address>:7819/RTCP/" />
```

Options	Description
serverURL	URL of Rational Test Control Panel.
domain	Domain name.
environment	Environment name.
component	Component the stub exists within (optional).
operation	The operation the stub exists within on the server (optional).
name	The name of the target stub to start.
version	Version of the target stub to start (optional).
async	Run asynchronously; defaults to synchronous (flag).
username	The user name under which to perform the operation (optional).
haltOnFailure	Set to true to fail the Ant script if the stub fails to start (flag).
failureProperty	Property to set to true if the stub fails to start (flag).

10.4.3 Start Scenario Command

Example syntax:

```
<startScenario domain="<Domain name>" environment="<Environment name>" name="<Scenario name>" serverURL="http://<Host name or IP address>:7819/" />
```

Options	Description
serverURL	URL of Rational Test Control Panel.
domain	Domain name.
environment	Environment name.
name	The name of the target scenario to start (optional).
owner	Filters the scenarios by owner (optional).
stopStubs	Available settings are as follows: <ul style="list-style-type: none">• ALL: Stop all stubs in the environment.• OPERATIONS: Stop the stubs for the operations in the scenario.• NONE (default): Leave any existing stubs running (optional).
async	Run asynchronously; defaults to synchronous (flag).
force	Attempt to first unlock the environment if another user has the lock (flag).
username	The user name under which to perform the operation (optional).
haltOnFailure	Set to true to fail the Ant script if the stub fails to start (flag).
failureProperty	Property to set to true if the stub fails to start (flag).

10.4.4 Stop Scenario Command

Example syntax:

```
<stopScenario domain="<Domain name>" environment="<Environment name>" name="<Scenario name>" serverURL="http://<Host name or IP address>:7819/" />
```

Options	Description
serverURL	URL of Rational Test Control Panel.
domain	Domain name.
environment	Environment name.
name	The name of the target scenario to start (optional).
owner	Filters the scenarios by owner (optional).
async	Run asynchronously; defaults to synchronous (flag).
username	The user name under which to perform the operation (optional).
haltOnFailure	Set to true to fail the Ant script if the stub fails to start (flag).
failureProperty	Property to set to true if the stub fails to start (flag).

10.4.5 Lock Environment Command

Example syntax:

```
<lockEnvironment domain="<Domain name>"
environment="<Environment name>" username="<User name>"
reason="<Reason>" duration="<Number of minutes>"
serverURL="http://<Host name or IP address>:7819/RTCP/" />
```

Options	Description
serverURL	URL of Rational Test Control Panel.
domain	Domain name.
environment	Environment name.
username	The user name under which to perform the operation (optional).
reason	The reason for locking the environment (optional).
duration	The number of minutes you expect to keep the environment locked (optional).
haltOnFailure	Set to true to fail the Ant script if the stub fails to start (flag).
failureProperty	Property to set to true if the stub fails to start (flag).

10.4.6 Unlock Environment Command

Example syntax:

```
<unlockEnvironment domain="<Domain name>"  
environment="<Environment name>" username="<User name>"  
serverURL="http://<Host name or IP address>:7819/RTCP/" />
```

Options	Description
serverURL	URL of Rational Test Control Panel.
domain	Domain name.
environment	Environment name.
username	The user name under which to perform the operation (optional).
haltOnFailure	Set to true to fail the Ant script if the stub fails to start (flag).
failureProperty	Property to set to true if the stub fails to start (flag).

10.5 Exit Codes

The following table lists the Ant application's common exit codes.

Code Range	Description	Examples	Description
		0	Success
1-9	(Reserved for future use.)	(Not applicable)	(Not applicable)
10-19	Server connection errors	10	RTCP connection error.
20-29	Auxiliary resource discovery errors	20	Either the domain or the environment were not found.
		21	A suitable agent could not be found.
30-39	Authentication and authorisation errors	30	The environment was locked.
		31	Unauthorized to perform the requested action.
40-49	Target discovery errors	40	The target resource (for example, stub to be started) was not found.
		41	Multiple targets were found for the specified details, but only one can be operated on.

Code Range	Description	Examples	Description
50-59	Action failure errors	50	The action requested by the user did not occur.
		51	The action requested by the user started but then failed with an error.
		52	A status (for example, "RUNNING", "STOPPING") has been retrieved from the server but was not recognized.
		53	Scenario-action-specific code for failures due to a scenario already running in the server.
90-99	Unexpected errors	90	The server reported an unexpected client error (for example, bad request).
		91	The server reported an unexpected server error (for example, internal server error).
		92	The server reported an error that the client does not understand.

NOTE: The Rational Integration Tester Command-Line Client and the Rational Test Control Panel Ant Client applications share the same exit codes. For information about the Rational Integration Tester Command-Line Client, refer to [Appendix D: Using the RTCP Command-Line Client](#).

Appendix D: Using the RTCP Command-Line Client

Contents

Introduction

Using Commands

Supported Commands

Exit Codes

This appendix describes how to use the Rational Test Control Panel Command-Line Client application, which is available in Rational Test Control Panel 8.0.1 (or later).

11.1 Introduction

If you are using Rational Test Control Panel 8.0.1 (or later), you have the option to use Rational Test Control Panel from the Rational Integration Tester Command-Line Client application (`GHTesterCmd.exe`), which enables you to use a command line to run scenarios and stubs, and to lock and unlock environments in Rational Test Control Panel.

NOTE: Rational Integration Tester supports command-line execution of Rational Integration Tester project resources. (For information about this, refer to *IBM Rational Integration Tester Reference Guide*.)

11.2 Using Commands

The following commands are available:

- Stubs selection
- Scenario selection
- Environment locking and unlocking

For more information about these commands, refer to [Appendix C: Using the RTCP Ant Client](#) (the Ant Client's tasks are the same as the Command-Line Client's commands).

NOTE: For information about using the REST API, refer to the HTML files in the docs folder of the Rational Test Control Panel 8.0.1 (or later) installation directory.

11.3 Supported Commands

The following sections describe the commands supported by the Rational Test Control Ant Client application. Information about common Ant application exit codes is also provided.

11.3.1 Start Stub Command

Start a stub on Rational Test Control Panel, optionally waiting for that stub to start before exiting.

NOTE: The command-line tasks provide an option to wait for all of the stubs in a scenario to start before returning. This is the default option but it can be disabled by specifying the “async” option. When in synchronous mode, the commands also alert users about whether a scenario was started successfully.

In synchronous mode, there is only one stub (for example, “<My Stub>”) in the environment, and it has only one version.

Synchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP address>:7819/RTCP/" --domain <Domain name> --environment <Environment name> --name <Stub name> start-stub
```

In asynchronous mode, the component, operation and version need to be specified and there may also be a need to get through an environment lock.

Asynchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP address>:7819/RTCP/" --domain <Domain name> --environment <Environment name> --component <Component name> --operation <Operation name> --name <Stub name> --version <Stub version number> --username <User name> --force --async start-stub
```

Options	Description
-serverUrl/-u	URL of Rational Test Control Panel.
-domain/-d	Domain name.
-environment/-e	Environment name.
-component/-c	Component the stub exists within (optional).

Options	Description
-operation/-o	The operation the stub exists within on the server (optional).
-name/-n	The name of the target stub to start.
-version/-v	Version of the target stub to start (optional).
-attributes/-t	Comma-separated list of agent attributes identifying the agent to run on (optional).
-async/-a	Run asynchronously; defaults to synchronous (flag).
-username/-l	The user name under which to perform the operation (optional).

11.3.2 Stop Stub Command

Stop all instances of a stub on Rational Test Control Panel, optionally waiting for that stub to stop before exiting.

In synchronous mode, there is only one stub (for example, “<My Stub>”) in the environment, and it has only one version.

Synchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP
address>:7819/RTCP/" --domain <Domain name> --environment
<Environment name> --name <Stub name> stop-stub
```

In asynchronous mode, the component, operation and version need to be specified and there may also be a need to get through an environment lock.

Asynchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP
address>:7819/RTCP/" --domain <Domain name> --environment
<Environment name> --component <Component name> --operation
<Operation name> --name <Stub name> --version <Stub version
number> --username <User name> --force --async stop-stub
```

Options	Description
-serverUrl/-u	URL of Rational Test Control Panel.
-domain/-d	Domain name.
-environment/-e	Environment name.
-component/-c	Component the stub exists within (optional).
-operation/-o	The operation the stub exists within on the server (optional).
-name/-n	The name of the target stub to start.
-version/-v	Version of the target stub to start (optional).
-async/-a	Run asynchronously; defaults to synchronous (flag).
-username/-l	The user name under which to perform the operation (optional).

11.3.3 Start Scenario Command

Start a scenario on Rational Test Control Panel, optionally waiting for it to start before exiting.

NOTE: The command-line tasks provide an option to wait for all of the stubs in a scenario to start before returning. This is the default option but it can be disabled by specifying the “async” option. If waiting, the commands must also alert users about whether a scenario was started successfully.

NOTE: The Rational Test Control Panel Command-Line Client does not support starting a scenario while another scenario is running in the specified environment. However, you can opt to “force” the start of a scenario, which will stop any other scenarios that are running. scenario first.

Synchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP address>:7819/RTCP/" --domain <Domain name> --environment <Environment name> --name <Scenario name> start-scenario
```

Asynchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP address>:7819/RTCP/" --domain <Domain name> --environment <Environment name> --name <Scenario name> --owner <Owner name> --username <User name> --force --async -stopStubs ALL start-scenario
```

Options	Description
--serverUrl/-u	URL of Rational Test Control Panel.
--domain/-d	Domain name.
--environment/-e	Environment name.
--name/-n	The name of the target stub to start.
--owner/o	Filters the scenarios by owner (optional).
--stopStubs/-t	Available settings are as follows: <ul style="list-style-type: none">• ALL: Stop all stubs in the environment• OPERATIONS: Stop the stubs for the operations in the scenario• NONE (default): Leave any existing stubs running (optional)

Options	Description
-async/-a	Run asynchronously; defaults to synchronous (flag).
-force/-f	Attempt to first unlock the environment if another user has the lock (flag).
-username/-l	The user name under which to perform the operation (optional).

11.3.4 Stop Scenario Command

Stop a scenario on Rational Test Control Panel, optionally waiting for it to stop before exiting.

Synchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP address>:7819/RTCP/" --domain <Domain name> --environment <Environment name> --name <Scenario name> --async stop-scenario
```

Asynchronous mode example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP address>:7819/RTCP/" --domain <Domain name> --environment <Environment name> --name <Scenario name> --owner <Owner user name> --username "<User name>" --async start-scenario
```

Options	Description
--serverUrl/-u	URL of Rational Test Control Panel.
--domain/-d	Domain name.
--environment/-e	Environment name.
--name/-n	The name of the target stub to start.
--owner/o	Filters the scenarios by owner (optional).
--async/-a	Run asynchronously; defaults to synchronous (flag).
--username/-l	The user name under which to perform the operation (optional).

11.3.5 Lock Environment Command

Example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP
address>:7819/RTCP/" --domain <Domain name> --environment
<Environment name> --username <User name> --duration <Number of
minutes> --reason <Reason> lock-environment
```

Options	Description
-serverUrl/-u	URL of Rational Test Control Panel.
-domain/-d	Domain name.
-environment/-e	Environment name.
-username/-l	The user name under which to perform the operation (optional).
-reason/-r	The reason for locking the environment (optional).
-duration/-t	The number of minutes you expect to keep the environment locked (optional).

11.3.6 Unlock Environment Command

Example syntax:

```
GHTesterCmd.exe --serverUrl "http://<Host name or IP  
address>:7819/RTCP/" --domain <Domain name> --environment  
<Environment name> --username <User name> unlock-environment
```

Options	Description
-serverUrl/-u	URL of Rational Test Control Panel.
-domain/-d	Domain name.
-environment/-e	Environment name.
-username/-l	The user name under which to perform the operation (optional).

11.3.7 Command Help Facility

To display help text for a specific command:

1. Open a command prompt.
2. Navigate to the Rational Integration Tester installation directory.
3. Run a command of the following format:

```
GHTesterCmd.exe help <command-name>
```

For example:

```
GHTesterCmd.exe help start-stub
```

11.4 Exit Codes

The Rational Test Control Panel Command-Line Client and the Rational Test Control Panel Ant Client applications share the same exit codes. For information about these codes, refer to [Appendix C: Using the RTCP Ant Client](#).

Glossary

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

Term	Description
Agent	Agents run stubs in the Rational Test Virtualization Server environment and are deployed on one or more computers within the environment.
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.
Message	A unit of information made up of a header consisting of meta-information and a body consisting of the message data.
Proxy	A proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers.
Server	A host computer on a network shared by more than one user.
Transport	Informally, the messaging software in use. For instance, TIBCO Rendezvous, TIBCO Active Enterprise, IBM WebSphere MQ, and HTTP.

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Trademarks and service marks

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

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

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

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

IBM[®]